arXiv:0906.4917v1 [cs.SC] 26 Jun 2009

Detecting Simultaneous Integer Relation for Real Vectors and Finding the Minimal Polynomial of an Algebraic Number

Chen Jing-wei[b], Feng Yong[*,a,b], Qin Xiao-lin[b]

[a]*Lab. of Computer Reasoning and Trustworthy Comput., University of Electronic Science and Technology of China, Chengdu 610054, China*
[b]*Lab. for Automated Reasoning and Programming, Chengdu Institute of Computer Applications, CAS, Chengdu 610041, China*

Abstract

Let $x_1, \cdots, x_t \in \mathbb{R}^n$. A simultaneous integer relation for x_i is a vector $m \in \mathbb{Z}^n \setminus \{\mathbf{0}\}$ such that $m^T x_i = \mathbf{0}$ for $i = 1, \cdots, t$. Combining advantages of HJLS algorithm and PSLQ algorithm, we propose a numerically stable simultaneous integer relation detecting algorithm, which will construct a relation for $x_1, \cdots, x_t$ within $\mathcal{O}(n^4 + n^3 \log \lambda(\mathbf{x}))$ arithmetic operations, where $\lambda(\mathbf{x})$ is the least Euclidean norm of relations for $x_1, \cdots, x_t$. Unlike PSLQ algorithm only finds a Gauss (Hamilton) integer relation for a complex (quaternion) vector, our detecting algorithm can be used to find an integer relation (in $\mathbb{Z}^n$) for a vector with entries either in complex number field or in quaternion number field. This property enables us to present an algorithm for finding the minimal polynomial of an algebraic number, which is of degree and height less than n and H respectively, from its approximation. We give a sufficient condition on the error tolerance controlling, which guarantees the correctness of our minimal polynomial finding algorithm under floating-point arithmetic. We also prove that both the running time and bit-complexity of

[☆]This research was partially supported by the National Key Basic Research Project of China (2004CB318003), the Knowledge Innovation Program of the Chinese Academy of Sciences (KJCX2-YW-S02), and the National Natural Science Foundation of China (10771205).

[*]Corresponding author.

Email addresses: `velen@casit.ac.cn` (Chen Jing-wei), `yongfeng@casit.ac.cn` (Feng Yong)

our finding algorithm are better than already existing methods.

Key words: algebraic number, HJLS algorithm, minimal polynomial, PSLQ algorithm, simultaneous integer relation

1. Introduction

For a given vector $x \in \mathbb{R}^n$, we say x has an integer relation if there exists a nonzero vector $m \in \mathbb{Z}^n \setminus \{\mathbf{0}\}$ such that $x^T m = 0$. The problem of finding integer relations among sets of rational and real numbers is quite old. For two numbers (a_1, a_2), the problem can be solved by applying the Euclidean algorithm, or, equivalently, by computing the ordinary continued fraction expansion of the real number a_1/a_2. For $n \geq 3$, many detecting algorithms under the names generalized Euclidean algorithm and multidimensional continued fraction algorithm have been proposed. Historical surveys about it can be found in [1, 2, 3, 4, 5]. Among these integer relation detecting algorithms, HJLS algorithm [6, 4] and PSLQ algorithm [7, 5] have been used frequently. Let $x_1, \cdots, x_t \in \mathbb{R}^n$ be linearly independent [1] vectors. We denote $(x_1, \cdots, x_t)$ by $\mathbf{x}$. A simultaneous integer relation for $x_1, \cdots, x_t$ is a vector $m \in \mathbb{Z}^n \setminus \{\mathbf{0}\}$ such that $m^T\mathbf{x} = \mathbf{0}$, i.e. $m^T x_i = 0$ for $i = 1, \cdots, t$. We also say m is a simultaneous integer relation for $\mathbf{x}$. In [6, 4], the authors not only presented HJLS algorithm and the first rigorous proof of a 'polynomial time' bound for a relation finding algorithm but also proposed a simultaneous integer relation detecting algorithm, whereas it is numerically unstable. The unstable examples can be found in [7, 5]. In their draft [8], Rössner and Schnorr studied the case of $t = 2$ by using a modified HJLS algorithm. But for the moment, the paper is still in a preliminary state with some open problems. PSLQ algorithm, together with related lattice reduction schemes such as LLL [9], was named one of ten "algorithms of the twentieth century" by the publication *Computing in Science and Engineering* (see [10, 11]), and now is extensively used in Experimental Mathematics, such as identification of multiple zeta constants, a new formula for π, finding algebraic relations, Ising integrals and so on (see [12, 11, 13]). What's more, PSLQ algorithm is numerically stable and can be easily generalized to complex number field and Hamiltonian quaternion number field, but it is not suit for detecting simultaneous integer relation for real vectors. In this paper,

[1]This implies that none of x_i is a zero vector for $i = 1, \cdots, t$.

we present a numerically stable simultaneous integer relation detecting algorithm which combines the technique of constructing the hyperplane matrix in HJLS algorithm and the method of matrix reduction in PSLQ algorithm.

Let α be an algebraic number. The minimal polynomial of α is the unique primitive polynomial $p(x) \in \mathbb{Z}[x]$ of least degree such that $p(\alpha) = 0$. In this paper, we also consider such an interesting question: Suppose we are given an approximation to an algebraic number α, and a bound on the degree and size of the coefficients of its minimal polynomial. Is it possible to infer the minimal polynomial? The question was raised by Manuel Blum in theoretical cryptography and Zhang Jing-zhong in automated reasoning, independently. In fact, the first affirmative answer of this question, KLL algorithm, was presented by R. Kannan, A.K. Lenstra and L. Lovász in [14, 15] by using the celebrated lattice reduction algorithm LLL. In the computer algebra system *Maple*, the built-in function `PolynomialTools:-MinimalPolynomial()` that is an function to find a polynomial of degree n (or less) with small integer coefficients which has the given approximation r of an algebraic number as one of its roots is based on KLL algorithm. The correctness of the polynomial returned by the built-in function depends on the accuracy of the approximation. The minimal polynomial of an algebraic number α with exact degree n can be found by detecting an integer relation for the vector $v = (1, \alpha, \cdots, \alpha^n)^T$. Besides HJLS algorithm and PSLQ algorithm, B. Just also presented an algorithm to detect integer relations for a given vector consists of algebraic numbers in [16]. We can apply Just's algorithm or HJLS algorithm to the vector v for finding the minimal polynomial of α. However, both Just's algorithm and HJLS algorithm use LLL lattice basis reduction technique which depends upon an orthogonal decomposition of some kind. Thus these algorithms are not numerically stable [2]. Using PSLQ, one can find algebraic relations. There are many literatures involving this problem, for example [18, 19, 20, 21]. Based on PSLQ, the authors of this paper also presented an algorithm in [22] for finding the minimal polynomial of a real algebraic number from its approximation. However, these algorithms can not deal with complex algebraic numbers since PSLQ algorithm can be generalized to complex field and Hamiltonian quaternion number field, but the

[2]Recently, Ivan Morel, D. Stehlé, and G. Villad [17] introduce Householder transformations to LLL to approximate the underlying Gram-Schmidt orthogonalizations and present a new LLL-type lattice reduction algorithm, H-LLL, which may be used to overcome the numerical instability of orthogonal decomposition.

corresponding returns are in Gaussian integer ring and Hamiltonian integer ring respectively. Our simultaneous integer relation detecting algorithm can be used to overcome the pitfall. Applying our detecting algorithm to one or two real vectors, we present another affirmative answer, which is another minimal polynomial finding algorithm and more efficient than KLL algorithm, for Blum and Zhang's question and give a sufficient condition of the error controlling, from which we can claim that the polynomial returned by our minimal polynomial finding algorithm is the exact minimal polynomial of a given algebraic number.

Contributions and Road-map. In this paper, we firstly generalize PSLQ algorithm to a numerically stable simultaneous integer relation detecting algorithm combing with the technique of constructing the hyperplane matrix in HJLS algorithm. Analyzing our detecting algorithm under exact arithmetic gives that the cost of our algorithm is similar to PSLQ, i.e. $\mathcal{O}(n^4 + n^3 \log \lambda(\mathbf{x}))$ exact arithmetic operations, where $\lambda(\mathbf{x})$ represents the least Euclidean norm of relations for $\mathbf{x}$. We also implement our detecting algorithm under floating-point arithmetic. These contents are included in Section 2. Let $\alpha = a + bi \in \mathbb{C}$ be an algebraic number with degree at most n. In Section 3, we propose an algorithm for finding the minimal polynomial of α by applying the relation detecting algorithm in Section 2. Our finding algorithm saves a n factor both in arithmetic operations and bit-complexity compared with KLL algorithm. Floating-point arithmetic and high precision computing are needed when we implement our finding algorithm, for which we also analyze how to control the error and give a general conclusion. Let $\max_{1\leq i\leq n} |\alpha^i - \bar{\alpha}_i| < \epsilon$. Then $\epsilon < \frac{(1-\kappa)M}{n\,H}$ is sufficient to enable the output of our finding algorithm to be the exact minimal polynomial of α, where κ is a constant and satisfies $0 < \kappa < 1$, H is an upper bound of the height of α, and M is a function in n and H. The error tolerance is better than a similar one presented in [15]. Furthermore, our algorithm is numerically stable since it is based on a generalized PSLQ algorithm which is numerically stable. We give concluding remarks in Section 4.

Notations. All vectors in this paper are column vectors. We denote $n \times n$ identity matrix by I_n. If $x \in \mathbb{R}^n$, then $\|x\|_2$ represents its Euclidean norm, i.e. $\|x\|_2 = \sqrt{< x, x >}$, where $< *, * >$ is the inner product of two vectors. Given a matrix $A = (a_{i,j})$, we denote its transpose by A^T, its Frobenius norm by $\|A\|_F = (\mathrm{tr}(A^T A))^{1/2}$, i.e. $\|A\|_F = (\sum a_{i,j}^2)^{1/2}$. We say a matrix A is lower trapezoidal if $a_{i,j} = 0$ for $i < j$. $GL(n, \mathbb{Z})$ is the group of unimodular matrix with entries in $\mathbb{Z}$. The height of a vector or a matrix is defined by the maximum of all the absolute value of its entries. For a polynomial $f(x) = \sum_{i=0}^{n} f_i x^i$, we denote by $\deg(f)$ its degree in x, $\|f\|_1 = \sum_{i=0}^{n} |f_i|$ its one norm, $\|f\|_2 = (\sum_{i=0}^{n} |f_i|^2)^{1/2}$ its Euclid length, and $height(f) = \max_{0 \leq i \leq n} |f_i|$ its height. The height of an algebraic number is the height of its minimal polynomial. The notation $\lfloor t \rceil$ denotes an arbitrary integer closest to t, i.e. $\lfloor t \rceil = \lfloor t + \frac{1}{2} \rfloor$.

2. A Simultaneous Integer Relation Detecting Algorithm

2.1. Hyperplane Matrix

In this sectrion, we suppose that $x_1, \cdots, x_t$ are linearly independent vectors in $\mathbb{R}^n$ and denote $(x_1, \cdots, x_t)$ by $\mathbf{x}$.

Definition 2.1. Let $\mathbf{x} = (x_1, \cdots, x_t) \in \mathbb{R}^{n \times t}$. A hyperplane matrix W with respect to $\mathbf{x}$ is any matrix such that $\mathbf{x}W = 0$ and the columns of W span $\mathbf{x}^{\perp} = \{ y \in \mathbb{R}^n : x_i^T y = 0, i = 1, \cdots, t\}$.

Given $\mathbf{x} = (x_1, \cdots, x_t) \in \mathbb{R}^{n \times t}$, where $x_i = (x_{i,1}, \cdots, x_{i,n})^T$ are linearly independent for $i = 1, \cdots, t$, we now introduce a method to construct a hyperplane matrix for $\mathbf{x}$. Let $b_1, \cdots, b_n$ form a standard basis of $\mathbb{R}^n$, i.e. the i-th entry of b_i is 1 and others are 0. By performing the process of standard Gram-Schmidt orthogonalization (or orthonormalization) we have

$$\begin{aligned}
x_1^* &= \frac{x_1}{\|x_1\|_2}, \\
x_k^* &= x_k - \sum_{j=1}^{k-1} \nu_{k,j} x_j^*, \quad x_k^* = \frac{x_k^*}{\|x_k^*\|_2}, \quad k = 2, \cdots, t, \\
b_1^* &= b_1 - \frac{< b_1, x_1^* >}{< x_1^*, x_1^* >} x_1^* - \cdots - \frac{< b_1, x_t^* >}{< x_t^*, x_t^* >} x_t^*, \quad b_1^* = \frac{b_1^*}{\|b_1^*\|_2}, \qquad (2.1) \\
b_i^* &= b_i - \frac{< b_i, x_1^* >}{< x_1^*, x_1^* >} x_1^* - \cdots - \frac{< b_i, x_t^* >}{< x_t^*, x_t^* >} x_t^* - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*, \\
b_i^* &= \frac{b_i^*}{\|b_i^*\|_2}, \quad i = 2, \cdots, n,
\end{aligned}$$

where

$$\nu_{k,j} = \frac{< x_k,\, x_j^* >}{< x_j^*,\, x_j^* >}, \qquad \mu_{i,j} = \begin{cases} \dfrac{< b_i,\, b_j^* >}{< b_j^*,\, b_j^* >}, & \text{if } \|b_j^*\|_2 \neq 0, \\ 0, & \text{if } \|b_j^*\|_2 = 0. \end{cases}$$

Lemma 2.2. *Let $x_1^* \cdots, x_t^*$ and b_i^* be as in equation (2.1). Then*

1. *there exist $j_1 \cdots, j_t$ in $\{1, \cdots, n\}$ such that $b_{j_1}^* = \cdots = b_{j_t}^* = \mathbf{0}$.*
2. *if we denote all the nonzero vectors of $b_1^*, \cdots, b_n^*$ by $H_1, \cdots, H_{n-t}$ in-order, and $H_{\boldsymbol{x}} = (H_1, \cdots, H_{n-t})$, then $H_{\boldsymbol{x}}$ is a trapezoidal matrix and the matrix $(x_1^*, \cdots, x_t^*, H_{\boldsymbol{x}})$ is orthogonal.*
3. *if*
$$\begin{vmatrix} x_{1,n-t+1} & x_{2,n-t+1} & \cdots & x_{t,n-t+1} \\ x_{1,n-t+2} & x_{2,n-t+2} & \cdots & x_{t,n-t+2} \\ \vdots & \vdots & \cdots & \vdots \\ x_{1,n} & x_{2,n} & \cdots & x_{t,n} \end{vmatrix} \neq 0, \tag{2.2}$$
then $b_{n-t+1}^* = \cdots = b_n^* = \mathbf{0}$. In this case, every diagonal element of $H_{\boldsymbol{x}}$ is nonzero.

Proof. Part 1 and part 2 easily follows from the process of standard Gram-Schmidt orthogonalization. We next prove $b_{n-t+1}^* = \cdots = b_n^* = \mathbf{0}$ when equation (2.2) holds. Set

$$a_1x_1 + \cdots + a_tx_t + l_1b_1 + \cdots + l_{n-t}b_{n-t} = 0.$$

Observing the last t entries of the equation above gives $a_1 = \cdots = a_t = 0$. And since $b_1, \cdots, b_{n-t}$ are linearly independent, we have $l_1 = \cdots = l_{n-t} = 0$. Thus the n vectors $x_1, \cdots, x_t, b_1, \cdots, b_{n-t}$ are linearly independent, which implies $b_{n-t+1}^* = \cdots = b_n^* = \mathbf{0}$. In this case, the diagonal elements of $H_{\mathbf{x}}$ are $b_{i,i}^*$ for $i = 1 \cdots, n-t$. Before normalizing b_i^* we have

$$b_{i,i}^* = 1 - \sum_{k=1}^{t} x_{k,i}^{*2} - \sum_{j=1}^{i-1} b_{j,i}^{*2}.$$

And at the same time,

$$0 \neq \|b_i^*\|_2^2 =< b_i^*,\, b_i^* >= 1 - \sum_{k=1}^{t} x_{k,i}^{*2} - \sum_{j=1}^{i-1} b_{j,i}^{*2}.$$

Thus all the diagonal elements of $H_{\mathbf{x}}$ are nonzero. □

In the rest of this paper, we always suppose that $\mathbf{x} \in \mathbb{R}^{n\times t}$ satisfies equation (2.2). Otherwise we set $\mathbf{x}' = C\,\mathbf{x}$ which satisfies equation (2.2), where C is an appropriate matrix in $GL(n, \mathbb{Z})$, and detect a simultaneous integer relation for $\mathbf{x}'$. If m is a relation for $\mathbf{x}'$, then $C^T m$ is a simultaneous integer relation for $\mathbf{x}$.

Lemma 2.3. *Let $\boldsymbol{x} \in \mathbb{R}^{n\times 2}$ and $H_{\boldsymbol{x}}$ be as above. Then*

1. $H_{\boldsymbol{x}}^T\, H_{\boldsymbol{x}} = I_{n-t}$*;*
2. $\|H_{\boldsymbol{x}}\|_F = \sqrt{n-t}$*;*
3. $\boldsymbol{x}^T\, H_{\boldsymbol{x}} = \boldsymbol{0}$*, i.e. $H_{\boldsymbol{x}}$ is a hyperplane matrix of $\boldsymbol{x}$.*

Proof. Since every two columns of $H_{\mathbf{x}}$ are orthogonal, part 1 follows. And part 2 follows from part 1. Let $\mathbf{x}^* = (x_1^*, \cdots, x_t^*)^T$. From the process of standard Gram-Schmidt orthogonalization we have $\mathbf{x}^{*T} H_{\mathbf{x}} = \mathbf{0}$ and $\mathbf{x} = \mathbf{x}^* Q$ where Q is an appropriate orthogonal matrix, which means $\mathbf{x}^T H_{\mathbf{x}} = Q^T x^{*T} H_{\mathbf{x}} = \mathbf{0}$. $\square$

So, we have a method to product a hyperplane matrix for a given $\mathbf{x} \in \mathbb{R}^{n\times t}$, which is from HJLS algorithm (see [6, 4]). The same strategy was also used in PSLQ algorithm (see [5]), however, in which partial sum was adopted instead of Gram-Schmidt orthogonalization.

Lemma 2.4. *For $\boldsymbol{x} = (x_1, \cdots, x_2) \in \mathbb{R}^{n\times t}$ define $P_{\boldsymbol{x}} = H_{\boldsymbol{x}} H_{\boldsymbol{x}}^T$. Then*

1. $P_{\boldsymbol{x}}^T = P_{\boldsymbol{x}}$*;*
2. $P_{\boldsymbol{x}} = I_n - \sum_{i=1}^t x_i^*\, x_i^{*T}$*;*
3. $P_{\boldsymbol{x}}^2 = P_{\boldsymbol{x}}$*;*
4. $\|P_{\boldsymbol{x}}\|_F = \sqrt{n-t}$*;*
5. $P_{\boldsymbol{x}}\, z = z$ *for any* $z \in \boldsymbol{x}^\perp$*. Particularly,* $P_{\boldsymbol{x}}\, m = m$ *for any simultaneous integer relation* $m \in \mathbb{Z}^n$ *for* $\boldsymbol{x}$*.*

Proof. The proof of the first part is easy. Let $U = (x_1^*, \cdots, x_t^*, H_{\mathbf{x}})$. From Lemma 2.2 we have $I_n = U\,U^T = H_{\mathbf{x}}\, H_{\mathbf{x}}^T + \sum_{i=1}^t x_i^*\, x_i^{*T}$. Thus part 2 follows. Part 3 and part 4 follow from

$$\begin{aligned} P_{\mathbf{x}}^2 &= (I_n - \sum_{i=1}^{t} x_i^* \, x_i^{*T})^2 \\ &= I_n - 2\, I_n \sum_{i=1}^{t} x_i^* \, x_i^{*T} + (\sum_{i=1}^{t} x_i^* \, x_i^{*T})^2 \\ &= I_n - \sum_{i=1}^{t} x_i^* \, x_i^{*T} = P_{\mathbf{x}} \end{aligned}$$

and $\|P_{\mathbf{x}}\|_F^2 = \mathrm{tr}(P_{\mathbf{x}}^T \, P_{\mathbf{x}}) = \mathrm{tr}(P_{\mathbf{x}}) = \mathrm{tr}(H_{\mathbf{x}}^T \, H_{\mathbf{x}}) = n - t$ respectively. Since $z \in \mathbf{x}^{\perp}$, we have $< x_i, \, z >= 0$ for $i = 1 \cdots, t$. And the process of standard Gram-Schmidt orthogonalization implies $< x_i^*, \, z >= 0$. So $P_{\mathbf{x}} \, z = z - (\sum_{i=1}^{t} x_i^* \, x_i^{*T}) z = z$. □

From Lemma 2.2 – 2.4 we can easily generalize the Theorem 1 in [5] to the case of $\mathbf{x} \in \mathbb{R}^{n \times t}$.

Theorem 2.5. *Let $\boldsymbol{x} \in \mathbb{R}^{n \times t}$ as above. Suppose that for any simultaneous integer relation for $\boldsymbol{x}$ and for any matrix $A \in GL(n, \, \mathbb{Z})$ there exists a orthogonal matrix $Q \in \mathbb{R}^{(n-t) \times (n-t)}$ such that $H = A H_{\boldsymbol{x}} Q$ is lower trapezoidal and all of the diagonal elements of H satisfy $h_{j,j} \neq 0$. Then*

$$\frac{1}{\max_{1 \leq j \leq n-2} |h_{j,j}|} = \min_{1 \leq j \leq n-2} \frac{1}{|h_{j,j}|} \leq |m|. \tag{2.3}$$

Theorem 2.5 easily follows from the proof of Theorem 1 of [5] with little modifications. So we omit it here. Theorem 2.5 suggests a strategy to construct a simultaneous integer relation detecting algorithm: Find a way to reduce the norm of the matrix $H_{\mathbf{x}}$ by multiplication by some unimodular A on the left. The inequality of Theorem 2.5 offers an increasing lower bound on the size of any possible relation. Theorem 2.5 can be used with any algorithm that produces any $GL(n, \, \mathbb{Z})$ matrices. Any $GL(n, \, \mathbb{Z})$ matrix A whatsoever can be put into Theorem 2.5. Additionally, the lower bound given in (2.3) is consistent with a similar lower bound in [2, 23].

2.2. Matrix Reduction

We now study many kinds of matrix reduction in this subsection. Firstly, we recall Hermite reduction and modified Hermite reduction in [5].

Definition 2.6 (**Modified Hermite reuction**). Let H be a lower trapezoidal matrix with $h_{j,j} \neq 0$ and set $D = I_n$. For i from 2 to n, and for j from $i-1$ to 1 by step -1, set $q = \lfloor h_{i,j}/h_{j,j} \rceil$; then for k from 1 to j replace $h_{i,k}$ by $h_{i,k} - qh_{j,k}$, and for k from 1 to n, replace $d_{i,k}$ by $d_{i,k} - qd_{j,k}$. We say DH is the Hermite reduction of H and D is the reducing matrix of H, where $\lfloor t \rceil = \lfloor t + 1/2 \rfloor$.

This reduction replaces the input H with DH while developing the left multiplying reduction matrix D. Hermite reduction, which is equivalent to modified Hermite reduction (see Lemma 3 in [5]) for a lower triangular matrix H with $h_{j,j} \neq 0$, is also presented in [5]. The two equivalent reductions have the following properties:

1. The reducing matrix $D \in GL(n, \mathbb{Z})$.
2. For all $k > i$, the Hermite reduced matrix $H' = (h'_{i,j}) = DH$ satisfies $|h'_{k,i}| \leq |h'_{i,i}|/2 = |h_{i,i}|/2$.

In order that the reduced matrix of $H_{\mathbf{x}} \in \mathbb{R}^{n\times(n-t)}$ of $\mathbf{x}$ satisfies the two properties above, we need generalized Hermite reduction as Definition 2.7.

Definition 2.7 (**Generalized Hermite reuction**). Let $H \in \mathbb{R}^{n\times(n-t)}$ and the (modified) Hermite reduced matrix of H be DH. For i from $n-1$ to n, for j from $n-t$ to 1, set $q = \lfloor h_{i,j}/h_{j,j} \rceil$; for k from 1 to n, set $d_{i,k} = d_{i,k} - qd_{j,k}$; set $H = DH$; For any s_1, $s_2 \in \{n-t+1, \cdots, n\}$ satisfy $s_1 < s_2$, if $h_{s_1,n-t} = 0$ and $h_{s_2,n-t} \neq 0$, then exchange the s_1-th row and the s_2-th row of D. We say DH is the generalized Hermite reduction of H, and that D is the reducing matrix of H.

We can easily check that generalized Hermite reduction remains the two properties above. Different from (modified) Hermite reduction, generalized Hermite reduction exchanges the s_1-th row and the s_2-th row of D if $h_{s_1,n-t} = 0$ and $h_{s_2,n-t} \neq 0$, which implies that if $h_{n-t+1,n-t} = 0$ after generalized Hermite reduction has been performed then $h_{n-t+2,n-t} = \cdots = h_{n,n-t} = 0$. This property is very important in the proof of Lemma 2.9.

2.3. The Algorithm Description

Applying the hyperplane matrix constructing method and generalize Hermite reduction in the previous subsetions we can get a simultaneous integer relation detecting algorithm (Algorithm 2.8).

Algorithm 2.8 (Simultaneous Integer Relation Detecting).

```
Input. $(x_1, \cdots, x_t) = \mathbf{x} \in \mathbb{R}^{n\times t}$, where $x_1, \cdots, x_t$ are linearly independent.
Output. either output a simultaneous integer relation for $\mathbf{x}$ or give a lower bound on $\lambda(\mathbf{x})$.
 1: Initiation. Compute the hyperplane matrix $H_{\mathbf{x}}$, set $H = H_{\mathbf{x}}$. Set $A = B = I_n$.
 2: Reduction. Perform generalize Hermite reduction on $H_{\mathbf{x}}$ producing the reducing matrix $D \in GL(n, \mathbb{Z})$. Set $\mathbf{x} = \mathbf{x}D^{-1}$, $H = DH$, $A = DA$, $B = BD^{-1}$.
 3: loop
 4:    Exchange. Let $H = (h_{i,j})$. Let
           $$\alpha = h_{r,r},\quad \beta = h_{r+1,r},\quad \lambda = h_{r+1,r+1},\quad \delta = \sqrt{\beta^2 + \lambda^2}. \qquad (2.4)$$
       Choose an integer $r$ such that $\gamma^r|h_{r,r}| \geq \gamma^i|h_{i,i}|$ for $1 \leq i \leq n - t$, where $\gamma > 2/\sqrt{3}$. Define the permutation matrix $R$ to be the identity matrix with the $r$ and $r+1$ rows exchanged. Update $\mathbf{x} = \mathbf{x}R$, $H = RH$, $A = RA$, $B = BR$.
 5:    Corner. At this point $\lambda$ may not be zero makes that $H$ may not be lower trapezoidal. Let $Q = I_{n-t}$. If $r < n - t$, then let the submatrix of $Q$ consisting of the $r$ and $r + 1$ rows of columns $r$ and $r + 1$ be $\begin{pmatrix} \beta/\delta & -\lambda/\delta \\ \lambda/\delta & \beta/\delta \end{pmatrix}$, where $\alpha$, $\beta$, $\lambda$, $\delta$ are defined in (2.4). Replace $H$ by $HQ$.
 6:    Reduction. Perform generalize Hermite reduction on $H$, producing $D$. Update $\mathbf{x} = \mathbf{x}D^{-1}$, $H = DH$, $A = DA$, $B = BD^{-1}$.
 7:    Compute $G := 1/\max_{1\leq j\leq n-t}|h_{j,j}|$. Then there exists no relation vector whose Euclidean norm is less than $G$.
 8:    if $x_j = 0$ for some $1 \leq j \leq n$, or, $h_{i,i} = 0$ for some $1 \leq i \leq n-t$ then
 9:       return the corresponding relation for $\mathbf{x}$.
10:    end if
11: end loop
```

2.4. Analysis of Our Detecting Algorithm

Let $H(k)$ be the result after k iterations of Algorithm 2.8.

Why do we set the parameter $\gamma > 2/\sqrt{3}$ at the beginning of the iteration step?

We firstly answer this question. Suppose the r chosen in Step 4 is not $n - t$. Let

$$\begin{pmatrix} \alpha & 0 \\ \beta & \lambda \end{pmatrix}$$

be the submatrix of $H(k-1)$ consisting of the r and $r+1$ rows of columns r and $r+1$. After Step 4 has been performed this submatrix becomes

$$\begin{pmatrix} \beta & \lambda \\ \alpha & 0 \end{pmatrix}.$$

Since r is chosen such that $\gamma^r|h_{r,r}(k-1)|$ is as large as possible, if $r < n-t$ then $|h_{r+1,r+1}(k-1)| \le \frac{1}{\gamma}|h_{r,r}(k-1)|$. In this case we let α, β, λ, δ be as in (2.4). Then $|\lambda| \le \frac{1}{\gamma}|\alpha|$. Let

$$\begin{pmatrix} \alpha & 0 \\ \beta & \lambda \end{pmatrix}$$

be the submatrix of $H(k-1)$ consisting of the r and $r+1$ rows of columns r and $r+1$, where $r < n-t$. After Step 5 has been performed we then replace $h_{r,r}(k-1)$ with δ and the result is

$$\begin{pmatrix} \beta & \lambda \\ \alpha & 0 \end{pmatrix} \begin{pmatrix} \beta/\delta & -\lambda/\delta \\ \lambda/\delta & \beta/\delta \end{pmatrix} = \begin{pmatrix} \delta & 0 \\ \alpha\beta/\delta & -\alpha\lambda/\delta \end{pmatrix}.$$

From the reduction of $H(k-1)$ we have that $|\beta| \le \frac{1}{2}|\alpha|$, which then gives

$$|\frac{\delta}{\alpha}| = \sqrt{\frac{\beta^2 + \lambda^2}{\alpha^2}} \le \sqrt{\frac{1}{4} + \frac{1}{\gamma^2}}. \tag{2.5}$$

Thus $|h_{r,r}(k-1)|$ is reduced as long as $\sqrt{\frac{1}{4} + \frac{1}{\gamma^2}} < 1$, i.e. $\gamma > 2/\sqrt{3}$. As is pointed out by Borwein (see Appendix B in [24]), although this increases $h_{r+1,r+1}(k)$, this is not a significant problem. At each step we force the larger diagonal elements of $H(k-1)$ toward $h_{n-t,n-t}(k)$, where their size can be reduced by at least a factor of 2 when $r = n-t$.

Lemma 2.9. *If $h_{j,j}(k) = 0$ for some $1 \le j \le n-t$ and no smaller k, then $j = n-t$ and a simultaneous integer relation for $\boldsymbol{x}$ must appear as a column of the matrix B.*

Proof. By the hypothesis on k we know that all diagonal elements of $H(k-1)$ is not zero. Now, suppose the r chosen in Step 4 is not $n-t$. Since generalized Hermite reduction does not introduce any new zeros on the diagonal, and by the analysis of Step 4 and 5 in the beginning of this subsection, we have that no diagonal elements of $H(k)$ is zero. This contradicts the hypothesis on k and our assumption that $r < n-t$ was false. Thus we have $r = n-t$ after the $(k-1)$-th iteration has been completed. In this case, $h_{n-t,\,n-t}(k) = 0$ implies $h_{n-t+1,\,n-t}(k-1) = 0$, and $h_{n-t,\,n-t}(k-1) \neq 0$. And from Definition 2.7, we have that $h_{n-t+1,\,n-t}(k-1) = 0$ must imply $h_{n-t+2,\,n-t}(k-1) = \cdots = h_{n,\,n-t}(k-1) = 0$.

Next we show that there must be a simultaneous integer relation for $\mathbf{x}$ appeared as a column of the matrix B. We have $\mathbf{x}H_{\mathbf{x}} = 0$ from 2.3. At each step in Algorithm 2.8, we have $BA = I_n$, and hence $0 = \mathbf{x}^T BAH_{\mathbf{x}} = \mathbf{x}^T BAH_{\mathbf{x}}Q = \mathbf{x}^T BH(k-1)$, where Q is an appropriate orthogonal $(n-t) \times (n-t)$ matrix. Let $(z_1, \cdots, z_t)^T = \mathbf{x}^T B$. Then

$$\begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \cdots & \vdots \\ 0 & \cdots & 0 \end{pmatrix} = \mathbf{x}^T BH(k-1) = \begin{pmatrix} z_1^T \\ \vdots \\ z_t^T \end{pmatrix} H(k-1)$$

$$= \begin{pmatrix} \cdots, & \sum_{k=n-t}^{n} z_{1,k} h_{k,n-t}(k-1) \\ \cdots, & \cdots \\ \cdots, & \sum_{k=n-t}^{n} z_{t,k} h_{k,n-t}(k-1) \end{pmatrix} = \begin{pmatrix} \cdots, & z_{1,n-t} h_{n-t,\,n-t}(k-1) \\ \cdots, & \cdots \\ \cdots, & z_{t,n-t} h_{n-t,\,n-t}(k-1) \end{pmatrix}.$$

Since $h_{n-t,\,n-t}(k-1) \neq 0$, we have $z_{1,\,n-t} = \cdots = z_{t,\,n-t} = 0$. Thus the $(n-t)$-th column of B is a simultaneous integer relation of $\mathbf{x}$. □

From Lemma 2.9 the correctness of Algorithm 2.8 has been proved. And going along with the routine of analyzing the PSLQ algorithm in [5], we can get the cost of the algorithm. So we state the conclusion directly without proof.

Theorem 2.10. *If $\boldsymbol{x} \in \mathbb{R}^{n\times 2}$ has simultaneous integer relations, then Algorithm 2.8 can be made to find one using $\mathcal{O}(n^4 + n^3 \log \lambda(\boldsymbol{x}))$ exact arithmetic operations, where $\lambda(\boldsymbol{x})$ is the least Euclidean norm of relations for $\boldsymbol{x}$.*

Remark 2.11. Besides these properties above, Algorithm 2.8 has many other good properties. We enumerate them as follows:

1. Algorithm 2.8 will produce lower bound on the Euclidean norm of any possible simultaneous integer relation for $\mathbf{x}$ (Theorem 2.5). Thus Algorithm 2.8 can be used to prove that there are no simultaneous relations for $\mathbf{x}$ of norm less than a given size.
2. Algorithm 2.8 is a generalization of PSLQ algorithm, so we could take PSLQ algorithm as a particular case of $t = 1$. PSLQ algorithm can easily be used to complex field and Hamiltonian quaternion number field, but the corresponding returns are in Gaussian integer ring and Hamiltonian integer ring respectively. However, Algorithm 2.8 can also be used to detect integer relation for a given complex vector. For example, suppose $z = x + yi$ in $\mathbb{C}^n$ with vector components $x,\, y \in \mathbb{R}^n$. Then Algorithm 2.8 can give a simultaneous integer relation m for $(x,\, y)$ in $\mathbb{Z}^n$, not in $\mathbb{Z}[i]^n$, and m is an integer relation for z. So do quaternion numbers. This is the biggest difference of Algorithm 2.8 from PSLQ algorithm.
3. Algorithm 2.8 construct the hyperplane matrix of $\mathbf{x}$ based on HJLS algorithm, however, the matrix reduction method is generalized Hermite reduction, which avoids LLL reduction. This is not only a difference between Algorithm 2.8 and HJLS algorithm, but also between Algorithm 2.8 and PSLQ algorithm because that (modified) Hermite reduction is not suit any more for detecting simultaneous integer relation. And just generalized Hermite reduction guarantees the correctness of Algorithm 2.8. In addition since the process of iteration is based on LQ decomposition (this is equivalent to QR decomposition), our detecting algorithm is numerically stable.
4. The parameter γ can be freely chosen in the open interval $(2/\sqrt{3},\, \infty)$.

2.5. Practical Implementation

We usually use floating-point arithmetic instead of exact arithmetic for speeding up algorithms. For PSLQ algorithm, Bailey and Broadhurst [25, 12] indicated that if one wishes to recover a relation $m \in \mathbb{Z}^n$ with coefficients of at most d digits in size, then the input vector x must be specified to at least nd digits and one must employ floating-point arithmetic accurate to at least nd digits.

In these cases, multiple precision arithmetic is used. *Maple* and *Mathematica* include multiple precision arithmetic facilities. One may also use any of several freeware multiprecision software packages, for example the ARPREC package [26]. Unfortunately, we do recover a suspected relation m using inexact arithmetic, but there is no guarantee that it is a true integer relation for x. However, it may be worthwhile to search for a rigorous proof. So PSLQ algorithm is extensively used in Experimental Mathematics, such as identification of multiple zeta constants, a new formula for π, finding algebraic relations, Ising integrals and so on (see [12, 11, 13]).

Similarly, it is often too far costly to find a simultaneous integer relation using exact arithmetic operation. So instead of the input $\mathbf{x} \in \mathbb{R}^{n\times t}$ we would like to let the value $\mathbf{x}'$ be a good rational approximation to $\mathbf{x}$ and attempt to detect a simultaneous integer relation for $\mathbf{x}$ using inexact arithmetic. At the end of every iteration, if we look only at the values

$$\max_{i\in\{1,\cdots,t\}}\{\frac{|x_i^T\cdot b_j|}{\|x_i\|_2}\},\ j = 1, \cdots, n,$$

then we may miss a relation when $\|b_j\|_2$ is large enough. Many experiments show that if we control error appropriately, Algorithm 2.8 under floating-point arithmetic will return an exact simultaneous integer relation for $\mathbf{x}$. Thus we take

$$\max_{i\in\{1,\cdots,t\}}\{\frac{|x_i^T\cdot b_j|}{\|x_i\|_2\cdot\|b_j\|_2}\} < \varepsilon$$

as a part of termination in Algorithm 2.8 when we implement it under floating-point arithmetic, where ε depends on the level of precision being used. It is difficult to choose such an ε because it may be very different depending on different problems. In addition, when we implement Algorithm 2.8 by using floating-point arithmetic, we will encounter the same question of PSLQ as mentioned above: we do recover a suspected relation by but there is no guarantee that it is a true simultaneous integer relation for $\mathbf{x}$. However, for finding the minimal polynomial of an algebraic number, we overcome the pitfalls and present a general conclusion about the error tolerance in Section 3.

In fact, we implement Algorithm 2.8 when $t = 2$ under floating-point arithmetic in *Maple* for more further discussions about finding the minimal polynomial for an algebraic number. The authors' *Maple* implementation is available by sending e-mail to `velen@casit.ac.cn`.

3. Finding Minimal Polynomial

Let $\alpha = a + bi \in \mathbb{C}$ be an algebraic number with degree at most n. The minimal polynomial of α is the unique primitive polynomial $p(x) \in \mathbb{Z}[x]$ of least degree such that $p(\alpha) = 0$. Let $v = (1, \alpha, \cdots, \alpha^n)^T$, and v_1, v_2 consist of the real part, image part respectively of each element of v, i.e. $v_1 = (1, \mathrm{Re}(\alpha), \cdots, \mathrm{Re}(\alpha^n))^T$, $v_2 = (0, \mathrm{Im}(\alpha), \cdots, \mathrm{Im}(\alpha^n))^T$.

3.1. Minimal Polynomial Finding Algorithm

Some computer algebra systems, such as *Maple*, have an LLL-based procedure for finding minimal polynomial. The procedure can find a simultaneous integer relation for (v_1, v_2) but can not find a simultaneous integer relation for two arbitrary vectors. The basic ideal is from [14, 15]. Of course the simultaneous integer relation detecting Algorithm 2.8 when $t = 1\,(b = 0)$ or $2\,(b \neq 0)$ can be used to find the minimal polynomial of α directly. We propose the algorithm as follows.

Remark 3.2. Let $\alpha = a + bi$. Worthy of being mentioned, we have that equation (2.2) always holds when $b \neq 0$ since

$$\begin{vmatrix} \mathrm{Re}(\alpha^i) & \mathrm{Re}(\alpha^{i+1}) \\ \mathrm{Im}(\alpha^i) & \mathrm{Im}(\alpha^{i+1}) \end{vmatrix} = b(a^2 + b^2)^i, \text{ for } i = 0, 1, \cdots . \tag{3.1}$$

Thus we can call Algorithm 2.8 directly without preprocessing (v_1, v_2). We check equation (3.1) here. Since

$$\mathrm{Re}(\alpha^i) = \sum_{k=0}^{\lfloor \frac{i}{2} \rfloor} (-1)^k \begin{pmatrix} i \\ 2k \end{pmatrix} a^{i-2k} b^{2k},$$

$$\mathrm{Im}(\alpha^i) = \sum_{k=1}^{\lceil \frac{i}{2} \rceil} (-1)^k \begin{pmatrix} i \\ 2k-1 \end{pmatrix} a^{i-2k+1} b^{2k-1}$$

for $i = 0, 1, \cdots$, we have

$$\mathrm{Re}(\alpha^{i+1}) = a\mathrm{Re}(\alpha^i) - b\mathrm{Im}(\alpha^i), \qquad \mathrm{Im}(\alpha^{i+1}) = a\mathrm{Im}(\alpha^i) + b\mathrm{Re}(\alpha^i).$$

Thus

$$\begin{vmatrix} \mathrm{Re}(\alpha^i) & \mathrm{Re}(\alpha^{i+1}) \\ \mathrm{Im}(\alpha^i) & \mathrm{Im}(\alpha^{i+1}) \end{vmatrix} = b(\mathrm{Re}(\alpha^i)^2 + \mathrm{Im}(\alpha^i)^2) = b|\,\alpha^i|^2 = b|\,\alpha|^{2i} = b(a^2 + b^2)^i.$$

Algorithm 3.1 (Minimal Polynomial Finding).

Input. an algebraic number $a + b\,i = \alpha \in \mathbb{C}$, an upper bound n on the degree of α, and an upper bound G on the Euclidean norm of the minimal polynomial of α.

Output. $p(x) \in \mathbb{Z}[x]$, the minimal polynomial of α.

```
1: if $\mathrm{Im}(\alpha) = 0$ then
2:   for $i$ from 2 to $n$ do
3:     $v := (1, \alpha, \cdots, \alpha^i)^T$
4:     Call Algorithm 2.8 when $t = 1$ with input $v$ and the upper bound $G$. If $G$ is smaller than the bound returned by Step 7 of Algorithm 2.8, then continue[3].
5:     if Algorithm 2.8 return $(p_0, p_1, \cdots, p_i)$, an integer relation for $v$ then
6:       $p := \sum_{j=1}^{i} p_j x^{j-1}$
7:       return pp($p$), i.e. the primitive part of $p$
8:     end if
9:   end for
10: else
11:   for $i$ from 2 to $n$ do
12:     $v := (1, \alpha, \cdots, \alpha^i)^T$
13:     $v_1 := (1, \mathrm{Re}(v[2]), \cdots, \mathrm{Re}(v[i]))^T$
14:     $v_2 := (0, \mathrm{Im}(v[2]), \cdots, \mathrm{Im}(v[i]))^T$
15:     Call Algorithm 2.8 when $t = 2$ with input $(v_1, v_2)$ and the upper bound $G$. If $G$ is smaller than the bound returned by Step 7 of Algorithm 2.8, then continue.
16:     if Algorithm 2.8 return $(p_0, p_1, \cdots, p_i)$, a simultaneous integer relation for $(v_1, v_2)$ then
17:       $p := \sum_{j=1}^{i} p_j x^{j-1}$
18:       return pp($p$), i.e. the primitive part of $p$
19:     end if
20:   end for
21: end if
```

Theorem 3.3. *Algorithm 3.1 correctly returns the minimal polynomial of a complex algebraic number using at most $\mathcal{O}(n^5 + n^4 \log G)$ exact arithmetic operations, where G is an upper bound on the Euclidean norm of the minimal polynomial of α.*

Proof. Suppose the degree of α is $i_0 \le n$. By the condition in Step 5 (or Step 16), $(p_0, p_1, \cdots, p_{i_0})$ is a (simultaneous) integer relation for v (or (v_{1,i_0}, v_{2,i_0})), so we have $p(\alpha) = 0$. Denote the minimal polynomial of α by $h(x)$. Then $h(x)|\mathrm{pp}(p)$. Algorithm 3.1 goes into Step 5 (or Step 16) that means no smaller i satisfies that there is a (simultaneous) integer relation for v (or $(v_{1,i}, v_{2,i})$) with Euclidean norm $\le G$. Thus $h(x)$ must be $\mathrm{pp}(p)$. The upper bound of the degree of α guarantees the termination of Algorithm 3.1. When $i_0 = n$, the worst case, Algorithm 3.1 needs $\mathcal{O}(n^5 + n^4 \log G)$ exact arithmetic operations by Theorem 2.10. $\square$

3.2. Implementation of Minimal Polynomial Finding Algorithm

When we implement Algorithm 3.1 using floating-point arithmetic, we call Algorithm 2.8 under floating-point arithmetic as a subalgorithm. However, the output may be not a true simultaneous integer relation for $(v_{1,i}, v_{2,i})$, which leads the output of Algorithm 3.1 is not true. In this subsection, we overcome this pitfall and present a strategy, by which the correctness of our finding algorithm is guaranteed. Before describing it in detail, we consider an example as follows.

Example 3.4. Let $\alpha = 2 + \sqrt{3}\,i$. We know that the minimal polynomial of α in $\mathbb{Z}[x]$ is $x^2 - 4x + 7$. We first let $\bar{\alpha} = 2.000 + 1.732\,i$ be the approximation of α with four significant digits. Hence $v_1 = (1., 2., 1.)^T$, $v_2 = (0., 1.732, 6.928)^T$. Feeding Algorithm 2.8 under floating point arithmetic v_1, v_2 as its input vectors gives a simultaneous integer relation for v_1, v_2 after only 2 iterations. The corresponding matrices B are

$$\begin{pmatrix} 2 & 1 & 0 \\ -1 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 7 & 0 & 2 \\ -4 & 0 & -1 \\ 1 & 1 & 0 \end{pmatrix}$$

respectively. It is obvious that the first column of the latter one is the simultaneous integer relation for v_1, v_2, which corresponds to the coefficients of the minimal polynomial of α. If we take only 3 significant digits for the same data, Algorithm 2.8 returns $(-699,\ 400,\ -100)$ after 3 iterations. For this reason, we have to appropriately control the error such that the output of the algorithm is correct.

In the rest of this subsection, let $\alpha = a + bi \in \mathbb{C}$ be an algebraic number with degree at most n and H be an upper bound on the height of α. Let $\bar{\alpha}$ be an approximation to α with $\max_{1\le i\le n}|\alpha^i - \bar{\alpha}^i| < \epsilon$. In fact, when we implement Algorithm 3.1 using floating-point arithmetic, we just need insert "**if** $|p(\bar{\alpha})| < LB(n,\ H)$ **then**" between Step 17 and Step 18 in Algorithm 3.1. Here, $LB(n,\ H)$ is a key bound, which is related to n and H, such that if $|p(\bar{\alpha})| < LB(n,\ H)$ then $p(\alpha) = 0$. This property guarantees that α must be a exact root of $p(x)$. For obtaining $LB(n,\ H)$ we need some lemmas below.

Lemma 3.5. *Let f be a polynomial in $\mathbf{Z}[x]$ of degree n. If $\max_{1\le i\le n}|\alpha^i - \bar{\alpha}^i| < \epsilon$, then*

$$|f(\alpha) - f(\bar{\alpha})| \le \epsilon \cdot n \cdot \mathit{height}(f). \tag{3.2}$$

Lemma 3.6 ([27], Lemma 3)**.** *Let $\alpha_1, \cdots, \alpha_q$ be algebraic numbers of exact degree of $d_1, \cdots, d_q$ respectively. Define $D = [\mathbb{Q}(\alpha_1, \cdots, \alpha_q) : \mathbb{Q}]$. Let $P \in \mathbb{Z}[x_1, \cdots, x_q]$ have degree at most N_h in x_h $(1 \le h \le q)$. If $P(\alpha_1, \cdots, \alpha_q) \ne 0$, then*

$$|P(\alpha_1, \ldots, \alpha_q)| \ge \|P\|_1^{1-D} \prod_{h=1}^{q} M(\alpha_h)^{-DN_h/d_h},$$

where $M(\alpha)$ is the Mahler measure [4] of α.

Proof. See Lemma 2 of [28]. □

Lemma 3.6 gives a lower bound on $|P(\alpha_1, \ldots, \alpha_q)|$ if $P(\alpha_1, \cdots, \alpha_q) \ne 0$ for an arbitrary multivariate polynomial $P \in \mathbb{Z}[x_1, \cdots, x_q]$. If we apply it to $g(x) = \sum_{i=0}^{m} g_i x^i \in \mathbb{Z}[x]$, then we have

[4]For any polynomial $g = \sum_{i=0}^{d} g_i x^i \in \mathbb{Z}[x]$ of degree d with the complex roots $z_1, z_2, \ldots, z_d$ we define the Mahler measure $M(g)$ by

$$M(g) = |g_d| \prod_{j=1}^{d} \max\{1, |z_j|\}.$$

The Mahler measure of an algebraic number α is defined to be the measure of its minimal polynomial.

Corollary 3.7. *Let α be an algebraic number with exact degree n_0 and $g(x) = \sum_{i=0}^{m} g_i x^i \in \mathbb{Z}[x]$. Suppose both height$(g)$ and height$(\alpha) \le H$. If $g(\alpha) \neq 0$, then*

$$|g(\alpha)| \ge (m + 1)^{-(\frac{m}{2}+n_0-1)} \cdot H^{-(m+n_0-1)}. \tag{3.3}$$

Proof. For $f(x) \in \mathbb{Z}[x]$ with degree n, we have

- Landau's inequality: $M(f) \le \|f\|_2$ (see [29], p. 154).
- $\text{height}(f) \le \|f\|_1 \le (n+1)\text{height}(f)$.
- $\text{height}(f) \le \|f\|_2 \le \sqrt{n+1}\,\text{height}(f)$.

The corollary easily follows from the facts above and Lemma 3.6. $\square$

Next we investigate how to choose ϵ to enable Algorithm 3.1 under floating-point arithmetic to return the minimal polynomial of α correctly. We denote the exact degree of α by n_0. When Algorithm 3.1 under floating-point arithmetic obtains a polynomial $p(x)$ the first time, we denote $m = i$, and hence $\deg(p) = m$. We want to decide whether $p(\alpha)$ is equal to 0 or not. Obviously, we have $m \le n_0 \le n$. From the Corollary 3.7, if $p(\alpha) \neq 0$, then we have

$$\begin{aligned} |p(\alpha)| &\ge (m + 1)^{-(\frac{m}{2}+n_0-1)} \cdot H^{-(m+n_0-1)} \\ &\ge (n + 1)^{-(\frac{3}{2}n-1)} \cdot H^{-(2n-1)} = M. \end{aligned} \tag{3.4}$$

Theorem 3.8. *Let α, $\bar{\alpha}$ be as above and $\max_{1\le i\le n} |\alpha^i - \bar{\alpha}^i| < \epsilon$. Then there exist some ϵ such that $|p(\bar{\alpha})| < \frac{M}{2}$ if and only if $p(\alpha) = 0$.*

Proof. ($\Rightarrow$) From Lemma 3.5, we have if $|p(\bar{\alpha})| < M - n\,\epsilon\,H$, then $|p(\alpha)| < M$. So we choose

$$0 < \epsilon < \frac{M}{2\,n\,H}, \tag{3.5}$$

which implies $\frac{M}{2} < M - n\,\epsilon\,H$. Thus $p(\alpha) = 0$, otherwise $|p(\alpha)| \ge M$ from equation (3.4).

($\Leftarrow$) Let $\epsilon < \frac{M}{2\,n\,H}$ and suppose $p(\bar{\alpha}) \ge \frac{M}{2}$. From Lemma 3.5, we have

$$|p(\alpha)| \ge |p(\bar{\alpha})| - n\,\epsilon\,H \ge \frac{M}{2} - n\,\epsilon\,H > 0,$$

which means $p(\alpha) \neq 0$. This contradiction completes the proof. $\square$

Remark 3.9 (Some remarks on Theorem 3.8).

1. Theorem 3.8 not only gives the value of $LB(n, H)$, which should be $M/2$, but also gives an error controlling of $\max_{1\le i\le n} |\alpha^i - \bar{\alpha}_i|$, which should be $< \frac{M}{2\,n\,H}$. Moreover, the equivalence between $g(\alpha) = 0$ and $|g(\bar{\alpha})| < M/2$ guarantees that we would never miss the correct answer. This means that if Algorithm 3.1 under floating-point arithmetic returns $p(x)$ with $\deg(p) = m$, then we have $p(\alpha) = 0$ and there must exist no polynomial $h(x)$ with degree $i < m$ such that $h(\alpha) = 0$. So the correctness of Algorithm 3.1 under floating-point arithmetic follows.
2. Of course $LB(n, H)$ could be κM, and the corresponding ϵ also could be changed to be $< \frac{(1-\kappa)M}{n\,H}$, where κ is a constant and satisfies $0 < \kappa < 1$.
3. What's more, our conclusion is also suit for real algebraic numbers. In addition the *Maple* built-in function `identify()`, which relies upon detecting integer relation algorithm, searches for an exact expression for a floating-point constant. When one wants to decide if α is algebraic by running `identify(`$\bar{\alpha}$`)`, the error tolerance is fairly arbitrary [20]. The error controlling given in Theorem 3.8 can give an answer to `identify()` in the case of algebraic nubmers.

Example 3.4 (con.). For $\alpha = 2 + \sqrt{3}\,i$ and its minimal polynomial $x^2 - 4x + 7$, let $n = 2$ and $H = 7$. Computing the error tolerance as in equation (3.5) gives $\epsilon < 294^{-2}$. From Theorem 3.8, more than $\lfloor -\log_{10} 294^{-2} \rfloor = 4$ correct decimal digits are needed to guarantee the output is correct.

From equation (3.5) we have $\log \epsilon \in \mathcal{O}(n(\log n + \log H))$. Combing with Theorem 3.3, we can give another answer of the Blum and Zhang's question without using LLL lattice reduce algorithm.

Theorem 3.10. *Let α be an algebraic number and let n and H be upper bounds of the degree and height of α respectively. Suppose we are given an approximation $\bar{\alpha}$ to α such that $\max_{1\le i\le n} |\alpha^i - \bar{\alpha}^i| < \epsilon$. Then the minimal polynomial of α can be determined in $\mathcal{O}(n^5 + n^4 \log H)$ arithmetic operations on floating-point numbers having $\mathcal{O}(n(\log n + \log H))$ bit-complexity.*

We can see that our finding minimal polynomial algorithm (Algorithm 3.1) saves a n factor either in running time or in bit-complexity compared with Theorem (1.19) of [15]. Although the same complexity as Theorem 3.10

can be achieved by using some improved lattice basis reduction algorithm, our algorithm has its own advantages. Since our algorithm is based on a generalized PSLQ algorithm, it is numerically stable. Furthermore, the necessary numbers of correct decimal digits of our algorithm is less than that of KLL.

4. Concluding Remarks

In this paper, we present a simultaneous integer relation algorithm, Algorithm 2.8, which is numerically stable. Using this algorithm we propose an finding minimal polynomial algorithm, Algorithm 3.1. We discuss the practical implementation of the two algorithms. When we analyze the floating-point arithmetical implementation of the minimal polynomial finding algorithm, we give an error tolerance such that the algorithm correctly returns the minimal polynomial of α from its approximation $\bar{\alpha}$. However, some empirical results show that the error tolerance (equation (3.5)) is not the best. So if we could get a better lower bound for $|g(\alpha)|$ when $g(\alpha) \neq 0$, the performance of our minimal polynomial finding algorithm will be improved.

We see that the minimal polynomial finding algorithm (Algorithm 3.1) can be used to factor f in $\mathbb{Z}[x]$ like this: solve an approximation root with accuracy satisfying equation (3.5), and call Algorithm 3.1 for finding its minimal polynomial which corresponds an irreducible factor of f, and then repeat the two step until f has been factored completely. Similarly with the method in [30, 31], we can also apply Algorithm 3.1 to factor multivariate polynomial with integral coefficients. All these are polynomial-time algorithms and different from traditional algorithms based on Hensel lifting.

References

[1] L. Bernstein, The Jacobi-Perron Algorithm, Its Theory and Application, Lecture Notes in Mathematics (207) (1971) 1–161.

[2] H. R. P. Ferguson, R. W. Forcade, Generalization of the Euclidean Algorithm for Real Numbers to All Dimensions Higher than Two, Bulletin of the American Mathematical Society 1 (6) (1979) 912–914.

[3] A. J. Brentjes, Multi-dimensional Continued Fraction Algorithms, Mathematisch Centrum Computational Methods in Number Theory, Pt. 2 p 287-319(see N 84-17999 08-67) .

[4] J. Hastad, B. Just, J. C. Lagarias, C. P. Schnorr, Polynomial Time Algorithms for Finding Integer Relations among Real Numbers, SIAM Journal on Computing 18 (5) (1989) 859–881.

[5] H. R. P. Ferguson, D. H. Bailey, S. Arno, Analysis of PSLQ, an Integer Relation Finding Algorithm, Math. Comput. 68 (225) (1999) 351–369.

[6] J. Hastad, B. Helfrich, J. C. Lagarias, C. P. Schnorr, Polynomial Time Algorithms for Finding Integer Relations among Real Numbers, in: STACS 86, 105–118, 1986.

[7] H. R. P. Ferguson, D. H. Bailey, Polynomial Time, Numerically Stable Integer Relation Algorithm, Tech. Rep. RNR-91-032, NAS Applied Research Branch, NASA Ames Research Center, 1992.

[8] C. Rössner, C. P. Schnorr, Diophantine Approximation of a Plane, available at http://citeseer.ist.psu.edu/193822.html, 1997.

[9] A. K. Lenstra, H. W. Lenstra, L. Lovász, Factoring Polynomials with Rational Coefficients, Math. Ann. 261 (4) (1982) 515–534.

[10] K. Unger, Helman Ferguson Profile: Carving His Own Unique Niche, in Symbols and Stone, Science 314 (5798) (2006) 412–413.

[11] D. H. Bailey, J. M. Borwein, N. J. Calkin, R. Girgensohn, D. R. Luke, V. H. Moll, Experimental Mathematics in Action, AK Peters, 2007.

[12] D. H. Bailey, D. J. Broadhurst, Parallel Integer Relation Detection: Techniques and Applications, Math. Comput. 70 (236) (2001) 1719 – 1736.

[13] D. H. Bailey, J. Borwein, PSLQ: An Algorithm to Discover Integer Relations, available at http://crd.lbl.gov/~dhbailey/dhbpapers/pslq-comp-alg.pdf, 2009.

[14] R. Kannan, A. K. Lenstra, L. Lovász, Polynomial Factorization and Nonrandomness of Bits of Algebraic and Some Transcendental Numbers, in: STOC '84: Proceedings of the sixteenth annual ACM symposium on Theory of computing, 191–200, 1984.

[15] R. Kannan, A. K. Lenstra, L. Lovász, Polynomial Factorization and Nonrandomness of Bits of Algebraic and Some Transcendental Numbers, Math. Comput. 50 (181) (1988) 235–250.

[16] B. Just, Integer Relations among Algebraic Numbers, in: Mathematical Foundations of Computer Science 1989, 314–320, 1989.

[17] I. Morel, D. Stehlé, G. Villad, H-LLL: Using Householder Within LLL, in: ISSAC '09, Seoul, Korea, 2009 (to appear).

[18] J. Borwein, R. Corless, Emerging Tools for Experimental Mathematics, American Mathematical Monthly (1999) 889–909.

[19] J. M. Borwein, P. Lisonek, Applications of Integer Relation Algorithms, Discrete Mathematics (Special issue for FPSAC 1997) 217 (2000) 65–82.

[20] P. Borwein, K. G. Hare, A. Meichsner, Reverse Symbolic Computations, the *identify* Function, in: Proceedings from the Maple Summer Workshop, Maple Software, Waterloo, 2002.

[21] D. Bailey, J. Borwein, V. Kapoor, E. Weisstein, Ten Problems in Experimental Mathematics, American Mathematical Monthly 113 (6) (2006) 481–509.

[22] X.-l. Qin, Y. Feng, J.-w. Chen, J.-z. Zhang, Finding Exact Minimal Polynomial by Approximations, in: SNC'09, ACM, Kyoto, Japan, 2009 (to appear).

[23] H. R. P. Ferguson, R. W. Forcade, Multidimensional Euclidean algorithms, (Crelle's) Journal für die reine und angewandte Mathematik 334 (1982) 171–181.

[24] P. Borwein, Computational Excursions in Analysis and Number Theory, Springer, New York, 2002.

[25] D. H. Bailey, Integer Relation Detection, Computing in Science and Engineering 2 (1) (2000) 24–28, ISSN 1521-9615.

[26] D. H. Bailey, Y. Hida, X. S. Li, B. Thompson, ARPREC: An Arbitrary Precision Computation Package, available at http://crd.lbl.gov/~dhbailey/dhbpapers/arprec.pdf, 2002.

[27] M. Mignotte, M. Waldschmidt, Linear Forms in Two Logarithms and Schneider's Method, Math. Ann. 231 (1978) 241–267.

[28] N. I. Fel'dman, Estimate for Linear Form of Logarithms of Algebraic numbers, Sbornik: Mathematics 5 (2) (1968) 291–307.

[29] J. von zur Gathen, J. Gerhard, Modern Computer Algebra, Cambridge University Press, London, 1999.

[30] M.-P. van der Hulst, A. Lenstra, Factorization of Polynomials by Transcendental Evaluation, in: EUROCAL '85, 138–145, 1985.

[31] J.-w. Chen, Y. Feng, X.-l. Qin, J.-z. Zhang, Exact Polynomial Factorization by Approximate High Degree Algebraic Numbers, in: SNC'09, ACM, Kyoto, Japan, 2009 (to appear).