
Most developments in numerical analysis reveal that computation consists mainly of mathematical tasks which a variety of users would like to delegate to a computer. Moreover, most well specified computations are hidden, that is “the human user sees neither the data nor the output. In a big calculation the data for a subtask (a Fourier transform, perhaps) will be generated by some program and the results promptly used by another. This is characteristic of introverted numerical analysis . . . Algorithms for hidden computations need to be more reliable than those for which results will be seen by a human eye” [26, p.448]. This implies that reliability, stability and time execution of many basic algebraic computations, who are necessary subtasks of complex computational strategies, should be carefully examined.

Most numerical problems are systematically reduced to matrix computation. In particular, the computational complexity of many fundamental methods for solving a system of linear equations, for calculating the maximum (or minimum) eigenvalue of a matrix or for solving a minimum problem depends typically on a matrix by vector multiplication $A\mathbf{b}$, which must be computed at each step of an iterative procedure. Often the same matrix $A$ must be applied to many different vectors $\mathbf{b}$, so a special representation of $A$, or an approximation of $A$ by another matrix with special structural properties, may reduce the cost of computation and improve the efficiency. The computational cost generally depends on the structure of the matrix, and must obey, in every case, to some fixed lower bound of arithmetic complexity.

One can view the problem of computing a matrix by vector multiplication from different points of view, depending on the model of computation, on the structure of the matrix and on the nature of input data. The principal and most general model for analyzing arithmetic complexity is the straight-line program. A straight program or straight-line algorithm is a sequence of instructions of the form $s = a \circ b$, where $\circ$ denotes one of the four basic arithmetic operations: multiplication $\cdot$, division $\div$, addition and subtraction $\pm$. Let $F[x_1, x_2, \ldots, x_n]$ be the ring of the polynomials with coefficients in a field $F$ in the variables (or indeterminates) $x_1, x_2, \ldots, x_n$, and let $F(x_1, x_2, \ldots, x_n)$ be the field of rational functions in the same variables with coefficients in the same field $F$. At each step of a straight-line program one calculates an element of $F[x_1, x_2, \ldots, x_n]$ or of $F(x_1, x_2, \ldots, x_n)$. The variables (indeterminates) and the elements of $F$ are the input data. The fact that this model of computation is sufficient to study arithmetic complexity may appear quite strange and non realistic, because the majority of interesting algorithms are computation trees with branching instructions.
(of the kind, for example, *if* \( x = 0 \) *then go to i else go to j*). For instance, the classical procedure for computing the greatest common divisor of two numbers exploits such instructions. However, as regards the point of view of algebraic complexity, one can restrict himself to straight-line programs (see [10, pp. 15 sgg.] for more details).

The idea of defining a process where at each step an element of \( F[x_1, x_2, \ldots, x_n] \) or of \( F(x_1, x_2, \ldots, x_n) \) is constructed depends on the general concept of *number field*, that is based on the idea of closure with respect to a set of operations. In fact, a set \( F \) is called a *number field* when \( F \) contains at least one element different from zero and the composition of two elements of \( F \) by one of the four basic arithmetic operations (·, ÷, ±) gives rise to an element of \( F \). The fields \( \mathbb{Q}, \mathbb{R}, \mathbb{C} \) are classical examples of number fields and it is easy to demonstrate that, if \( F \) is a number field, then \( \mathbb{Q} \subseteq F \). If \( G \) and \( F \) are number fields and \( G \subseteq F \), one says that \( F \) is an *extension* of \( G \). For example, if \( q \) is a rational number and \( r = \sqrt{-1} \), then the set \( \mathbb{Q}(r) = \{ a + rb \mid a, b \text{ rational} \} \) is a field obtained by *extension* from \( \mathbb{Q} \). Analogously, one obtains \( F(x_1, x_2, \ldots, x_n) \) by extension from \( F \) adding the indeterminates \( x_1, x_2, \ldots, x_n \) and operates in \( F(x_1, x_2, \ldots, x_n) \) by extending formally the four arithmetic operations defined in \( F \).

The number of steps in a *straight-line algorithm* for computing an algebraic expression \( E \) depends on the choice of the field \( F \). For example, if \( E = x^2 + y^2 \), 2 multiplications are required in the real field, that is

\[
s_1 = x \cdot x, \quad s_2 = y \cdot y, \quad s_3 = s_1 + s_2,
\]

whereas only one multiplication, that is \( s = (x + iy) \cdot (x - iy) \) (\( i = \sqrt{-1} \)), is sufficient in the complex field.

A first basic result stating lower bounds of complexity, due to Winograd [31], deals with computation of a set of algebraic functions that are elements of a vector \( Ay + b \), where \( A \) is a matrix \( m \times n \) with elements in \( F = G(x_1, x_2, \ldots, x_t), \) \( G \) is a field, \( x_1, x_2, \ldots, x_t \) are indeterminates with respect to \( G \), \( y \) is a vector of \( n \) indeterminates \( y_1, y_2, \ldots, y_n \) with respect to \( F \) and \( b \) is a vector whose elements vary in \( F \).

**Definition 1.** Let * denote a multiplication or a division. The operation \( f * g \) is said to be *inactive* relative to \( F \) and \( G \) if one of the following holds:

- \( g \) or \( f \in G \) and * denotes a multiplication,
- \( g \in G \) and * denotes a division,
- both \( f \) and \( g \) belong to \( F \).

Thus an inactive operation is a scalar multiplication or division (a multiplication or a division by an element of \( G \)) or an operation involving only
rational functions of the variables $x$, that is only elements of $F$. An operation that is not inactive is called *active*.

**Definition 2.** Let $G^m$ and $G^m(x)$ be the spaces of vectors with $m$ elements in $G$ and in $G(x)$, respectively. A set of $q$ vectors $v_1, v_2, \ldots, v_q$ in $G^m(x)$ are said to be linearly independent modulo $G^m$ (or modulo $G$) if there is no nontrivial sequence $c_1, c_2, \ldots, c_q$ in $G$ such that $\sum_{i=1}^q c_i v_i \in G^m$.

The following theorem states a lower bound of active operations required to compute a matrix-vector product $A y$, plus a residual vector $b$ of $F^m$ in the field $F(y_1, y_2, \ldots, y_n)$.

**Theorem 1.** [31] Let $A$ be a matrix $m \times n$ with elements in $F$ and let $G$ have infinitely many elements. If $A$ has $q$ columns linearly independent modulo $G^m$, then $q$ active operations are necessary to compute $A y + b$.

**Proof**
Let $A$ be an algorithm computing $A y + b$ in $F(y_1, y_2, \ldots, y_n)$. It is easy to prove by induction on $k$ that if the first $k$ steps do not involve active operations, then, at each of these steps, the most general expression computed has the form

$$f + \sum_{i=1}^n g_i y_i$$

where $f \in F, g_i \in G$.

We prove the theorem by induction on $q$. If $q = 1$, then there exist $r, s$ such that $A r, s$ does not belong to $G$. As the algorithm $A$ computes $A y + b$, in a step of $A$ we obtain an expression of the form

$$\sum_{j=1}^n A_{r,j} y_j + b_r.$$

If no active operation appears in $A$, then the above expression is identical to an expression that has the form (1), a contradiction. Thus at least one active operation is necessary when $q = 1$.

Let the theorem hold for $q - 1$ and let $A$ be an algorithm computing $A y + b$, where $A$ has $q$ columns independent modulo $G$, in less than $q$, say in $q - 1$ active operations. Let $k$ be the smallest integer such that an active operation occurs at step $k$. The result $r_k$ of this operation has the form

$$r_k = (f + \sum_{i=1}^n g_i y_i) \ast (f' + \sum_{i=1}^n h_i y_i),$$

where $\ast$ is a multiplication or a division, for some $g_i, h_i \in G, i = 1, 2, \ldots, n$, and $f, f' \in F$. Either one of the $h_i$ or else one of the $g_i$ is not zero, otherwise $r_k \in F$ and the operation $\ast$ is inactive. With no loss of generality, assume that $g_n \neq 0$ and, since scalar operations are not counted, $g_n = -1$. Let
$g \in G$ be such that if we substitute $g - f - \sum_{i=1}^{n-1} g_i y_i$ for $y_n$, the resulting algorithm has not divisions by zero. This is possible because $G$ has infinitely many elements, $A$ has a finite number of steps and at each steps a fraction is computed of polynomials in $y_n$, with only a finite number of substitutions for $y_n$ causing divisions by zero.

By replacing $g - f - \sum_{i=1}^{n-1} g_i y_i$ for $y_n$ the first active operation $\ast$ becomes inactive (if $\ast$ is a division and the coefficient $h_n$ is not zero, then make the substitution of $y_n$ in the divisor, so $\ast$ becomes a scalar division). Then we obtain a new algorithm $A'$ which computes with only $q - 2$ active operations $A'y' + b'$, where $A'$ is a matrix $m \times (n - 1), y'$ has $n - 1$ elements $y_1, y_2, \ldots, y_{n-1}$, $a'_j = a_j - g_j a_n, j = 1, 2, \ldots, n - 1$ and $b' \in F^m$.

To gain the thesis we prove that the matrix $A'$ has $q - 1$ columns linearly independent modulo $G$, so we have a contradiction with the induction hypothesis. Assume that $A'$ has not $q - 1$ columns linearly independent modulo $G$ and let $\{a_{i1}, \ldots, a_{iq}\}$ be a set of $q$ columns of $A$ linearly independent modulo $G$. There exist $q - 1$ elements of $G, d_1, \ldots, d_{q-1}$ not all zero such that

$$\sum_{j=1}^{q-1} d_j a'_{ij} = \sum_{j=1}^{q-1} d_j a_{ij} + k_1 a_n \in G^m$$

(2)

where $k_1$ is a scalar different from zero, because the columns $a_{ij}, j = 1, 2, \ldots, q$ are linearly independent modulo $G$. With no loss of generality, let $d_1 \neq 0$. For the same reason there exist $q - 1$ elements of $G, e_2, \ldots, e_q$, not all zero, such that

$$\sum_{j=2}^{q} e_j a'_{ij} = \sum_{j=2}^{q-1} e_j a_{ij} + k_2 a_n \in G^m$$

(3)

where $k_2$ is a scalar different from zero. Now eliminate $a_n$ by a linear combination of (2) and (3), so to obtain a linear non trivial combination, with the coefficient of $a_{i1}$ different from zero, of the columns $a_{ij}, j = 1, 2, \ldots, q$, belonging to $G^m$, a contradiction.

From now on the symbols m/d and a/s will be used to denote multiplications/divisions and additions/subtractions, respectively.

**Corollary 1.** Every algorithm for evaluating a polynomial $\sum_{i=0}^{n} a_i x^i$ in a point $x$, where $x, a_0, a_1, \ldots, a_n$ are indeterminates with respect to a field $G$, requires at least $n$ active operations. Thus Horner’s rule minimizes the number of m/d for computing a polynomial.

**Proof**

Let $F = G(x)$. Let $A$ be the matrix formed by the unique vector whose elements are $1, x, x^2, \ldots, x^n$. Except 1, all elements of this vector are linearly
independent modulo $G$. The assertion follows from the equality $\sum_{i=0}^{n} a_i x^i = A a$, where $a$ is the vector whose elements are $a_0, a_1, \ldots, a_n$.

**Corollary 2.** Every algorithm for computing a matrix by vector product $Yx$, where $Y = (y_{ij})$ is a matrix $p \times q$ and $x = [x_1 \ldots x_q]^T$ is a $q$–vector $(y_{ij}, x_j = \text{indeterminates with respect to a field } G)$, requires at least $pq$ active multiplications. Thus the standard algorithm for a matrix by vector multiplications minimizes the number of active m/d.

**Proof**

Let $F = G(x_1, \ldots, x_q)$. Let $A$ be the $p \times pq$ matrix whose element $A_{ij}$ is $x_k$ if $j = iq + k, 1 \leq k \leq q$, and 0 otherwise. Then $Yx = Ay$ where $y = [y_{11} \ldots y_{1q} \ y_{21} \ldots y_{2q} \ldots y_{pq}]^T$. The assertion follows from the fact that the columns of $A$ are linearly independent with respect to $G$.

**Exercise 1.** Let $Q \subset G \subset \mathbb{C}$ and assume that instead of $Ay + b$, we want to compute $m$ functions $u_1, \ldots, u_m$ such that $\|u_i - \sum_{j=1}^{n} a_{ij}y_j - b_i\|_{\infty} \leq a$, for some $0 \leq a < \infty$. Modify the proof of the Theorem 1 to show that at least $q m/d$ are necessary to compute the functions $u_1, \ldots, u_m$.

The following Remarks add some necessary comments to the previous Winograd’s theorem (see [3], [9], [10], [27], [31]):

**Remark 1.** The field $G$ in Theorem 1 can be chosen freely, but the larger $G$ is the fewer sets of columns of $A$ are independent mod $G$ and then the fewer active operations are counted.

**Remark 2.** The number of active m/d required to compute a polynomial $\sum_{i=0}^{n} a_i x^i$, in the Corollary 1, can not be reduced through possible preliminary operations on the only indeterminate $x$. Analogously, The number of m/d required to compute $Yx$, in the Corollary 2, can not be reduced by operations on the only indeterminates $x_j$.

**Remark 3.** In the proof of the Corollaries 1 and 2 we have assumed that multiplication is commutative over the indeterminates, but this assumption is not really necessary. In fact one can prove a symmetric form of Theorem 1 for computing a vector $y^T A + b^T$.

The number of active operations required to compute a polynomial $p(x) = \sum_{i=0}^{n} a_i x^i$ can not be reduced by preliminary operations on $x$, but the following example (due to Todd), shows that by performing rational operations on the coefficient $a_i$, without counting these operations, can reduce the number of m/d. Here the operations on $a_i$ have the meaning of a preprocessing, a kind of preconditioning of $p(x)$ in case the same polynomial $p(x)$ has to be computed in many points $x$. The concept of preconditioning of the coefficients was introduced by Motzkin [25].

**Example 1.** Consider the polynomial $\sum_{i=0}^{4} a_i x^i$ and define
\[ y = (x + \alpha_0)x + \alpha_1, \quad z = ((y + x + \alpha_2)y + \alpha_3)\alpha_4 \]

Where the \( \alpha \) are parameters. The explicit formula of \( z \) is

\[ z = \alpha_4 x^4 + \alpha_4 (2\alpha_0 + 1)x^3 + \alpha_4 (2\alpha_1 + \alpha_2 + \alpha_0^2 + \alpha_0)x^2 + ((2\alpha_0 + 1)\alpha_1 + \alpha_0 \alpha_2)\alpha_4 x + (\alpha_1^2 + \alpha_1 \alpha_2)\alpha_4 + \alpha_3 \alpha_4. \]

In order that \( z = p(x) \) the parameters \( \alpha \) must satisfy the conditions

\[ \alpha_4 = a_4, \quad \alpha_0 = \frac{1}{2} \frac{a_3}{a_4} - 1 \]

and also

\[ \alpha_4 (2\alpha_1 + \alpha_2 + \alpha_0^2 + \alpha_0) = a_2, \quad ((2\alpha_0 + 1)\alpha_1 + \alpha_0 \alpha_2)\alpha_4 = a_1. \]

To calculate \( \alpha_1, \alpha_2 \) from the last two equalities observe that the matrix

\[
\begin{bmatrix}
2 & 1 \\
2\alpha_0 + 1 & \alpha_0
\end{bmatrix}
\]

has determinant equal to \(-1 \neq 0\). Now by equating the terms of degree 0 we compute \( \alpha_3 \). If we do not count operations to calculate the functions \( \alpha \), then we compute \( p(x) \) with only 3 multiplications and 5 additions.

The idea of preprocessing can be extended to matrix by vector multiplication \( Yx \) when the same matrix \( Y \) has to be multiplied by many vectors \( x \). The preprocessing is based on a new procedure for inner vector multiplication. For any vector \( z = [z_1, z_2, \ldots, z_n]^T \), with \( n \) even, define

\[ W(z) := z_1 z_2 + z_3 z_4 + \ldots + z_{n-1} z_n. \]

For each row \( y_i \) of the matrix \( Y \) compute \( W(y_i) \) and if \( y = y_i \) then compute the inner vector product \( y \times x \) as follows:

\[ y \times x = (y_1 + x_2)(y_2 + x_1) + (y_3 + x_4)(y_4 + x_3) + \ldots + (y_{n-1} + x_n)(y_n + x_{n-1}) - W(y) - W(x). \]

Thus, if we not count active multiplications involving only the elements \( y_{ij} \) of \( Y \), the vector \( Yx \) can be computed with \( \frac{1}{2} n^2 + \frac{1}{2} n \) multiplications.

**Exercise 2.** Is the previous preprocessing for inner vector multiplication numerically stable? Try to compute with \( n = 2 \), basis \( B = 10, 4 \) significant digits, \( x_1 = x_2 = (.1000)10^3 \) and \( y_1 = y_2 = (.1000)10^{-1} \). Find a simple strategy to avoid instability.

**Exercise 3.** Exploit the previous algorithm for inner vector multiplication to the case where \( n \) is odd. How many multiplications are saved by applying the same algorithm to matrix by matrix multiplication? Is the
same algorithm useful to reduce the asymptotic complexity of matrix by matrix multiplication by the same strategy used by Strassen in his celebrated article [29]?

**Exercise 4.** Apply the previous algorithm for matrix by vector multiplication to Gaussian elimination applied to a linear system \( Ax = b \) when \( n = k \cdot l \), so that \( A \) can be partitioned into a \( l \times l \) matrix whose entries are \( k \times k \) matrices. How many multiplications can be saved?

The following theorem (we omit the proof) states a lower bound of complexity for computing a matrix by vector product with preconditioning on the matrix.

**Theorem 2.** [31] Every algorithm for computing \( Ay \), where \( A \) is a \( p \times q \) matrix and \( y \) is a \( q \)-vector, requires at least \( \lfloor \frac{1}{2} pq \rfloor \) m/d which do not depend only on the entries of \( A \) (or only on the entries of \( y \)).

An analogous result, due to Motzkin [25], holds for polynomials with a preprocessing on the coefficients:

**Theorem 3.** [25] Every algorithm for computing a polynomial \( p(x) = \sum_{i=0}^{n} a_i x^i \) requires at least \( \lceil \frac{1}{2} n \rceil \) m/d which do not depend only on the coefficients \( a_i \) (or only on \( x \)).

If a matrix \( A \) has a special structure, then the complexity of a matrix by vector multiplication \( Ab \) may be reduced with respect to the lower bound stated by Winograd’s theorem. Also, the relevance of more efficient formulas for \( A \) or \( A^{-1} \) increases dramatically in situations in which we apply the same operator \( A \) to many vectors \( b \), in which a linear system \( Ax = b \) has to be solved or, equivalently, the corresponding matrix by vector product \( A^{-1}b_j \) has to be computed for different vectors \( b_j \). We find some instances of these situations in matrix by matrix multiplications, in the main iterative algorithms for solving systems of linear equations (Jacobi, Gauss-Seidel, Richardson-Euler, CG, GMRES [1, p.539]), in the power method for computing the maximum eigenvalue of a matrix and in the iterative procedures for unconstrained minimum problems.

Some different approaches are possible in order to define the *structure* of a matrix, but no final formalization of this concept of structure can be definitely stated. Moreover, a relationship between the structure of a matrix and his informational content should be investigated. In fact, numerical work (like numerical treatment of elliptic problems, of time series or minimum problems) is often concerned with operations on matrices belonging to special classes. Within a class the generic matrix is often specified by a number \( k \) of parameters less than the number of elements. This fact justifies the introduction of an idea of informational content of a matrix. A measure of *informational content*, as was initially proposed by Forsythe [17], could be the amount of memory required to store the matrix as compactly as possible.
in a computer. But a very notion of informational content should be studied through the following two main criteria:

- Best way of representing a matrix in a computer (Forsythe [17]).
- Computational complexity of matrix operations [8], [5].

Moreover one could usefully compare a possible notion of informational content of a matrix to work of Chaitin [13] and of Kolmogorov [21] on the informational content of a string of 0 and 1, which is defined as the complexity, that is as the minimum length of a program that generates the string.

The following different techniques (the list is far to be exhaustive) are able, in different ways, to clarify the informal ideas of structure and of informational content of a matrix:

- investigation of the structure of the inverses of tridiagonal and band matrices, semi-separable matrices [8], [12], [13], [15], [19];
- mosaic-skeleton approximation (Tyrtyshnikov [30]);
- bilinear programs, decomposition of three-dimensional arrays, tensor rank of a class of matrices;
- structure of matrices that can be reduced to diagonal form through fast transforms, perturbations of such matrices by matrices of small rank;
- displacement rank, applications to Toeplitz matrices [20];
- matrix by vector product in a linear model (Savage [28], Morgenstern [24]).

Regarding the first point, the structure of a tridiagonal matrix $T_n$ and of its inverse, we give only a simple example, for $n = 4$:

$$T_n = \begin{bmatrix} u_1 & v_1 & 0 & 0 \\ w_1 & u_2 & v_2 & 0 \\ 0 & w_2 & u_3 & v_3 \\ 0 & 0 & w_3 & u_4 \end{bmatrix}, \quad T_n^{-1} = \begin{bmatrix} a_1 b_1 & a_1 b_2 & a_1 b_3 & a_1 b_4 \\ c_2 d_1 & a_2 b_2 & a_2 b_3 & a_2 b_4 \\ c_3 d_1 & c_3 d_2 & a_3 b_3 & a_3 b_4 \\ c_4 d_1 & c_4 d_2 & c_4 d_3 & a_4 b_4 \end{bmatrix},$$

where $a_i b_i = c_i d_i$ and $a_1 = c_1 = 1$. In general, for every $n$, both $T_n$ and $T_n^{-1}$ are defined by $3n - 2$ parameters. For $n = 2^k$, divide $T_n^{-1}$ in 4 block of dimension $\frac{n}{2} \times \frac{n}{2}$. In fact we have

$$T_n^{-1} = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix}.$$
Clearly $T_{11}, T_{22}$ are tridiagonal, and $T_{12}, T_{21}$ have rank $= 1$. But $\text{rank}(C) = 1$, that is $C = st^T$, implies that $Cy = (st^T)y = s(t^Ty)$, which requires $2n$ multiplications. Then a product $T_n^{-1}$ by a vector can be computed with $M(n) = 2M(\frac{n}{2}) + 2(\frac{n}{2} + 2) = 2n \log_2 n$ multiplications. We are able to prove that $3n - 2$ multiplications are necessary [33].

**Research problem.** Find the minimum multiplicative complexity of the set of trilinear forms defining a product $T_n^{-1}y$ when $T_n$ is symmetric.

### 2. Bilinear programs. Tensor Rank.

This section will be mainly concerned with the optimal computation of a set of bilinear forms. Studying the multiplicative complexity of a set $\{f_h(a, b)\}$ of bilinear forms will be a strategy for finding a best representation of a matrix $A$ in order to minimize the computational cost of a product matrix by vector $Ab$. First recall the following

**Definition 3.** A function $f(a, b) \in F[a, b]$ of the indeterminates $a_1, \ldots, a_m, b_1, \ldots, b_n$ is a bilinear form in $a, b$, when it can be expressed in the form

$$f(a, b) = a^T M b$$

where $M$ is a matrix of $F^{m \times n}$.

Matrix by matrix, matrix by vector multiplications and polynomial multiplication define sets of bilinear forms and the complexity of these multiplications depends on the structure of matrices $M$. Here the structure usually depends on the fact that the generic element of a space of $n \times n$ matrices may be specified by a number $k$ of parameters with $k < n^2$. In this case we may say that the informational content is less than $n^2$.

We may first conceive a space $C^k_n$ of matrices $n \times n$ of informational content $k$ as a manifold of dimension $k, k \leq n^2$, in the space of dimension $n^2$ of all real $n \times n$ matrices. The case when $C^k_n$ is an algebra spanned by $k$ linearly independent matrices $J_i, i = 1, 2, \ldots, k$, is especially significant. Let

$$A = \sum_{i=1}^{k} a_i J_i, \quad B = \sum_{j=1}^{k} b_j J_j, \quad J_i J_j = \sum_{h=1}^{k} t_{ijh} J_h,$$

so that the 3-dimensional array (tensor) $t_{ijh}$ defines the multiplication table of the algebra spanned by the set $\{J_i\}$. Then the product $C = AB = \sum_{h=1}^{k} c_h J_h$ of two elements of the algebra is expressed as

$$C = AB = \sum_{i, j=1}^{k} a_i b_j J_i J_j = \sum_{h=1}^{k} \left[ \sum_{i, j=1}^{k} t_{ijh} a_i b_j \right] J_h = \sum_{h=1}^{k} f_h(a, b) J_h$$
where the coefficients \( c_h = f_h(a, b) = \sum_{i,j=1}^k t_{ijh} a_i b_j \) are a set of bilinear form in the variables (or indeterminates) \( a_i, b_j \). That is the coefficients \( c_h \) of the representation of \( AB \) in the basis \( \{ J_i \} \) are identical to \( f_h(a, b) \). Hence the last formula exhibits possible reductions in computational complexity.

In fact, define \( \text{rk}(t_{ijh}) \), the rank of the tensor \( t_{ijh} \) as follows:

**Definition 4.** \( \text{rk}(t_{ijh}) \), the rank of the tensor \( t_{ijh} \) in a field \( F \), is the minimum integer \( q \) such that

\[
t_{ijh} = \sum_{r=1}^q u_{ri} v_{rj} w_{rh}
\]

for \( 3q \) vectors \( u_r, v_r, w_r, r = 1, 2, \ldots, q \) with elements in \( F \). The above formula for \( t_{ijh} \) gives its \( 3 \)-adic decomposition, so a \( 3 \)-adic decomposition is defined by the set of three matrices \( \{ W, U, V \} \). This definition naturally extends to \( n \)-adic decompositions of an \( n \)-array \( t_{i_1i_2\ldots i_n} \) [22].

If the rank of \( t_{ijh} \) is \( q \), then the coefficients \( c_h \) of \( C = AB \) are

\[
c_h = \sum_{i,j=1}^k a_i b_j \sum_{r=1}^q u_{ri} v_{rj} w_{rh} = \sum_{r=1}^q w_{rh} \left( \sum_{i=1}^k a_i v_{ri} \right) \cdot \left( \sum_{j=1}^k b_j v_{rj} \right)
\]

i.e. \( q \) non-scalar multiplications are sufficient to compute \( c_h \). Then the rank of the tensor \( t_{ijh} \) of the multiplication table of \( C^k \) states an upper bound of the multiplicative complexity of the product of two elements of \( C^k \).

**Remark 4.** The problem of computing a set of \( p \) bilinear forms \( f_h(a, b), h = 1, 2, \ldots, p \), where \( a \) and \( b \) are vectors of \( m \) and of \( n \) indeterminates, respectively, can also be formulated as the multiplication in an algebra [16]. In fact, if \( f_h(a, b) = \sum_{i=1}^m \sum_{j=1}^n t_{ijh} a_i b_j \) is a bilinear map \( f : F^m \times F^n \to F^p \), then choose a basis \( \{ e_1, e_2, \ldots, e_s \} \), with \( s \geq \max(m, n, p) \) and define the multiplication table

\[
e_i e_j = \begin{cases}
\sum_{h=1}^p t_{ijh} e_h & \text{if } 1 \leq i \leq m, 1 \leq j \leq n, \\
0 & \text{otherwise}.
\end{cases}
\]

The algebra resulting from this construction should not be confused with algebras which may or may not already exist [16]. As an example consider the bilinear forms defined by the matrix-vector product

\[
\begin{bmatrix}
a_1 & a_2 & a_3 \\
a_4 & a_1 & a_2 \\
a_5 & a_4 & a_1
\end{bmatrix}
\times
\begin{bmatrix}
b_1 \\
b_2 \\
b_3
\end{bmatrix}.
\]

The 5-dimensional algebra where multiplication is equivalent to computing the above matrix by vector product is defined by the following multiplication table:
Exercise 4. Consider the bilinear forms defining the multiplication of two symmetric $2 \times 2$ matrices. The symmetric matrices $2 \times 2$ form a 3-dimensional space, but they are not a subalgebra of the algebra of $2 \times 2$ matrices. In spite of this define a 4-dimensional algebra in which multiplication is equivalent to multiplying two symmetric $2 \times 2$ matrices.

We will prove that in the general (non commutative) case the rank of the tensor $t_{ijk}$ defines a lower bound of multiplicative complexity of the bilinear forms $f_h(a, b) = \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ijk}a_ib_j$. This implies that, if $\text{rk}(t_{ijk}) = q$, then $q$ non scalar multiplications are necessary and sufficient to compute all $f_h(a, b)$ if the indeterminates $a, b$ do not commute with each other.

In order to gain this result a preliminary definition of a model of computation is necessary. The useful strategy is to restrict gradually the general notion of straight line program, in order to exploit the specific nature of the functions to be computed, that is bilinear forms. We will prove that the most general model to compute $f_h(a, b)$, $1 \leq h \leq p$, is a $F$–bilinear program, which can be informally described as follows:

Definition 5. An $F$–bilinear program is a procedure organized in three stages:

- compute linear combinations of the indeterminates $a$ and $b$,
- compute a number of products whose factors are the linear combinations of the previous point,
- compute required linear combinations of the products of the previous point.

All linear combinations have coefficients in $F$, so that the only non-scalar multiplications appear at the second stage.

A first justification of this model is that divisions, in a straight-line program computing $\{f_h(a, b)\}$, $1 \leq h \leq p$, can be simulated by a set of additions and multiplications. This fact is stated in the following Theorem 5. As regards the complexity of a bilinear program we may count all arithmetic operations or we may be concerned only with non-scalar operations. The point of view that non-scalar multiplications dominate the complexity has many justifications. First of all, the variables $a$ and $b$ may assume values outside $F$, and may be polynomials or matrices (the Strassen’s bilinear
program for computing the product of two matrices $2 \times 2$ with 7 multiplications can be extended to block matrices [29]). Moreover, a lower bound of complexity given in terms of non-scalar multiplications is stronger than a lower bound in terms of multiplications *tou-cour.* Now we will prove that in general (when commutativity does’nt hold), dealing with non-scalar multiplications, we should consider only programs whose non-scalar multiplications have the simple form $l_1(a) \times l_2(b)$, where $l_1$ and $l_2$ denote linear combinations over $F$.

In the following theorem we first want to show how to simulate a computation in $F[x_1, \ldots, x_n]$ by a computation in $F[x_1, \ldots, x_n]$ modulo $J$, where $F$ is a field and $J$ is the ideal generated by $\{x_i x_j : 1 \leq i \leq j \leq k \leq n\}$, that is dropping all terms of degree $\geq 3$. Moreover, reducing the computation modulo $J$ does not modify asymptotically the total number of operations.

**Theorem 4.** [9, p. 35] Let $S = \{f_h(x)\}$ be a set of polynomials in the variables $x_1, \ldots, x_n$, where each $f_h(x)$ has degree $\leq 2$. Suppose an algorithm $A$ computes $S$ in $F(x_1, \ldots, x_n)$ with $k_1$ additions/subtractions, $k_2$ scalar and $k_3$ non-scalar multiplications. Then there is an algorithm $A'$ that computes $S$ with $k_3$ non-scalar multiplications of the form $l_1(x) \cdot l_2(x)$, where both $l_1$ and $l_2$ denote linear combinations of $x_1, \ldots, x_n$, and with less than $9(k_1 + k_2 + k_3)$ total operations.

**Proof.** The algorithm $A$ can be reduced modulo $J$, because $A$ does not use divisions and $f_h(x)$ has degree $\leq 2$. If $S \in F[x_1, \ldots, x_n]$, then call $L_j(S)$ the homogeneous part of $S$ of degree $j$ and $S_i$ the expression computed at the step $i$. Let $S'_i = (L_0 + L_1 + L_2)S_i$ and $S''_i = L_1(S_i)$. The reduction modulo $J$ can be performed as follows:

1. If $S_i = \lambda \cdot S_j$, $j < i$, $\lambda \in F$, then we have $S'_i = \lambda \cdot S'_j$ and $S''_i = \lambda \cdot S''_j$.
2. If $S_i = S_j \pm S_k$, $j, k < i$, then we have $S'_i = S'_j \pm S'_k$ and $S''_i = S''_j \pm S''_k$.
3. If $S_i = S_j \cdot S_k$, $j, k < i$, then we have $S''_i = (c_j \cdot S''_k) + (c_k \cdot S''_j)$

and $S'_i = (c_j \cdot S'_k) + (c_k \cdot S'_j) + (S''_j \cdot S''_k) - c_j c_k$,

where $c_j = L_0(S_j)$ and $c_k = L_0(S_k)$. Thus the reduction modulo $J$ replaces any non scalar multiplication by a non scalar multiplication of the form $S''_j \cdot S''_k$, where both $S''_j$ and $S''_k$ are linear combinations of $x_1, \ldots, x_n$ plus other a/s and scalar multiplications. The total number of operations is bounded by $9(k_1 + k_2 + k_3)$.

To see how one can simulate divisions by multiplications we first recall the definition and the essential properties of formal power series [2, pp.41 sgg]. A formal power series in $x$ is an algebraic expression of the form $A(x) = \sum_{k=0}^{\infty} a_k x^k$, where $x$ is never assigned a numerical value and questions of convergence and divergence are not of interest. The set of these expressions
form a ring with sum and product defined by

\[ A(x) + B(x) = \sum_{k=0}^{\infty} a_k x^k + \sum_{k=0}^{\infty} b_k x^k = \sum_{k=0}^{\infty} (a_k + b_k) x^k \]

and

\[ A(x)B(x) = \sum_{k=0}^{\infty} c_k x^k, \quad c_k = \sum_{s=0}^{k} a_s b_{k-s}. \]

The zero element and the identity element are, respectively,

\[ 0 = \sum_{k=0}^{\infty} a_k x^k, \quad c_k = 0 \quad \forall k \geq 0, \]

and

\[ 1 = \sum_{k=0}^{\infty} a_k x^k, \quad a_0 = 1, \quad a_k = 0 \quad \forall k \geq 1. \]

If a formal power series \( A(x) = \sum_{k=0}^{\infty} a_k x^k \) has the constant coefficient \( a_0 \) different from zero, then there is a uniquely determined formal power series \( B(x) = \sum_{k=0}^{\infty} b_k x^k \) such that \( A(x)B(x) = 1 \). The coefficients of \( B(x) \) can be computed effectively, step by step, through the equations

\[ a_0 b_0 = 1, \quad a_0 b_1 + a_1 b_0 = 0, \quad a_0 b_2 + a_1 b_1 + a_2 b_0 = 0, \ldots. \]

For example, if \( A(x) = 1 + \sum_{k=1}^{\infty} a_k x^k \), (the geometric series), then its inverse is the formal polynomial \( B(x) = 1 - ax \). Analogous properties hold for formal power series of several variables \( x_1, \ldots, x_n \). The set of formal power series in \( x_1, \ldots, x_n \) with coefficients in \( F \) is denoted by \( F[[x_1, \ldots, x_n]] \) or \( F[[x]] \) and an invertible element of \( F[[x_1, \ldots, x_n]] \) is called a unit.

We have the following theorem, from which one deduces that no loss of generality follows by considering algorithms without divisions in computing a set of polynomials of degree at most 2. The same result can not be extended to polynomials of degree \( \geq 3 \), which implies that a possible \( F-n \)-linear program (defined in an obvious way in conformity with a bilinear program) would not be the most general model of computation of a set \( S \) of \( n \)-linear forms. It follows that a possible \( n \)-adic decompositions of an \( n \)-array \( t_{i_1i_2\ldots i_n} \) would not be a lower bound of multiplicative complexity for \( S \) for \( n > 2 \).

**Theorem 5.** Let \( S = \{f_h(x)\}, h = 1, 2, \ldots p, \) be a set of polynomials in the variables \( x_1, \ldots, x_n \), where each \( f_h(x) \) has degree \( \leq 2 \). If an algorithm \( \mathcal{A} \) computes \( S \) in \( F(x_1, \ldots, x_n) \) with \( q \) non-scalar m/d, then there is an algorithm \( \mathcal{A}' \) that computes \( S \) in \( F[x_1, \ldots, x_n] \) with \( q \) non-scalar multiplications.

**Proof.** To simulate the algorithm \( \mathcal{A} \) in \( F[x] \) we introduce the ring \( F[[x]] \) of the formal power series in the variables \( x_1, \ldots, x_n \). We first prove that
if $\mathcal{B}$ is an algorithm computing $\{f_h(x)\}$ in $F[[x]]$ with $k$ $m/d$, where each division has the form $A(x) \div B(x)$ and $B(x)$ is a unit of $F[[x]]$, then there exist an algorithm $\mathcal{B}'$ in $F[x_1, \ldots, x_n]$ computing $\{f_h(x)\}$ with $k$ non scalar multiplications. In fact we can define $\mathcal{B}'$ as follows:

1. replace the multiplications in $\mathcal{B}$ by multiplications modulo $J$, so to obtain at each step an expression of the form $c+l+q$, where $c, l, q$ have degree $0, 1, 2$, respectively. Then each multiplication can be reduced to a multiplication where both factors are linear combinations of $x_1, \ldots, x_n$ (by Theorem 4), plus a number of scalar multiplications and of $a/s$.

The first division in $\mathcal{B}$ has the form $A(x) \div B(x)$, where $A(x) = c_1 + l_1 + q_1$ and $B(x) = c_2 - (l_2 + q_2), c_2 \neq 0$. Then

$$A(x) \div B(x) = (c_1 + l_1 + q_1) \cdot (\frac{1}{c_2} + \frac{1}{c_2^2}(l_2 + q_2) + \frac{1}{c_2^3}(l_2 + q_2)^2 + \ldots).$$

Reducing mod $J$, that is dropping all terms of degree $\geq 3$, gives rise to the expression

$$\frac{c_1}{c_2} + \frac{c_1}{c_2^2}(l_2 + q_2) + \frac{l_1}{c_2} + \frac{q_1}{c_2} + \left(\frac{c_1}{c_2^2}l_2 + \frac{1}{c_2^3}l_1\right) \cdot l_2$$

where we have only one non scalar multiplication of the simplified form described in Theorem 4. The next division, and then each division in $\mathcal{B}$, can be replaced by a non scalar multiplication plus other scalar multiplications and $a/s$ by the same technique.

If $\mathcal{B}$ has some divisions where the divisor $B(x)$ is not a unit, that is $c_2 = 0$, then define the new variables $\tilde{x}_i = x_i - \theta_i, \theta_i \in F$, in order to obtain all divisions by units. Hence we are able to obtain from $\mathcal{A}$ an algorithm with $k$ non scalar multiplications of the simplified form $l(x) \cdot l'(x)$ in $F[\tilde{x}_1, \ldots, \tilde{x}_n]$, and then eventually an algorithm $\mathcal{A}'$ of the same complexity in $F[x_1, \ldots, x_n]$.

Now, by restricting the analysis to the non-scalar complexity, we have proved that it is sufficient to have all non-scalar operations of the form $l_1(a, b) \times l_2(a, b)$, where $l_1$ and $l_2$ denote linear combinations over $F$. Then, when the number of these special multiplications is $q$, we are able to write

$$f_h(a, b) = \sum_{r=1}^{q} w_{rh}l_{r1}(a, b) \cdot l_{r2}(a, b), \quad w_{rh} \in F$$

and to consider this formula, defining an $F$-bilinear program, as the most general strategy to compute the set $\{f_h(a, b)\}$. If the variables do not commute, then in the above sum the contribution of the cross-products $b \times a$ is identical to zero and the same conclusion holds for all product $a \times a$ and $b \times b$. In this case we can assume, in general, $l_{r1}(a, b) = u_r^T a = a^T u_r = \sum_{i=1}^{m} u_{ri} a_i$ and $l_{r2}(a, b) = v_r^T b = b^T v_r = \sum_{j=1}^{n} v_{rj} b_j$ where $u_{ri}, v_{rj} \in F$ and then
\[ f_h(a, b) \equiv a^T M_h b = \sum_{r=1}^{q} w_{rh}(a^T u_r) \cdot (v_r^T b) = a^T \left[ \sum_{r=1}^{q} w_{rh}(u_r v_r^T) \right] b. \quad (4) \]

From the last formula we deduce that the complexity, defined as the number \( q \) of non-scalar multiplications, depends on the fact that one can writes each \( M_h \) as a linear combination on \( F \) of \( q \) matrices \( u_r v_r^T \) of rank 1. This leads to search the minimum integer \( q \) such that each \( M_h \) can be expressed as a linear combination of \( q \) matrices of rank = 1 with elements in \( F \). But it easy to verify, by considering the Definition 4 and by rewriting the formula \( M_h = \sum_{r=1}^{q} w_{rh}(u_r v_r^T) \), that this minimum \( q \) is equal precisely to the rank of the tensor \( t_{ijh} \) defining the set of matrices \( M_h \), that is \( (M_h)_{ij} = t_{ijh} \). Then we have the following alternative definition of \( \text{rk}(t_{ijh}) \):

**Definition 4bis.** \( \text{rk}(t_{ijh}) \), the rank of the tensor \( t_{ijh} \) in a field \( F \), is the minimum integer \( q \) such that each matrix \( M_h \), \( (M_h)_{ij} = t_{ijh} \) can be expressed in the form \( \sum_{r=1}^{q} w_{rh}(u_r v_r^T) \), for \( 3q \) vectors \( u_r, v_r, w_r \) with elements in \( F \). The matrices of rank 1 \( u_r v_r^T \) form a tensor basis of the set \( \{ M_h \} \) and clearly \( q \geq d \), where \( d \) is the dimension of the space spanned by \( M_h \).

A third equivalent definition of the tensor rank is obtained by rewriting the formula \( M_h = \sum_{r=1}^{q} w_{rh}(u_r v_r^T) \) as a split of the matrix \( M_h \) as a product \( U D_h V \), where \( D_h \) is a diagonal matrix of dimensions \( q \times q \) whose diagonal elements are \( w_{rh} \), \( U \) is the \( m \times q \) matrix whose columns are \( u_r \) and \( V \) is the matrix \( q \times n \) whose rows are \( v_r^T \).

**Definition 4bisbis.** \( \text{rk}(t_{ijh}) \), the rank of the tensor \( t_{ijh} \) in a field \( F \), is the minimum integer \( q \) such that each matrix \( M_h \) can be written in the form \( U D_h V \), where \( D_h \) is diagonal \( q \times q \) and \( U, V \) are matrices of dimensions \( m \times q \) and \( q \times n \), respectively, with elements in \( F \).

Now the relationship between tensor rank and multiplicative complexity is stated in the following

**Theorem 6.** If \( \text{rk}(t_{ijh}) = q \), then the bilinear forms \( f_h(a, b) = a^T M_h b, h = 1, \ldots, p \), defined by the matrices \( M_h \) whose elements \( (M_h)_{ij} = t_{ijh} \) can be computed with \( q \) non-scalar multiplications. Moreover, in the general, non commutative case, \( q \) non-scalar multiplications are necessary to compute \( \{ f_h(a, b) \} \).

**Proof.** By the previous formula (4) \( q \) non-scalar multiplications are sufficient. To prove that \( q \) non-scalar multiplications are necessary consider a bilinear program of complexity \( s \) that computes \( \{ f_h(a, b) \} \). Then we can write

\[ f_h(a, b) = \sum_{r=1}^{s} w_{rh}(a^T u_r) \cdot (v_r^T b) = a^T \left[ \sum_{r=1}^{s} w_{rh}(u_r v_r^T) \right] b, \]

which implies \( M_h = \sum_{r=1}^{s} w_{rh}(u_r v_r^T) \). As \( q = \text{rk}(t_{ijh}) \) is the minimum
number of rank 1 matrices spanning the space of matrices generated by the set \( \{ M_h \} \), we have, by Definition 4bis, \( s \geq q \).

**Exercise 5.** Prove that if one can use commutativity in the products of variables, then \( \operatorname{rk}(t_{ijh}) = q \) implies that \( q/2 \) non-scalar multiplications are necessary to compute \( f_h(a, b) \), \( h = 1, \ldots, p \). Moreover, if \( d \) is the dimension of the space spanned by \( M_h \), then \( d \) non-scalar multiplications are necessary in both commutative and non commutative cases.

An example of a bilinear program exploiting commutativity is given by the inner vector multiplication defining the preprocessing in the matrix by vector product in the previous section. Notice [22, p.10], that the dyadic decompositions of tensors (Definition 4) are not unique but they have a link (as it is shown by Definition 4bis) with singular value decomposition (SVD) and with eigenvector decomposition (ED). The uniqueness of SVD and of ED can be ascribed to added constraints. Suppose that \( A = UDV \) is the SVD of \( A \), so \( U \) and \( V \) each have orthonormal columns and \( D \) is diagonal with positive entries on the main diagonal. Then \( A \) is a linear combination of the outer products of the columns of \( U \) with the corresponding rows of \( V \), using coefficients from \( D \). The uniqueness property of this dyadic decomposition is a consequence of the well known uniqueness property of SVD when the diagonal entries of \( D \) are unequal. The orthogonality of the columns of \( U \) and of the rows of \( V \) are added constraints that suffice to provide uniqueness. If \( \{ M_h \} \) is a set of symmetric and commutative matrices, then the unitary matrix \( U \) defining on a field \( F \) their common ED \( UDV \) defines also the tensor basis of the space spanned by \( \{ M_h \} \), and the tensor rank is \( n \) on the field \( F \).

We state once for all, for the indices \( i, j, h \), the following ranges:

\[
1 \leq i \leq m, \quad 1 \leq j \leq n, \quad 1 \leq h \leq p.
\]

For any 3-adic tensor \( (t_{ijh}) \) it is convenient to distinguish their different sections:

- the \( m \times n \) matrices \( \{ M_h \} \) associated to \( f_h(a, b), h = 1, \ldots, p \), with \( (M_h)_{ij} = t_{ijh} \) are called 3-sections;
- the \( n \times p \) matrices \( B_i \), with \( (B_i)_{jh} = t_{ijh} \) are the 1-sections;
- the \( m \times p \) matrices \( C_j \), with \( (C_j)_{ih} = t_{ijh} \) are the 2-sections.

In the following we will suppose, for the sake of simplicity, that all sections are composed of linear independent matrices. A tensor \( t_{ijh} \) can be represented through their 3-sections \( M_h, h = 1, 2, \ldots, p \), as well as through the matrix \( M = \sum_{h=1}^{p} c_h M_h \), where the \( c_h \) are variables. A third way to represent the same tensor consists of assigning the trilinear form \( g(a, b, c) := \sum_{h=1}^{p} f_h(a, b)c_h = \sum_{ijh} t_{ijh}a_ib_jc_h \).
Let the set \( \{W, U, V\} \) define a 3-adic decomposition of \( t_{ijh} \), i.e. 
\[
t_{ijh} = \sum_{r=1}^{q} u_{ri} v_{rj} w_{rh},
\]
and then 
\[
M_{h} = \sum_{r=1}^{q} w_{rh} (u_{r} v_{r}^{T}).
\]
Consider the (new) tensor defined by a cyclic permutation of the indices \( i, j, h \) of \( t_{ijh} \), that is the tensor \( t' \) whose elements \( t'_{hij} \) are equal to \( t_{ijh} \). The tensor \( t' \) defines a (new) set of bilinear forms associated to the \( p \times m \) matrices \( P_{j} \) (their 3-sections), with \( (P_{j})_{hi} = t_{hij} \). We have 
\[
M_{h} = \sum_{r=1}^{q} w_{rh} (u_{r} v_{r}^{T}),
\]
that is the two tensors \( t \) and \( t' \) have the same rank and if \( (u_{r} v_{r}^{T}) \) is a tensor basis of \( t_{ijh} \), then \( (w_{r} u_{r}^{T}) \) is a tensor basis of \( t'_{hij} \).

In general, if \( \{U_{1}, U_{2}, U_{3}\} = \{W, U, V\} \) defines a 3-adic decomposition of \( t_{ijh} \) and if \( \sigma \) is any permutation of the indices of \( t_{ijh} \), then \( \{U_{\sigma(1)}, U_{\sigma(2)}, U_{\sigma(3)}\} \) defines a 3-adic decomposition of \( t'_{\sigma(i)\sigma(j)\sigma(h)} \). So two tensors \( t \) and \( t' \) such that one is obtained from the other by a permutation of the indices have the same rank and are associated to sets of bilinear forms that have the same multiplicative complexity. These sets define dual problems.

**Example 2.** Consider the matrix by vector product
\[
\begin{bmatrix}
a_1 & a_2 \\
a_2 & a_3
\end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \end{bmatrix},
\]
giving rise to the bilinear forms 
\[
f_1 = a_1 b_1 + a_2 b_2, \quad f_2 = a_2 b_1 + a_3 b_2
\]
defined by the tensor \( t_{ijh}, 1 \leq i \leq 3, 1 \leq j \leq 2, 1 \leq h \leq 2 \), whose 3-sections are
\[
\{M_{h}\} = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \right\}.
\]
We have \( t_{111} = t_{221} = t_{212} = t_{322} = 1 \) and \( t_{ijh} = 0 \) otherwise. Now consider the tensor \( t' \) whose elements \( t'_{jhi} \) are equal to \( t_{ijh} \), that is 
\[
\begin{align*}
t'_{111} &= t_{111}, \\
t'_{122} &= t_{122} = t_{212} = t_{221} = t'_{223} = t_{322} = 1,
\end{align*}
\]
and \( t'_{jhi} = 0 \) otherwise. The 3-sections of the tensor \( t' \) are the matrices \( B_{i} \) such that \( (B_{i})_{jhi} = t_{ijh} \), that is 
\[
\{B_{i}\} = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \right\}.
\]
Notice that the space of matrices \( \begin{bmatrix} a_1 & a_2 \\ a_2 & a_3 \end{bmatrix} \) are spanned by the matrices 
\( B_{i}^{T} = B_{i} \).

Now we can explain more precisely the formal relationship between matrix by vector product and bilinear forms simply as follows: if \( t_{ijh} \) is the
tensor associated to the set \( \{ f_h(a, b) \} \), then the matrices transposed of its 1-sections are \( B_i^T \) and the matrix by vector product

\[
(\sum_{i=1}^m a_i B_i^T) b
\]
gives rise to a vector whose element \( h \) is equal precisely to \( f_h(a, b) \).

In fact, set \( c_h := \left[ (\sum_{i=1}^m a_i B_i^T) b \right]_h \). We have:

\[
c_h = \sum_{i=1}^m a_i (B_i^T b)_h = \sum_{i=1}^m a_i \sum_{j=1}^n (B_i^T)_{hj} b_j = \sum_{i=1}^m \sum_{j=1}^n (B_i)_{jih} a_i b_j
\]
and then

\[
c_h = \sum_{i=1}^m \sum_{j=1}^n t_{ijh} a_i b_j = f_h(a, b).
\]

Notice that the matrix

\[
B = \sum_{i=1}^m a_i B_i
\]
represents the tensor obtained from \( t_{ijh} \) by the permutation (231). So the rank of the tensor \( t_{ijh} \) defines the exact multiplicative complexity of the matrix by vector product \( (\sum_{i=1}^m a_i B_i^T) b \).

We can interpret the rank of \( t_{ijh} \) as the rank of the space spanned by \( \{ B_i \} \). By the Definition 4bis and by the dual property, the multiplicative complexity of \( f_h(a, b) \) depends on the minimum number \( q \) such that we can write each \( \{ B_i^T \} \) as a linear combination of \( q \) matrices of rank 1. If \( \text{rank}(\{ B_i^T \}) = q \), that is \( B_i^T = \sum_{r=1}^q u_{ri}(w_r v_r^T) \), then the algorithm for the matrix by vector product \( (\sum_{i=1}^m a_i B_i^T)b \) with the minimum number of non scalar multiplications is given by

\[
\sum_{i=1}^m a_i \sum_{r=1}^q u_{ri}(w_r v_r^T) b = \sum_{r=1}^q w_r \left( \sum_i u_{ri} a_i \right) \cdot \left( \sum_j v_{rj} b_j \right).
\]

That is

\[
\left[ (\sum_{i=1}^m a_i B_i^T) b \right]_h = \sum_{r=1}^q w_{rh} \left( \sum_i u_{ri} a_i \right) \cdot \left( \sum_j v_{rj} b_j \right),
\]
which is precisely a bilinear program of complexity \( q \). As a consequence, the complexity of each step \( k \) of an iterative algorithm based on a matrix by vector product \( B x_k \), for different vectors \( x_k \), depends on the representation of \( B \) as a linear combination of the minimum number \( q \) of rank 1 matrices. This number \( q \) is precisely the tensor rank of the space spanned by the matrices \( B_i \).
3. Tensor rank of Toeplitz (and Hankel) Matrices. Circulant matrices and the space $\tau$.

An $n \times n$ Toeplitz matrix $T_n$ has all elements equal on the main diagonal and on the diagonal parallel to the main diagonal. An $n \times n$ Hankel matrix $H_n$ has all elements equal on the antidiagonal and on the diagonal parallel to the antidiagonal. A Toeplitz matrix is obtained from a Hankel matrix by a permutation of columns. For $n = 5$ we have

$$T_5 = \begin{bmatrix} t_0 & t_5 & t_6 & t_7 & t_8 \\ t_1 & t_0 & t_5 & t_6 & t_7 \\ t_2 & t_1 & t_0 & t_5 & t_6 \\ t_3 & t_2 & t_1 & t_0 & t_5 \\ t_4 & t_3 & t_2 & t_1 & t_0 \end{bmatrix}, \quad H_5 = \begin{bmatrix} h_0 & h_1 & h_2 & h_3 & h_4 \\ h_1 & h_2 & h_3 & h_4 & h_5 \\ h_2 & h_3 & h_4 & h_5 & h_6 \\ h_3 & h_4 & h_5 & h_6 & h_7 \\ h_4 & h_5 & h_6 & h_7 & h_8 \end{bmatrix}.$$  

A Toeplitz matrix has elements $T_{ps} = t_{p-s}$, $t_{-k} = t_{n+k-1}$.

**Theorem 8.** Both spaces $T_n$ and $H_n$ have tensor rank $2n - 1$ on the rational field.

**Proof.** As a Toeplitz matrix is obtained from a Hankel matrix by a permutation of columns, it is sufficient to prove the theorem for $H_n$. To this aim consider the Hankel rank-1 matrix ($\lambda =$ parameter)

$$H(\lambda) = \begin{bmatrix} 1 & \lambda & \lambda^2 & \ldots & \lambda^{n-1} \\ \lambda & \lambda^2 & \lambda^3 & \ldots & \lambda^n \\ \lambda^2 & \lambda^3 & \lambda^4 & \ldots & \lambda^{n+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \lambda^{n-1} & \lambda^n & \lambda^{n+1} & \ldots & \lambda^{2n-2} \end{bmatrix}.$$  

Call $H_{2n-1}$ the matrix whose entries are all zero, but the entry at position $(n,n)$ that is equal to 1. Let $\lambda_1, \lambda_2, \ldots, \lambda_{2n-1}$ be $2n - 1$ distinct non null values of $\lambda$ in $Q$. Now the matrices $H(\lambda_i), i = 1, \ldots, 2n - 1$, are linear independent. In fact, consider a linear combination, in $Q$, of the matrices $H(\lambda)$:

$$\alpha_1 H(\lambda_1) + \alpha_2 H(\lambda_2) + \ldots + \alpha_{2n-1} H(\lambda_{2n-1}) = 0, \quad \alpha_i \in Q.$$  

Writing the above equality in explicit form gives an homogeneous system of $2n - 1$ linear equations in the $2n - 1$ unknowns $\alpha$, whose coefficients form a non singular Vandermonde matrix. Then all $\alpha$ are zero. An analogous result is obtained by considering the matrices $H(0), H_{2n-1}$ and other $2n - 3$ matrices $H(\lambda_i)$ for $2n - 3$ distinct rational values $\lambda_i$. The thesis follows from the fact that the rank of the space $H_n$ is $\geq$ of its dimension, which is equal to $2n - 1$.

The formal product of two polynomials of degree $n-1$, $p(x) = \sum_{k=0}^{n-1} a_k x^k$ and $q(x) = \sum_{k=0}^{n-1} b_k x^k$ is a polynomial $r(x) = \sum_{h=0}^{2n-2} c_h x^h$ whose coefficients
$c_h$ are defined by a set of $n$ bilinear forms $f_h(a, b) = a^TM_hb$, $h = 1, 2, \ldots, n$, where the $M_h$ are $n \times n$ Hankel matrices. Then $2n-1$ non-scalar multiplications are sufficient and necessary to compute $p(x)q(x)$.

**Exercise 6.** For $n = 3$, multiply two polynomials of degree $n$ by means of the optimal decomposition of the associated tensor. Consider the proof of the previous Theorem 8, choosing the tensor basis $H(0), H(1), H(-1), H(2), H_{2n-1}$ and prove that the only significant scalar operation is a division by $3$.

A special class of Toeplitz matrices are the well known algebra of $n \times n$ circulant matrices, a space generated by the first $n$ powers of the circulant permutation matrix $P$ whose first row is $[0 \ 1 \ 0 \ \ldots \ 0]$. A $5 \times 5$ circulant matrix and the correspondent $P$ have the form

$$C_5 = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 \\ a_4 & a_0 & a_1 & a_2 & a_3 \\ a_3 & a_4 & a_0 & a_1 & a_2 \\ a_2 & a_3 & a_4 & a_0 & a_1 \\ a_1 & a_2 & a_3 & a_4 & a_0 \end{bmatrix}, \quad P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$  

It is easy to prove the following equalities for a $n \times n$ circulant matrix $C_n$ (see also [5]):

$$C_n = \sum_{h=0}^{n-1} a_h P^h, \quad P^i P^j = \sum_{h=0}^{n-1} t_{ijh} P^h, \quad t_{ijh} = (P^i)_{jh} = (P^j)_{ih}.$$  

Moreover, $C_n = F_n^*DF_n$, with $(F_n)_{ij} = \omega^{(i-1)(j-1)}$, $\omega = e^{i2\pi/n}$. In other words all $n \times n$ circulant matrices can be reduced to diagonal form by the Fourier matrix $F_n$. Then the multiplicative table $t_{ijh}$ has the same structure of the matrices $P^h$, so the tensor $t_{ijh} = (P^i)_{jh}$, whose rk($t_{ijh}$) is equal to $n$ over the complex field, is the tensor associated to the discrete convolution on $n$ points, defined precisely, for $n = 5$, as the product of $C_n$ by a vector $b = [b_0 \ b_4 \ b_3 \ b_2 \ b_1]^T$. In fact, the convolution product of two vectors $a = [a_0 \ a_1 \ \ldots \ a_{n-1}]^T$ and $b = [b_0 \ b_1 \ \ldots \ b_{n-1}]^T$ is a vector $c := a \ast b = [c_0 \ c_1 \ \ldots \ c_{n-1}]^T$ whose component $c_h$ is given by

$$c_h = \sum_{i+j \mod h} a_i b_j.$$  

The formal product of the polynomials $p(x)$ and $q(x)$ of degree $n-1$ is defined by the convolution of the two $(2n-1)$-vectors $a = [a_0 \ a_1 \ \ldots \ a_{n-1} \ 0 \ \ldots \ 0]^T$ and $b = [b_0 \ b_1 \ \ldots \ b_{n-1} \ \ldots \ 0]^T$.

To compute $c = a \ast b$ the following equality, known as the Convolution Theorem, is usually exploited:

$$c = a \ast b = F_n^{-1}[(F_n a) \cdot (F_n b)]$$
where \( \cdot \) denotes the element by element vector multiplication. The Convolution Theorem is a consequence of the eigenvector decomposition of circulant matrices (verify that!), and the previous equality defines a bilinear program on \( C \) to compute \( a \star b \). The total asymptotic complexity of this program depends on the complexity of the FFT, which is \( O(n \log_2 n) \).

The following space, called space \( \tau \), is an algebra of \( n \times n \) symmetric and persymmetric matrices reduced to diagonal form by a unitary matrix with real elements. In [8] and in [32] some interesting links of \( \tau \) with symmetric Toeplitz matrices have been discovered. We write here below a matrix of the algebra \( \tau \) for \( n = 5 \):

\[
\tau_5 = \begin{bmatrix}
t_1 & t_2 & t_3 & t_4 & t_5 \\
t_2 & t_1 + t_3 & t_2 + t_4 & t_3 + t_5 & t_4 \\
t_3 & t_2 + t_4 & t_1 + t_3 + t_5 & t_2 + t_4 & t_3 \\
t_4 & t_3 + t_5 & t_2 + t_4 & t_1 + t_3 & t_2 \\
t_5 & t_4 & t_3 & t_2 & t_1
\end{bmatrix}.
\]

The space \( \tau \) is defined, in general, by a cross-sum property: \( t_{i-1,j} + t_{i+1,j} = t_{i,j-1} + t_{i,j+1} \), with suitable "boundary conditions". The eigenvectors are defined by \( u_{ij} = (2n+1)^{-1/2} \sin \frac{ij\pi}{n+1} \), so \( \tau = UDU^H \). The space \( \tau \) is generated in \( \mathbb{R} \) by the non-derogatory matrix

\[
H = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0
\end{bmatrix}.
\]

Both circulant and \( \tau \) matrices can be exploited to define fast algorithms for matrix by vector product when the matrix has Toeplitz form. In fact, a general \( n \times n \) Toeplitz matrix is a part of a circulant matrix of dimension \( 2n \), so the problem \( T_n \times \text{vector} \) can be reduced to a problem \( C_{2n} \times \text{vector} \), and then to a number of FFT on \( 2n \) points, with only \( O(n \log_2 n) \) operations. Consider here below the case \( n = 3 \):

\[
\begin{bmatrix}
t_0 & t_3 & t_4 & 0 & t_2 & t_1 \\
t_1 & t_0 & t_3 & t_4 & 0 & t_2 \\
t_2 & t_1 & t_0 & t_3 & t_4 & 0 \\
0 & t_2 & t_1 & t_0 & t_3 & t_4 \\
t_4 & 0 & t_2 & t_1 & t_0 & t_3 \\
t_5 & t_4 & 0 & t_2 & t_1 & t_0
\end{bmatrix} \times \begin{bmatrix}
s_0 \\
s_1 \\
s_2 \\
0 \\
0 \\
0
\end{bmatrix}.
\]

Notice that the previous decomposition of the tensor of Toeplitz or Hankel matrices, stated in Theorem 8, is not optimal, on the real field, for symmetric Toeplitz matrices. In fact, consider the following split of a general
symmetric Toeplitz matrix (for \( n = 5 \)):

\[
T_5 = \begin{bmatrix}
t_1 & t_2 & t_3 & t_4 & t_5 \\
t_2 & t_1 & t_2 & t_3 & t_4 \\
t_3 & t_2 & t_1 & t_2 & t_3 \\
t_4 & t_3 & t_2 & t_1 & t_2 \\
t_5 & t_4 & t_3 & t_2 & t_1 \\
\end{bmatrix}
= \begin{bmatrix}
t_1 & t_2 & t_3 & t_4 & t_5 \\
t_2 & t_1 + t_3 & t_2 & t_4 & t_3 + t_5 \\
t_3 & t_2 & t_4 & t_1 + t_3 + t_5 & t_2 + t_4 \\
t_4 & t_3 & t_1 + t_3 & t_2 + t_4 & t_1 \\
t_5 & t_4 & t_3 & t_2 & t_1 \\
\end{bmatrix}
\]

\[
- \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & t_3 & t_4 & t_5 & 0 \\
0 & t_4 & t_3 + t_5 & t_4 & 0 \\
0 & t_5 & t_4 & t_3 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{bmatrix} \Rightarrow T_5 = \tau_5 - \begin{bmatrix}
0 & 0^T & 0 \\
0 & \tau_3 & 0 \\
0 & 0^T & 0 \\
\end{bmatrix}.
\]

We see that, in general, \(2n - 2\) non-scalar multiplications are sufficient to compute a matrix by vector product \(T_n \times b\) on the real field when \(T_n\) is a \(n \times n\) Toeplitz symmetric matrix. Moreover \(2n - 2\) non-scalar multiplications are necessary, as \(rk(T_n) = 2n - 2\) in \(\mathbb{R}\) [32].

**Research problem.** Find a tensor basis of the space of symmetric Toeplitz matrices on the rational field.


When \(T\) is an upper (lower) triangular non singular Toeplitz matrix \(T^{-1}\) is also an upper (lower) triangular Toeplitz matrix, because Toeplitz upper or lower triangular matrices form an algebra (closed under multiplication and inversion). But if \(T\) is a non singular Toeplitz matrix, \(T^{-1}\) has not, in general, a Toeplitz structure. Yet we should investigate the very nature of the structure of the matrix \(T\) or \(T^{-1}\) apart from its Toeplitzness, which is, in \(T\), the most obvious and visible structural property. Now consider the case of a general \(4 \times 4\) symmetric Toeplitz matrix, together with the matrix \(Z\) defined here below:

\[
T_4 = \begin{bmatrix}
t_0 & t_1 & t_2 & t_3 \\
t_1 & t_0 & t_1 & t_2 \\
t_2 & t_1 & t_0 & t_1 \\
t_3 & t_2 & t_1 & t_0 \\
\end{bmatrix}, \quad Z = \begin{bmatrix}
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
\end{bmatrix}.
\]

The effect of \(Z\) consists of shifting downwards the elements of a vector by one position and replacing the first element with a zero entry:

\[
Z \begin{bmatrix}
x_0 \\
x_1 \\
x_2 \\
x_3 \\
\end{bmatrix} = \begin{bmatrix}
0 \\
x_0 \\
x_1 \\
x_2 \\
\end{bmatrix}.
\]

22
Now let $T$ a Toeplitz matrix and define the operator $\nabla T$ as follows:

$$
\nabla T := T - ZTZ^T = T - \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & t_0 & t_1 & t_2 \\
0 & t_1 & t_0 & t_1 \\
0 & t_2 & t_1 & t_0
\end{bmatrix} = \begin{bmatrix}
t_0 & t_1 & t_2 & t_3 \\
t_1 & 0 & 0 & 0 \\
t_2 & 0 & 0 & 0 \\
t_3 & 0 & 0 & 0
\end{bmatrix}
$$

We have (for $t_0 = 1$)

$$
\nabla T = GJG^T, \quad G = \begin{bmatrix}
1 & 0 \\
t_1 & t_1 \\
t_2 & t_2 \\
t_3 & t_3
\end{bmatrix}, \quad J = \begin{bmatrix}
1 & 0 \\
0 & -1
\end{bmatrix}, \quad \text{rk}(GJG^T) = 2.
$$

Notice that $\nabla T$ is symmetric when $T$ is symmetric, so the eigenvalues of $\nabla T$ are real. Moreover the two diagonal elements of $J$ indicate that $\nabla T$ has one eigenvalue positive and the other negative.

The same operator $\nabla$ can be applied to a general Hermitian matrix. In fact let, for an Hermitian matrix $R$,

$$
\nabla R = R - ZRZ^H.
$$

The matrix $ZRZ^H$ corresponds to shifting $R$ downwards along the main diagonal by one position, which explains the name displacement for $\nabla R$. If $\nabla R$ has low rank, say $r << n$, then $R$ is said to be structured with respect to the displacement defined by the operator $\nabla R$, and $r$ is referred as the displacement rank of $R$ [20]. The matrix $\nabla R = R - ZRZ^H$ is Hermitian, its eigenvalues are real and we can define the displacement inertia of $R$ as the pair $(p,q)$, where $p$ and $q$ are the number of the positive and negative eigenvalues, respectively. The displacement rank $r$ is equal precisely to $p+q$ and then

$$
\nabla R = R - ZRZ^H = GJG^H, \quad J = (I_p \oplus -I_q)
$$

where $J$ and $G$ have dimensions $r \times r$ and $n \times r$, respectively. In [20] the pair $\{G, J\}$ is called a generator of $R$. In fact it contains all information regarding $R$, that is the information on the structure allowing to reduce the space for representing $R$ and the useful information on the complexity of a matrix by vector product $Ry$. It is easy to prove that the only solution of the above equation is

$$
R = \sum_{i=0}^{n-1} Z^i GJG^H(Z^H)^i.
$$

In fact, let

$$
R = GJG^H + ZGJG^HZ^H + Z^2GJG^H(Z^H)^2 + \ldots + Z^{n-1}GJG^H(Z^H)^{n-1}.
$$
Now multiply on the left by \( Z \) and on the right by \( Z^H \). As \( Z \) is nilpotent, i.e. \( Z^n = 0 \), we have

\[
Z R Z^H = Z G J G^H Z^H + Z^2 G J G^H (Z^H)^2 + \ldots + Z^{n-1} G J G^H (Z^H)^{n-1}.
\]

Then subtract term by term the last two equalities so to obtain, eventually,

\[
R - Z R Z^H = G J G^H.
\]

If

\[
G = [x_0 x_1 \ldots x_{p-1} y_0 y_1 \ldots y_{q-1}],
\]

then we can rewrite the equality \( R = \sum_{i=0}^{n-1} Z^i G J G^H (Z^H)^i \) in the form

\[
R = \sum_{i=0}^{p-1} L(x_i) L^H(x_i) - \sum_{i=0}^{q-1} L(y_i) L^H(y_i),
\]

where

\[
L(z) = \begin{bmatrix}
z_0 & 0 & \ldots & \ldots & 0 \\
z_1 & z_0 & 0 & \ldots & 0 \\
\vdots & \ldots & \ldots & \ldots & \ldots \\
z_{n-2} & \ldots & \ldots & 0 & 0 \\
z_{n-1} & \ldots & \ldots & z_1 & z_0
\end{bmatrix}.
\]

**Exercise 7.** Prove that (5) is identical to the equality \( R = \sum_{i=0}^{n-1} Z^i G J G^H (Z^H)^i \).

A general result concerning matrices with displacement structure states that the displacement inertia of a matrix \( R \) is in some way inherited by its inverse. More precisely, we have the following

**Theorem 9.** [20] The displacement inertia of a Hermitian non singular matrix \( R \) with respect to \( R - Z R Z^H \) is equal to the displacement inertia of its inverse with respect to \( R^{-1} - Z^H R^{-1} Z \), i.e. \( \text{Inertia}(R - Z R Z^H) = \text{Inertia}(R^{-1} - Z^H R^{-1} Z) \).

**Proof.** The theorem follows from the two identities, exploiting the Schur complements of \( R \),

\[
\begin{bmatrix}
R & Z \\
Z^H & R^{-1}
\end{bmatrix} = \begin{bmatrix}
I & 0 \\
Z^H R^{-1} & I
\end{bmatrix} \begin{bmatrix}
R & 0 \\
0 & R^{-1} - Z^H R^{-1} Z
\end{bmatrix} \begin{bmatrix}
I & 0 \\
Z^H R^{-1} & I
\end{bmatrix}^H
\]

and

\[
\begin{bmatrix}
R & Z \\
Z^H & R^{-1}
\end{bmatrix} = \begin{bmatrix}
I & Z R \\
0 & I
\end{bmatrix} \begin{bmatrix}
R - Z R Z^H & 0 \\
0 & R^{-1}
\end{bmatrix} \begin{bmatrix}
I & Z R \\
0 & I
\end{bmatrix}^H.
\]

The matrices \( R - Z R Z^H \) and \( R^{-1} - Z^H R^{-1} Z \) are called Schur complements.

As in general \( \text{Inertia}(A B A^H) = \text{Inertia}(B) \) (Sylvester’s theorem: congruence transformations preserve Inertia), the two matrices

\[
\begin{bmatrix}
R & 0 \\
0 & R^{-1} - Z^H R^{-1} Z
\end{bmatrix}, \quad \begin{bmatrix}
R - Z R Z^H & 0 \\
0 & R^{-1}
\end{bmatrix}
\]

\]
have the same Inertia. This implies the final identity $\text{Inertia}(R - ZRZ^H) = \text{Inertia}(R^{-1} - Z^H R^{-1} Z)$.

**Exercise 8.** Find (possibly in the matrix literature) a proof of the Sylvester’s theorem: congruence transformations preserve Inertia.

A consequence of the above theorem is concerned with the Toeplitz matrices: the inverse of a non singular symmetric Toeplitz matrix $T$ has the same displacement Inertia of $T$. In fact $\text{Inertia}(T - ZTZ^H) = (1, 1) = \text{Inertia}(T^{-1} - Z^HT^{-1} Z) = \text{Inertia}(T^{-1} - ZT^{-1} Z^H)$, because

$$\tilde{I}T\tilde{I} = T, \quad \tilde{I}T^{-1}\tilde{I} = T^{-1}, \quad \tilde{I}Z^H\tilde{I} = Z,$$

where $\tilde{I}$ is the reverse identity matrix with ones on the antidiagonal and zeros elsewhere. Then we have precisely, for a pair of vectors $\{x, y\}$,

$$T^{-1} - ZT^{-1}Z^H = xx^H - yy^H = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x^H \\ y^H \end{bmatrix},$$

and

$$T^{-1} = L(x)L(x)^H - L(y)L(y)^H,$$

which is the celebrated formula of Gohberg-Semencul. By this formula the matrix-vector product $T^{-1}z$ can be computed, as the product $Tz$, through a small number of FFT, and then in only $O(n \log_2 n)$ arithmetic operations. Thus iterative solvers for Toeplitz systems may be quite competitive because of a fast matrix by vector procedure with a small number of fast transforms. Notice that the result of Theorem 9 does not depend on the special form of the matrix $Z$ and so the matrix $Z$ could be replaced by a general matrix $\Omega$: the displacement inertia of an Hermitian non singular matrix $R$ with respect to $R - \Omega R\Omega^H$ is equal to the displacement inertia of its inverse with respect to $R^{-1} - \Omega^H R^{-1} \Omega$.

**References**


