METHODS FOR THE CALCULATION OF VALUES OF A NORM FORM

A. BÉRCZES AND J. KÖDMÖN

Abstract. In this paper we analyse an important property of norm form functions, i. e. how hard is the computation of the function value if the arguments are given. We will present three algorithms for the calculation of the value of a norm form. Their complexity will be discussed and the running time of their implementations in MAPLE will be compared.

1. Introduction

The calculation of the values of a multivariate polynomial usually is a difficult task. Let $P(\underline{X})$ be a k-variable polynomial of total degree n. We can write

$$P(\underline{X}) = \sum_{j=1}^{m} a_j \underline{X}^{\underline{e}_j},$$

where $\underline{e} = (e_1, ..., e_k) \in \mathbb{Z}^k$, $0 \leq e_1 + ... + e_k \leq n$ and $\underline{X}^{\underline{e}} := X_1^{e_1} X_2^{e_2} \cdots X_k^{e_k}$.

Let $P(\underline{x})$ denote the value of the polynomial $P(\underline{X})$ at the point $\underline{x} = (x_1, ..., x_k)$. The number of the terms of P is at most $\binom{n+1+k-1}{k} = \binom{n+k}{k} \approx \frac{2^{n+k}-2}{n+k-1} < 2^{n+k}$. For the naive calculation of one term $O(n^2 \log^2 \mathbb{X})$ bit operations are needed where $\mathbb{X} = \max\{|x_i|\}$ and the constant in O depends only on $\max\{|a_j|\}$. Thus, the complexity of the determination of the substitution value is $O(2^{n+k} n^2 \log^2 \mathbb{X})$, which is exponential in $n + k$.

There are also more efficient multiplication methods. The best algorithm, due to Schönhage and Strassen (see [11]), runs in $O(n \log n \log \log n)$ bit operations. However, this algorithm is worth using only for numbers having

1991 *Mathematics Subject Classification.* 11Y40, 11Y16.

Key words and phrases. norm forms, complexity, algorithms.

The authors would like to thank professor A. Pethő for the help.

more than thousand digits. For smaller numbers can be useful the method of Karatsuba-Ofman (see [7]).

If the vector $\underline{x} = (x_1, ..., x_k)$ has rational integer components we can use an intelligent powering method such as the Right-Left Binary algorithm (see Algorithm 1.2.1 in [3]) which has a cost of $O(\log n \log^2 \mathbb{X})$ binary operations. In this best case the complexity of the substitution is $O(2^{n+k+1} \log n \log^2 \mathbb{X})$.

So, determining the value $P(\underline{x})$ is in general a really difficult task.

There are, however, special polynomials where the calculation of the value is much easier. Let us take the following example

$$P(\underline{X}) = \begin{vmatrix} L_{1,1}(\underline{X}) & \cdots & L_{1,n}(\underline{X}) \\ \cdots & & \cdots \\ L_{n,1}(\underline{X}) & \cdots & L_{n,n}(\underline{X}) \end{vmatrix},$$

where $L_{i,j}(\underline{X}) = a_{ij1}X_1 + ... + a_{ijk}X_k$, $a_{ijk} \in \mathbb{Z}$; $1 \leq i, j \leq n$, $1 \leq k \leq n$ are linear forms. After expanding the determinant the result is a k-variable homogeneous polynomial of degree n. However, the value of this can be calculated in polynomial time in n since after determining the value of the linear forms $L_{i,j}(\underline{X})$ the calculation of the determinant has only polynomial complexity in n.

When the norm of a linear form with algebraic integer coefficients is to be computed we also can do it by computing a determinant similar to the one above. The polynomial given this way is actually a norm form. Norm forms are playing an important role in the theory of diophantine equations (see [1], [5] and [9]).

2. Norm Forms

Let θ be an algebraic integer of degree n, and denote by $T(X) \in \mathbb{Z}[X]$ its minimal polynomial over $\mathbb{Q}$. Put $K := \mathbb{Q}(\theta)$, and denote by $\sigma_i : K \to \mathbb{C}$ the distinct embedings of K into the field of complex numbers. Further, for any $\alpha \in K$ denote by $\alpha_i := \sigma_i(\alpha)$ $(i = 1, \ldots, n)$ the field conjugates of α.

Let $\alpha_1, \alpha_2, ..., \alpha_u \in K$ be $\mathbb{Q}$-linearly independent algebraic integers. Consider the linear form

$$L(\underline{X}) = \alpha_1 X_1 + ... + \alpha_u X_u, \tag{1}$$

where $u \leq n$ and put

$$L^{(i)}(\underline{X}) = \alpha_1^{(i)} X_1 + ... + \alpha_u^{(i)} X_u. \tag{2}$$

The polynomial

$$Norm_{\mathbb{K}/\mathbb{Q}}(L(\underline{X})) = \prod_{i=1}^{n} L^{(i)}(\underline{X})$$

is called a norm form. It is easily seen that $Norm_{\mathbb{K}/\mathbb{Q}}(L(\underline{X}))$ is a homogeneous polynomial of degree n, with integer coefficients.

Consider now the special case $\alpha_1 = 1, \alpha_2 = \theta, ..., \alpha_u = \theta^{u-1}$. Since each norm form can be transformed by a linear transformation with rational coefficients to a norm form of the above special type, i.e. to a norm form

$$\mathcal{N}(\underline{X}) = Norm_{\mathbb{K}/\mathbb{Q}}(X_1 + \theta X_2 + \cdots + \theta^{(u-1)} X_u). \tag{3}$$

In the rest of the paper we will restrict our investigations to the case of norm forms of the shape (3). Further, due to some technical issues, in sections 4 and 5 we also suppose that $1, \theta, ..., \theta^{n-1}$ is a power integral basis for $\mathbb{Z}_K$, where $\mathbb{Z}_K$ denotes the ring of integers of K.

In the following sections we consider and compare three different methods to compute values of $\mathcal{N}(\underline{X})$ at points $\underline{x} \in \mathbb{Z}^u$.

3. Calculation of $\mathcal{N}(\underline{x})$ by the Definition

In the first method we use a floating point approximation $\widetilde{\alpha}_j^{(i)}$ of each algebraic integer $\alpha_j^{(i)}$. Knowing that $\mathcal{N}(\underline{x})$ is an integer for $\underline{x} \in \mathbb{Z}^u$ it is enough to approximate $\alpha_j^{(i)}$ by $\widetilde{\alpha}_j^{(i)}$ with such a precision that $\left|\mathcal{N}(\underline{x}) - \widetilde{\mathcal{N}}(\underline{x})\right| < \frac{1}{2}$, where $\widetilde{\mathcal{N}}(\underline{x}) := \prod_{i=1}^{n} \left(\sum_{j=1}^{u} \widetilde{\alpha}_j^{(i)} x_j\right)$. Let us estimate the upper bound on the error of the calculation.

Lemma 1. *Using the above notation put* $\overline{|\alpha_j|} = \max\left\{\left|\alpha_j^{(i)}\right| : 1 \le i \le n\right\}$, $A = \sum_{j=1}^{u} \overline{|\alpha_j|} \ge 2$, $\Delta = \max\left\{\left|\alpha_j^{(i)} - \widetilde{\alpha}_j{}^{(i)}\right| : 1 \le i \le n,\ 1 \le j \le u\right\}$ *and* $\mathbb{X} = \max\{|x_1|, \ldots, |x_u|\}$. *Then the upper bound on the error is:*

$$\Delta_{\widetilde{\mathcal{N}}(\underline{x})} = \left|\mathcal{N}(\underline{x}) - \widetilde{\mathcal{N}}(\underline{x})\right| \le nu\Delta(A\mathbb{X})^n$$

provided that $\Delta \le \frac{A}{nu\mathbb{X}}$.

Proof. Let $\beta^{(i)} = \sum_{j=1}^{u} \alpha_j^{(i)} x_j$, $\widetilde{\beta}^{(i)} = \sum_{j=1}^{u} \widetilde{\alpha}_j^{(i)} x_j$ and $\widetilde{\beta}^{(0)} = 1$. Then the estimation will have the form:

$$\begin{aligned}
\Delta_{\widetilde{\mathcal{N}}(\underline{x})} &= \left|\mathcal{N}(\underline{x}) - \widetilde{\mathcal{N}}(\underline{x})\right| = \left|\prod_{i=1}^{n} \beta^{(i)} - \prod_{i=1}^{n} \widetilde{\beta}^{(i)}\right| = \\
&= \left|\sum_{j=0}^{n-1} \left(\prod_{i=0}^{j} \widetilde{\beta}^{(i)} \prod_{i=j+1}^{n} \beta^{(i)} - \prod_{i=1}^{j+1} \widetilde{\beta}^{(i)} \prod_{i=j+2}^{n} \beta^{(i)}\right)\right| = \\
&= \left|\sum_{j=0}^{n-1} \left(\beta^{(j+1)} - \widetilde{\beta}^{(j+1)}\right) \prod_{i=0}^{j} \widetilde{\beta}^{(i)} \prod_{i=j+2}^{n} \beta^{(i)}\right|.
\end{aligned}$$

Now using that

$$\left|\beta^{(i)}\right| \leq \mathbb{X} \sum_{j=1}^{u} \left|\alpha_j\right| = A\mathbb{X},$$

$$\left|\widetilde{\beta}^{(i)}\right| \leq \mathbb{X} \sum_{j=1}^{u} \left(\left|\alpha_j\right| + \Delta\right) = \mathbb{X}\left(A + u\Delta\right)$$

$$\left|\beta^{(i)} - \widetilde{\beta}^{(i)}\right| = \left|\sum_{j=1}^{u} \left(\alpha_j^{(i)} - \widetilde{\alpha_j}^{(i)}\right) x_j\right| \leq \mathbb{X} u \Delta$$

and

$$\Delta \leq \frac{A}{nu\mathbb{X}}$$

we obtain that

$$\Delta_{\widetilde{\mathcal{N}}(\underline{x})} \leq u\Delta\mathbb{X}^n \sum_{j=0}^{n-1} \left(A + u\Delta\right)^j A^{n-j-1} = u\Delta\mathbb{X}^n A^{n-1} \sum_{j=0}^{n-1} \left(1 + \frac{u\Delta}{A}\right)^j =$$

$$= u\Delta\mathbb{X}^n A^{n-1} \frac{\left(1 + \frac{u\Delta}{A}\right)^n - 1}{\frac{u\Delta}{A}} = \left(A\mathbb{X}\right)^n \left(\left(1 + \frac{u\Delta}{A}\right)^n - 1\right) =$$

$$= \left(A\mathbb{X}\right)^n \sum_{i=1}^{n} \binom{n}{j} \left(\frac{u\Delta}{A}\right)^j \leq \left(A\mathbb{X}\right)^n \sum_{i=1}^{n} \frac{(nu\Delta)^j}{A^j j!} \leq$$

$$\leq \left(A\mathbb{X}\right)^n \frac{nu\Delta}{2A} \sum_{i=1}^{infty} \frac{(nu\Delta)^j}{A^j j!} \leq \left(A\mathbb{X}\right)^n \frac{nu\Delta}{2A} \exp\left(\frac{nu\Delta}{A}\right) <$$

$$\frac{enu\Delta}{2A} \left(A\mathbb{X}\right)^n \leq nu\Delta \left(A\mathbb{X}\right)^n.$$

This concludes the proof of Lemma 1. □

The definition of the norm function implies that $\mathcal{N}(\underline{x}) \in \mathbb{Z}$ for $\underline{x} \in \mathbb{Z}^u$, thus the upper bound on the error of the calculation has to satisfy the inequality $\Delta_{\widetilde{\mathcal{N}}(\underline{x})} < \frac{1}{2}$. From this follows that we must choose the precision of the calculation such that:

$$\Delta < \frac{1}{2nu\left(A\mathbb{X}\right)^n}.$$

After determining the necessary precision of the approximations $\widetilde{\alpha}_i^{(j)}$ to $\alpha_i^{(j)}$ we study the complexity of the calculation of $\mathcal{N}(\underline{x})$ using directly the definition of the norm form and floating point approximations to its coefficients. We have the following theorem:

Theorem 1. *The complexity of determination of the function value* $\mathcal{N}(\underline{x})$ *according to* (3) *is* $O\left(n^6 + n^4 \log^2 \mathbb{X}\right)$*, where the constant in* O *depends only on* A*.*

Proof. Let $m := n \log \mathbb{X} + \log(2nuA^n)$ and choose approximations $\widetilde{\alpha}_i^{(j)}$ to each $\alpha_i^{(j)}$ having the following properties

- both the real and the imaginary part of $\widetilde{\alpha}_i^{(j)}$ has a fractional part of at most m+1 digits in its binary representation
- $|\Re(\alpha_i^{(j)}) - \Re(\widetilde{\alpha}_i^{(j)})| < 2^{-m-1}$ and
- $|\Im(\alpha_i^{(j)}) - \Im(\widetilde{\alpha}_i^{(j)})| < 2^{-m-1}$.

Then we have $\Delta < 2^{-m}$. Further, it is easily seen that

$$m \leq n \log \mathbb{X} + C_1 nu.$$

For any complex number z denote by $l(z)$ the maximum of the binary length of the real and imaginary part of the number. Since $u \leq n$ we get

$$l\left(\widetilde{\alpha}_i^{(j)}\right) \leq \log \overline{|\alpha_i|} + n \log \mathbb{X} + C_1 n^2 = n \log \mathbb{X} + C_1 n^2 + C_2.$$

Now let us determine the complexity of the calculation of the values $L_j(\underline{x}) = \sum_{i=1}^{u} \widetilde{\alpha}_i^{(j)} x_i$.

First of all we estimate the number of binary operations needed to calculate the appropriate approximations to the algebraic integers $\alpha_i^{(j)}$. According to Theorem 19.2 of [10] this can be done in

$$O(n^3 \log n + n^4 \log \mathbb{X} + C_1 n^5 + C_2 n^3) = O(n^5 + n^4 \log \mathbb{X}) \tag{4}$$

bit operations.

Now we turn to estimate the complexity of the remaining part of the calculation. The number of the binary operations needed for one multiplication is at most

$$C_3(n \log^2 \mathbb{X} + C_1 n^2 \log \mathbb{X} + C_2 \log \mathbb{X}),$$

hence u $(\leq n)$ pieces of such multiplications demand less than

$$C_3 n^2 \log^2 \mathbb{X} + C_4 n^3 \log \mathbb{X} + C_5 n \log \mathbb{X}$$

binary operations.

Let $l := \max_j \{l(L_j)\} := \max_j \{l(L_j(\underline{x}))\} \leq \max_{i,j} \left\{ l\left(\widetilde{\alpha}_i^{(j)}\right) \right\} + \log \mathbb{X} + n \leq (n+1) \log \mathbb{X} + (C_1 + 1) n^2 + C_2$ be the maximal length of one $L_j(\underline{x})$.

Let us determine the complexity of the calculation of $\prod_{j=1}^{n} L_j(\underline{x})$. The number of the binary operations needed for it is obviously the sum of the elements of the sequence

$$l(L_1)l(L_2), [l(L_1)+l(L_2)]l(L_3), ..., [l(L_1)+...+l(L_{n-1})]l(L_n).$$

times a constant. This can be estimated from above by the sum of the elements of the sequence $l^2, 2l^2, ..., (n-1)l^2$ times a constant. So, the complexity of the calculation of the complete product is:

$$C_6\frac{n(n-1)}{2}l^2 = C_6\frac{n(n-1)}{2}\left((n+1)\log\mathbb{X}+C_1n^2+C_2\right)^2 \approx \approx C_7n^6+C_8n^4\log^2\mathbb{X}. \tag{5}$$

Further, the calculation of n pieces of L_j demands

$$C_3n^3\log^2\mathbb{X}+C_4n^4\log\mathbb{X}+C_2n^2\log\mathbb{X} \tag{6}$$

binary operations.

Now (4), (5) and (6) imply that the complexity of the whole calculation is:

$$O\left(n^6+n^4\log^2\mathbb{X}\right).$$

This concludes the proof of Theorem 1. □

4. Calculation of $\mathcal{N}(\underline{x})$ by Matrix Representation

If $\Omega=\{\omega_1,...,\omega_n\}$ is a $\mathbb{Q}$-basis of an algebraic number field K and if $\alpha\in K$, then multiplication by α is an endomorphism of the $\mathbb{Q}$-vector space K, and we can represent α by the matrix M_α of this endomorphism in the basis Ω. This matrix has in general rational entries. This representation is unique, and the map $\alpha\mapsto M_\alpha$ is a homomorphism from K to the algebra of $n\times n$ matrices over $\mathbb{Q}$. If $\Omega:=\{1,\theta,\theta^2,\dots,\theta^{n-1}\}$ is a power integral basis of K and $\alpha=a_1\omega_1+...+a_n\omega_n$ where $a_i\in\mathbb{Z}$ for $1\le i\le n$, then M_α has integral entries. It is well known that the norm of α is the determinant of the corresponding matrix M_α. Further details concerning matrix representation of algebraic numbers can be found in [3] and [1].

Theorem 2. *Let K be an algebraic number field with a power integral basis $1,\theta,\dots,\theta^{n-1}$, $\underline{x}\in\mathbb{Z}^u$ and define the norm form $\mathcal{N}(\underline{X})$ by (3). Then $\mathcal{N}(\underline{x})$ can be determined with integer arithmetic.*

In the proof the matrix representation of the linear form $L(\underline{X})$ will be applied.

Proof. In the constructive proof concrete algorithm is given for the determination of $\mathcal{N}(\underline{x})$.

As earlier, we consider the number field $K=\mathbb{Q}(\theta)$ and we also recall the following notation. Let $u\le n$ and put $\alpha_1=1,\alpha_2=\theta,...,\alpha_u=\theta^{u-1}$.

Consider the following linear form used earlier:

$$L(\underline{X}) = \alpha_1 X_1 + ... + \alpha_u X_u = \sum_{i=1}^{u} \theta^{i-1} X_i.$$

For each $\underline{x} \in \mathbb{Z}^u$ $L(\underline{x})$ will be an element of the field K, thus we can compute its matrix representation on the basis Ω. However, we can first compute a matrix representation $\Lambda(\underline{X})$ of $L(\underline{X})$ on the basis Ω, which will have linear forms in $\underline{X}$ as its entries, and which will have the property that $\Lambda(\underline{x})$ gives the matrix representation of the element $L(\underline{x})$ for each $\underline{x} \in \mathbb{Z}^u$. Now we compute this matrix representation of $L(\underline{X})$.

In order to determine the elements of this matrix consider the following products:

$$\theta^k L(\underline{X}) = \sum_{i=0}^{u-1} \theta^{i+k} X_i, \text{ where } 0 \leq k \leq n-1. \tag{7}$$

In (7) we shall substitute each θ^s with its standard representation in the base Ω. Put

$$\theta^s = \sum_{j=0}^{n-1} r_{s,j} \theta^j, \text{ where } 0 \leq s \leq 2n-2. \tag{8}$$

The standard representation of θ^s for $0 \leq s \leq n-1$ is trivial and we have $r_{s,j} = 1$ if $j = s$ and $r_{s,j} = 0$ otherwise. The standard representation of the powers θ^{n+k} $(0 \leq k \leq n-2)$ can be determined from the coefficients of the minimal polynomial $T(X)$ of θ using the Newton recursion formulae. The method will be described later in Lemma 2.

Then by (7) and (8) we have

$$\theta^k L(\underline{X}) = \sum_{i=0}^{u-1} \left(\sum_{j=0}^{n-1} r_{i+k,j} \theta^j \right) X_i.$$

Changing the order of summation we get

$$\theta^k L(\underline{X}) = \sum_{j=0}^{n-1} \left(\sum_{i=0}^{u-1} r_{i+k,j} X_i \right) \theta^j = \sum_{j=0}^{n-1} F_{kj}(\underline{X}) \theta^j, \text{ where } 0 \leq k \leq n-1.$$

Thus the matrix representation of the linear form $L(\underline{X})$ is the matrix

$$\Lambda(\underline{X}) = \begin{pmatrix} F_{0,0}(\underline{X}) & ... & F_{0,n-1}(\underline{X}) \\ ... & ... & ... \\ F_{n-1,0}(\underline{X}) & ... & F_{n-1,n-1}(\underline{X}) \end{pmatrix},$$

where $F_{k,j}(\underline{X}) := \sum_{i=0}^{u-1} r_{i+k,j} X_i$; $(0 \leq k, j \leq n-1)$ are u-variable linear forms. Since $\mathcal{N}(\underline{x}) = \det(\Lambda(\underline{x}))$, the norm of $L(\underline{x})$ can be calculated by

determining the determinant $\det(\Lambda(\underline{x}))$. So the calculation of the desired function value is possible with integer arithmetic and Theorem 2 is proved. □

Lemma 2. *Let $T(X) = \sum_{s=0}^{n} t_s X^s \in \mathbb{Z}[X]$ be a monic irreducible polynomial over $\mathbb{Q}$ of degree n. Denote by θ one of the roots of $T(X)$. Then $\theta^{n+i} = \sum_{j=0}^{n-1} r_{n+i,j}\theta^j$, $i \geq 0$, where $r_{n,j} = -t_j$ for $j = 0, \ldots, n-1$ and*

$$r_{n+i+1,j} = \begin{cases} r_{n+i,j-1} - t_j r_{n+i,n-1} & \text{if } j \geq 1 \\ -t_0 r_{n+i,n-1} & \text{if } j = 0. \end{cases}$$

Proof. See chapter 4.2.2 of [3]. □

Now we summarize the main steps of the above construction, which enables us to determine the value $\mathcal{N}(\underline{x})$ from the given numbers $x_1, ..., x_u \in \mathbb{Z}$ and the coefficients $t_0, ..., t_n \in \mathbb{Z}$ of the polynomial $T(X)$:

1. The determination of the values $r_{i,j}$ with Newton's recursion formulae

In this step we calculate the values $r_{i,j}$ using Lemma 2.

This calculation can be done in advance since only the coefficients t_s of the polynomial $T(X)$ are needed.

The matrix representation of the linear form $L(\underline{X})$ will be the matrix $\Lambda(\underline{X})$ and its entries $F_{kj}(\underline{X})$ are u-variable linear forms having $r_{i+k,j}$ as their coefficients.

2. The determination of the values of the linear forms $F_{kj}(\underline{X})$

In this step the values of the n^2 pieces of $F_{kj}(\underline{X})$ are calculated at the vector $\underline{x} = (x_1, ..., x_u) \in \mathbb{Z}^u$.

3. The calculation of $\det(\Lambda)$

Here, we calculate the determinant of the matrix $\Lambda(\underline{x})$ which has rational integer entries.

Theorem 3. *The complexity of the computation of $\mathcal{N}(\underline{x})$ with integer arithmetic, using the algorithm described above is $O(n^7 + n^6 \log \mathbb{X} + n^5 \log^2 \mathbb{X})$, where the constant in O depends only on the coefficients of $T(X)$.*

Proof. First we estimate the complexity of the three steps above. The whole complexity of the algorithm is given by the sum of these.

For the calculation of each $r_{i,j}$ only the coefficients t_i of the polynomial $T(X) = \sum_{i=0}^{n} t_i X^i$ are needed. Let $\mathbb{X} = \max\{|x_i|\}$ and $t = \max\{|t_i|\}$. The binary lengths of $\mathbb{X}$ and t are $\log \mathbb{X}$ and $\log t$, respectively. For the calculation of the entries of the matrix of size $n \times (n-1)$ consisting of $r_{i,j}$ with the Newton's recursion formulae are needed altogether $n^2 - 2n$ multiplications, thus its complexity is

$$O(n^2). \tag{9}$$

The calculation of the values of the linear polynomials $F_{kj}(\underline{x})$ is possible with at most u multiplications. Since the length of $r_{i,j}$ is $C_1 n \log t = C_2 n$ and $u \leq n$, the complexity of the determination of the n^2 values is

$$O(n^4 \log \mathbb{X}). \tag{10}$$

The binary length of the numbers obtained is at most $C_3(C_2 n + \log \mathbb{X}) = C_4 n + C_3 \log \mathbb{X}$.

For the calculation of $\det(\Lambda(\underline{x}))$ the matrix $\Lambda(\underline{x})$ is reduced to triangular form by Gaussian-elimination and the product of the elements of the principal diagonal is taken. For this, however, operations in $\mathbb{Q}$ are to be done. This problem can be solved by the multiplication with the common denominator of each of the eliminated lines. At the end the value of the determinant obtained this way should be divided by these multipliers, but the result will surely be a rational integer. Let l denote the maximum length of the elements of $\Lambda(\underline{x})$. Thus the length of the numbers obtained by the elimination of the j-th column will be jl. Since it is highly probable that the denominators of the obtained numbers are coprime the determination of the common denominator requires $O(jl(jl-1+1)) = O(j^2 l^2)$ binary operations. The steps of the elimination can be done altogether by $O(n^3)$ operations, so to transform our matrix into triangular form using integer arithmetic is possible in at most $O(n^5 l^2)$ binary operation. The maximum length of the elements of the principal diagonal is nl, therefore $O\left(\frac{n^3-n^2}{2} l^2\right)$ binary operations are needed for the calculation of their product.

Since $l = C_4 n + C_3 \log \mathbb{X}$, for the calculation of the determinant at most

$$O\left(n^5 + \frac{n^3 - n^2}{2}\right)(C_4 n + C_3 \log \mathbb{X})^2 = O(n^7 + n^6 \log \mathbb{X} + n^5 \log^2 \mathbb{X}) \tag{11}$$

operations are needed.

The sum of (9), (10) and (11) is $O(n^7 + n^6 \log \mathbb{X} + n^5 \log^2 \mathbb{X})$ and Theorem 3 is proved. □

Remark 1. *Since the calculation of step 1 can be done in advance, in practice only steps 2 and 3 need to be applied with the knowledge of the numbers $x_1, ..., x_u$ in order to calculate the value $\mathcal{N}(\underline{x})$. The complexity of steps 2 and 3 basically corresponds with the one proved in Theorem 3.*

5. Calculation of $\mathcal{N}(\underline{x})$ with Modular Arithmetic

Now another version of the algorithm described in the previous section will be presented where the necessary operations are done by modular arithmetic.

Before describing the algorithm we outline the essence of the use of modular arithmetic.

In the present section mod m denotes the remainder function such that $-\left[\frac{m-1}{2}\right] \leq x \bmod m \leq \left[\frac{m}{2}\right]$, for every $x \in \mathbb{Z}$. Here [ ] denotes the integer part function.

Let $m_1, ..., m_v > 0$ be pairwise coprime integers and $M := m_1 m_2 \cdots m_v$. Then we assign to each $x \in \mathbb{Z}$ with $-\left[\frac{M-1}{2}\right] \leq x \leq \left[\frac{M}{2}\right]$ a v-tuple in the following way:

$$\varphi(x) = x^{(M)} = (x \bmod m_1, ..., x \bmod m_v).$$

The map φ: $x \leftrightarrow x^{(M)}$ is a ring homomorphism from $\mathbb{Z}$ to $\mathbb{Z}/(m_1) \times ... \times \mathbb{Z}/(m_v) \cong \mathbb{Z}/(m_1 \cdots m_v)$. The vector $x^{(M)}$ is called the modular representation of the rational integer x.

Let now $\circ$ denote addition, subtraction or multiplication. If we obtain that

$$-\left[\frac{M-1}{2}\right] \leq x, y, x \circ y \leq \left[\frac{M}{2}\right]$$

then the result of the operation $x \circ y$ can be computed with modular arithmetic in the following steps:

1. Determination of $x^{(M)}$ and $y^{(M)}$ with Euclidean division.
2. Calculation of $x^{(M)} \circ y^{(M)}$ in the residue class rings.
3. Determination of $x \circ y = \varphi^{-1}(x^{(M)} \circ y^{(M)})$ with CRA (Chinese Remainder Algorithm).

The advantage of the procedure outlined above is that in step 2 the operations can be done with relatively small integers. Its disadvantage is that we have to determine in advance an upper bound for the possible values of $x \circ y$. Furthermore, the operation of division can only be done in a complicated way by determining modulo inverse and the comparison of numbers is not possible at all.

In chapter 4.3 of [8] there is an effective Chinese Remainder Algorithm, which applies recursion and certain parameters can be calculated in advance reducing the complexity of the algorithm.

Now let us present the modular arithmetic version of the algorithm described in the previous section.

Again let $\mathbb{X} := \max\{|x_i|\}$, $T(X) := \prod_{i=0}^{n} t_i X^i$ and $t := \max\{|t_i|\}$. Then $|\mathcal{N}(\underline{x})| \leq B := n^{2n} t^{n^2} \mathbb{X}^n$. We shall chose moduli $m_i := p_i^{k_i}$, $k_i \in \mathbb{Z}_{\geq 0}$ for $1 \leq i \leq v$, with distinct primes $p_1, ..., p_v$ such that $M := p_1^{k_1} p_2^{k_2} \cdots p_v^{k_v} > 2B$. With such a choice we can guarantee

$$-\left[\frac{M-1}{2}\right] \leq \mathcal{N}(\underline{x}) \leq \left[\frac{M}{2}\right].$$

Now we summarize in 4 steps the algorithm which enables us to determine the function value $\mathcal{N}(\underline{x})$ at $x_1, ..., x_u \in \mathbb{Z}$ with the knowledge of the coefficients $t_0, ..., t_n \in \mathbb{Z}$ of the polynomial T using modular arithmetic. In advance are calculated prime power moduli $p_1^{k_1}, ..., p_v^{k_v}$ having the properties $p_i \leq C$ for $i = 1, \ldots, v$ with some constant C and $M := p_1^{k_1} p_2^{k_2} \ldots p_v^{k_v} > 2B$. In order to be able to guarantee $M > 2B$ it is enough to suppose $\prod_{p \leq C} p > 2B$, and since $\prod_{p \leq C} p > \sqrt{C}^{\pi(C)-\pi(\sqrt{C})}$ it is sufficient to have $\sqrt{C}^{\pi(C)-\pi(\sqrt{C})} > 2B$. Thus we suppose $C = C'\sqrt{\log(2B)}$.
The four main steps of our algorithm are the following:

1. Determination of the values $r_{i,j}$ with Newton's recursion formulae

In this step the modular arithmetic is not yet used since on the one hand the maximum length of the numbers $r_{i,j}$ is $Cn \log t$, on the other hand these calculations can be performed in advance as only the coefficients t_i of the polynomial $T(X)$ are needed. Hence, the numbers $r_{i,j}$ are determined according to the description in the previous section.

2. Determination of the values of the linear polynomials $F_{kj}(\underline{x})$

Here the modular arithmetic is already applied. In this step the values of the linear polynomials $F_{kj}(\underline{x}) \bmod p_1^{k_1}, ..., \bmod p_v^{k_v}$ are calculated at the vector $\underline{x} = (x_1, ..., x_u) \in \mathbb{Z}^u$. This way in fact we determine the matrices $\Lambda^{(1)}(\underline{x}), ..., \Lambda^{(v)}(\underline{x})$ which all have rational integer entries.

3. Calculation of $\det(\Lambda^{(M)}(\underline{x})) = \mathcal{N}^{(M)}(\underline{x})$

Now we determine the determinants of the matrices $\Lambda^{(1)}(\underline{x}), ..., \Lambda^{(v)}(\underline{x})$. This will be the modular representation $\left(\mathcal{N}^{(1)}(\underline{x}), \ldots, \mathcal{N}^{(v)}(\underline{x})\right)$ of the function value $\mathcal{N}(\underline{x})$.

4. Determination of the function value $\mathcal{N}(\underline{x})$

The value of $\mathcal{N}(\underline{x})$ is computed using the Chinese Remainder Algorithm from the modular representation $\left(\mathcal{N}^{(1)}(\underline{x}), \ldots, \mathcal{N}^{(v)}(\underline{x})\right)$.

Theorem 4. *The complexity of the determination of $\mathcal{N}(\underline{x})$ with modular arithmetic is $O(n^7 + n^6 \log \mathbb{X} + n^2 \log^{3/2} \mathbb{X})$. where the constant in O depends only on the coefficients of $T(X)$.*

Proof. First we estimate the complexity of the four steps described above. The whole complexity of the algorithm is given by the sum of these. Let $t := \max\{|t_i|\}$ and $P := \max\left\{p_i^{k_i}\right\}$.

Since the numbers $r_{i,j}$ are computed in the same way as in the algorithm presented in the previous section, also the complexity of this step is

$$O(n^2). \tag{12}$$

In order to compute the values of the linear polynomials $F_{kj}(\underline{x})$ we have to determine first the remainders $x_i \bmod p_1^{k_1}, ..., x_i \bmod p_v^{k_v}$, $i = 1, \dots, u$ and $r_{i,j} \bmod p_1^{k_1}, ..., r_{i,j} \bmod p_v^{k_v}$, $i = n, \dots, n-2$ and $j = 0, \dots, n-1$. This needs $O(vn \log P \log \mathbb{X} + vn^3 \log P)$ binary operations. ¿From now on the length of every occuring number can be at most $\log P$. Since $u \leq n$ the complexity of the determination of the n^2 values is $O(vn^3 \log^2 P)$. Thus the whole complexity of the second step is

$$O(vn \log P \log \mathbb{X} + vn^3 \log^2 P). \tag{13}$$

In the third step the determinants of the matrices $\Lambda^{(M)}(\underline{x}) \in \mathbb{Z}^{n \times n}$ are calculated. The complexity of the calculation of the computation of each determinant is $O(n^5 \log^2 P)$. Therefore, the complexity of the third step is

$$O(vn^5 \log^2 P). \tag{14}$$

In the fourth step we compute the function value $\mathcal{N}(\underline{x})$ from its modular representation $\left(\mathcal{N}^{(1)}(\underline{x}), \dots, \mathcal{N}^{(v)}(\underline{x})\right)$ using the Chinese Remainder Algorithm. Now Algorithm 4.1. of [8] will be used. The input of this will be $\left(\mathcal{N}^{(1)}(\underline{x}), \dots, \mathcal{N}^{(v)}(\underline{x})\right)$, $p_1^{k_1}, \dots, p_v^{k_v}$, and $q_2, ..., q_v$ and $s_2, ..., s_v$, where

$$q_i = \prod_{j=1}^{i-1} p_j^{k_j}, \quad s_i \equiv \prod_{j=1}^{i-1} s_j^{(i)} \bmod p_i^{k_i} \quad (i = 2, ..., v),$$

where

$$s_j^{(i)} p_j^{k_j} + s_i^{(j)} p_i^{k_i} = 1.$$

The numbers $s_j^{(i)}$ and $s_i^{(j)}$ can be determined by $O(\log^2 P)$ binary operations by the extended Euclidean algorithm (see Theorem 3.8 in [8]). The determination of the numbers s_i demands $O((v-1)^2 \log^2 P)$ binary operations, and for the calculation of the values q_i $O(\frac{1}{2}(v^2 - 3v + 2) \log^2 P)$ binary operations are needed. After these precomputations the CRA needs $O(v \log^2 P)$ binary operations. Therefore the whole complexity of the operations of the fourth step is

$$O(v^2 \log^2 P). \tag{15}$$

By (12), (13), (14) and (15) we can estimate the complexity of the whole algorithm by

$$O(v^2 n^5 \log^2 P + vn \log P \log \mathbb{X}). \tag{16}$$

Since $v \leq \frac{cC}{\log C}$ and $\log P \leq \log C$ we have $v \log P \leq cC$. Thus (16) takes the form

$$O(C^2 n^5 + Cn \log \mathbb{X}). \tag{17}$$

Further, since $C = C'\sqrt{\log(2B)}$ and $B = n^{2n}t^{n^2}\mathbb{X}^n$ we see that

$$C^2 \leq c_1 n^2 + c_2 n \log \mathbb{X}$$

and

$$C \leq c_3 n + c_4 n^{1/2} \log^{1/2} \mathbb{X} < c_5 n \log^{1/2} \mathbb{X}.$$

This shows that the complexity of the whole algorithm is at most

$$O(n^7 + n^6 \log \mathbb{X} + n^2 \log^{3/2} \mathbb{X})$$

and this was to be proved. □

Remark 2. *The calculation of step 1 can be done in advance and the numbers s_i and q_i of step 4 can also be determined in advance. However, the complexity basically corresponds with the one proved in Theorem 4.*

6. Comparison of the Algorithms

For the calculation of the function value $\mathcal{N}(\underline{x})$ three algorithms were discussed. The complexity of the algorithm using directly the definition of the norm form is $O(n^6 + n^4 \log^2 \mathbb{X})$. Its remarkable disadvantage is that the we have to do operations with real numbers with precision $\frac{1}{2nu(A\mathbb{X})^n}$.

The complexity of the algorithm using the matrix representation of the linear form $L(\underline{X})$ is $O(n^7 + n^6 \log \mathbb{X} + n^5 \log^2 \mathbb{X})$. However, its important advantage is that it computes with integers.

The complexity of the algorithm using the matrix representation with modular arithmetic is $O(n^7 + n^6 \log \mathbb{X} + n^2 \log^{3/2} \mathbb{X})$.

The above three algorithms have been implemented in *MAPLE V Release 4* (see [2]). Let us refer to these procedures as *ALG1*, *ALG2* and *ALG3*, respectively. These algorithms were also compared with an algorithm *ALG4* which uses the own *Norm()* procedure of *MAPLE*. All the four algorithms were run with randomly generated polynomials and data. Every algorithm was run with the same parameters 50 times and the running time was measured then later their average was taken. The following table shows the characteristic results of the test:

	n = 5 u = 4 Et = 10 Ex = 50	n = 5 u = 4 Et = 10 Ex = 70	n = 5 u = 4 Et = 10 Ex = 90	n = 5 u = 4 Et = 10 Ex = 110	n = 5 u = 4 Et = 10 Ex = 130	n = 5 u = 4 Et = 10 Ex = 150
ALG1	0.2427	0.4892	0.5942	0.6128	1.233	1.244
ALG2	0.0130	0.0115	0.0126	0.0215	0.0181	0.0206
ALG3	0.0697	0.1032	0.1727	0.3099	0.2329	0.2897
ALG4	0.0221	0.0220	0.0262	0.0395	0.0506	0.0542

In the table n denotes the degree of the minimal polynomial, u denotes the number of the variables of the linear form, *Et* denotes the size of the

coefficients of the minimal polynomial and Ex denotes the size of the substitution values.

The table shows that the fastest is the algorithm *ALG2*, which uses integer arithmetic and applies the matrix representation of the linear form. *ALG1* counts in the first test with precision of 400 digits and in the sixth with the precision of 900 digits, this is why its running time is the worst.

The increase of the substitution values slows down the most the algorithm *ALG1* and the least the algorithm *ALG2*. The real slowing factor is, however, the increase of the degree. The following table indicates this:

	n = 4 u = 4 Et = 10 Ex = 100	n = 5 u = 4 Et = 10 Ex = 100	n = 6 u = 5 Et = 10 Ex = 100	n = 7 u = 6 Et = 10 Ex = 100	n = 8 u = 7 Et = 10 Ex = 100	n = 9 u = 8 Et = 10 Ex = 100
ALG1	0.3996	0.6247	1.533	4.035	6.751	10.13
ALG2	0.0085	0.0160	0.0527	0.1161	0.2153	0.3811
ALG3	0.0757	0.1570	0.5232	1.137	2.245	4.316
ALG4	0.0245	0.0306	0.0581	0.0847	0.1383	0.2104

Here, in the sixth test *ALG1* counts with the precision of 1500 digits, so its running time increases a lot. Up to the third test the running time of *ALG2* is the best. In the additional tests *ALG4* will become the fastest. It should be noted that the algorithm *ALG4* uses the original *Norm()* procedure of the *MAPLE* which was not implemented in the *MAPLE*'s own language but in *C*. This partly explains the differences in speed.

The algorithm *ALG3* using modular arithmetic is not fast enough because it uses the *chrem()* Chinese Remainder Algorithm of the *MAPLE* in which the speeding possibilities deriving from the calculation in advance mentioned in Remark 2 cannot be utilized.

References

[1] Z. I. Borevich and I. R. Shafarevich, *Number Theory*, Academic Press, New-York, 1967, 2nd ed.

[2] B.W. Char, K.O. Geddes, G.H. Gonnet, B.L. Leong, M.B. Monagan and S.M. Watt, *Maple V Language reference manual*, Springer Verlag, 1990.

[3] H. Cohen, *A course in computational algebraic number theory*, Springer Verlag, 1993.

[4] J.-H. Evertse, K. Győry, *Finiteness criteria for decomposable form equations*, Acta Arith. **50** (1988), 357-379.

[5] J.-H. Evertse, K. Győry, C.L. Stewart and R. Tijdeman, *S-unit equations and their applications*, in: New Advances in Transcendence Theory (A. Baker ed.), Cambridge Univ. Press, 1988, pp. 110-174.

[6] P. Henrici, *Elements of numerical analysis*, Wiley and Sons, Inc. 1964.

[7] A. Karatsuba and Yu. Ofman, *Multiplication of Many-Digital Numbers by Automatic Computers*, Doklady Akad. Nauk SSSR **145** (1962), 293-294.

[8] A. Pethő, *Algebraische Algorithmen*, Vieweg, 1999.

[9] W.M. Schmidt, *Diophantine Approximation*, Lecture Notes in Mathematics **785** (1980), Springer Verlag, Berlin.
[10] A. Schönhage, *The fundamental theorem of algebra in terms of computational complexity (Preliminary report)*, unpublished manuscript.
[11] A. Schönhage and V. Strassen, *Schnelle Multiplikation grosser Zalen*, Computing **7** (1971), 281-292.
[12] F. Winkler, *Polynomial algorithms in computer algebra*, Springer Verlag, 1996.

A. Bérczes
Institute of Mathematics and Informatics
University of Debrecen
H-4010 Debrecen, P.O. Box 12, Hungary
E-mail address: berczesa@math.klte.hu

J. Ködmön
Faculty of Health College
University of Debrecen
H-4400 Nyíregyháza, Sóstói 2., Hungary
E-mail address: kodmonj@de-efk.hu