Incremental PSLQ with Application to Algebraic Number Reconstruction *

Yong Feng, Jingwei Chen, Wenyuan Wu
Automated Reasoning and Cognition Key Laboratory of Chongqing, CIGIT, CAS

{yongfeng, chenjingwei, wuwenyuan}@cigit.ac.cn

1. Introduction. A vector $\boldsymbol{m} = (m_1, \ldots, m_n) \in \mathbb{Z}^n \setminus \{\mathbf{0}\}$ is called an *integer relation* for $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ if $\sum_{i=1}^{n} m_i x_i = 0$. In literature, the HJLS algorithm [3, Sec. 3] and the PSLQ algorithm [2] solve the problem of finding integer relation polynomially. Although it has been theoretically proved that PSLQ is to some extent equivalent to HJLS, under the exact real arithmetic computational model (see, e.g., [7, 1]), the PSLQ algorithm seems more practical.

The problem of finding the *minimal polynomial* from an approximation $\overline{\alpha}$ of a d_0 degree *algebraic number* α, equivalent to finding an integer relation for the vector $(1, \alpha, \ldots, \alpha^{d_0})$, was first solved in [5] by using the celebrated LLL algorithm [6]. This routine has been recently improved in [4]. Naturally, the PSLQ algorithm applies to the algebraic number reconstruction problem as well [8].

Given an approximation to an unknown algebraic number α, a degree bound d and an upper bound M on its height, if the exact degree d_0 of α is also unknown, then no matter whether one uses PSLQ or LLL, one has to search an integer relation for the vector $(1, \alpha, \ldots, \alpha^i)$ from $i = 2, 3, \ldots$ until d_0 $(\leq d)$. Hence, if the complexity of a polynomial algorithm for finding an integer relation is $\mathcal{O}(P(n, M))$ for an n-dimensional vector, then the complexity of the minimal polynomial algorithm, based on the integer relation finding algorithm, is $\mathcal{O}(d_0 \cdot P(d_0, M))$. Our main contribution in the present work is to give an *incremental* version of PSLQ, which leads to an efficient algebraic number reconstruction algorithm with complexity only $\mathcal{O}(P(d_0, M))$, even though the exact degree of the algebraic number is unknown.

Algorithm 1 (IPSLQ).

```
Input: A vector $\boldsymbol{x} = (x_1, \cdots, x_n) \in \mathbb{R}^n$ with $x_i \neq 0$ for $i = 1, \ldots, n$ and a positive number $M$.
Output: Either return an integer relation for $\boldsymbol{x}$, or return "no relation with length smaller than $M$".
1. Construct $H_x \in \mathbb{R}^{n \times (n-1)}$. Set $H := H_x$, $A := I_n$ and $B := I_n$. Size-reduce $H$ and update $A$ and $B$.
2. For $k$ from $n-1$ to 1 do
    (a) While $h_{n-1,n-1} \neq 0$ do
          i. Choose $r$ such that $\gamma^r |h_{r,r}| = \max_{j \in \{k,\cdots,n-1\}} \{\gamma^j |h_{j,j}|\}$.
         ii. Swap the $r$-th and the $(r+1)$-th rows of $H$ and update $A$ and $B$.
        iii. If $r < n-1$ then update $H$ to L-factor of $H$.
         iv. Size-reduce $H$ and update $A$ and $B$.
          v. If $\max_{j \in \{k,\cdots,n-1\}} |h_{j,j}| < 1/M$ then do the following: If $k > 1$ then go to Step 2; Else return "no relation with length
             smaller than $M$".
    (b) Return the last column of $B$.
```

2. The Incremental PSLQ Algorithm. The main difference between IPSLQ (Algorithm 1) and PSLQ is that PSLQ considers $x_1, \ldots, x_n$ directly, while IPSLQ considers $x_i, \ldots, x_n$ gradually, i.e., if the vector $(x_i, \ldots, x_n)$ has no relation with 2-norm less than M then add x_{i-1} to the left; see Step 2(a)v.

The application of IPSLQ to efficient reconstructing minimal polynomial depends on the following two key points: (1) We use $(x_1, \ldots, x_n) = (\alpha^{n-1}, \ldots, \alpha, 1)$, which is the reverse order of the traditional version, to construct the matrix H_x (see [2, Def. 2] for the construction). (2) The important observation is that

*This work was partially supported by NKBRPC (2011CB302400) and NSFC (11001040, 11171053, 91118001).

the matrix H_x for $(x_i, \ldots, x_n)$ is exactly the right-bottom most submatrix of H_x for $(x_{i-1}, x_i, \ldots, x_n)$. Thus, the results produced by the previous iterations are still valid for the new matrix H. However, the traditional methods can not reuse those previous information. Therefore, the complexity of IPSLQ for minimal polynomial without knowing degree is only $\mathcal{O}(P(d_0, M))$, which is the same as PSLQ for minimal polynomial with knowing degree.

3. Experiments. The following experiments are preliminary and to compare the performance between traditional PSLQ and IPSLQ for minimal polynomial reconstruction. Consider approximations of $\alpha = 3^{1/s} + 2^{1/t}$ with 500 decimal digits. Running these experiments in Maple 15 with `Digits :=500` gives a preliminary experimental results in Table 1. Note that here `Digits :=500` may not be necessary (see [8] for the a detailed error control). In Table 1, the input degree bound and height bound in these tests are d and $M+1$; the exact degree and height of α are $d-1$ and M, respectively. All these experimental results are obtained by using a Windows 7 (32 bits mode) PC with AMD Athlon II X4 645 processor (3.10 GHz) and 4 GB memory. Note that there exists a built-in function `IntegerRelations:-PSLQ` in Maple 15, but for the comparison in Table 1, we implement the PSLQ algorithm by ourselves. The reasons we do not use the built-in function is that there does not exist a height parameter in the built-in function. This may cause that the built-in function will go on the iterations even if the height has been greater than M. In our implementations of PSLQ and IPSLQ, the same function uses the same technique for fairness. According to Table 1, the IPSLQ algorithm is faster than the PSLQ algorithm. Meanwhile the ratio between T_{PSLQ} and T_{IPSLQ} seems to get larger and larger with increasing d, but always smaller than d.

No.	s	t	d	M	T_{IPSLQ}	T_{PSLQ}	$\frac{T_{PSLQ}}{T_{IPSLQ}}$
1	2	2	5	10	0.08	0.16	2.00
2	2	3	7	36	0.16	0.64	4.00
3	3	3	10	125	0.89	5.34	6.00
4	3	4	13	540	3.14	21.34	6.79
5	2	7	15	5103	6.91	45.91	6.64
6	3	6	19	10278	23.37	144.11	6.17
7	4	5	21	11160	32.73	249.54	7.62
8	5	5	26	57500	78.95	838.99	10.63
9	5	6	31	538380	186.28	2089.87	11.22
10	6	6	37	4281690	421.94	4313.99	10.22

Table 1: IPSLQ VS PSLQ for minimal polynomial

References

[1] J. Chen, D. Stehlé, and G. Villard. A new view on HJLS and PSLQ: Sums and projections of lattices. In *Proc. ISSAC '13*, Boston, USA, 2013. To appear.

[2] H. R. P. Ferguson, D. H. Bailey, and S. Arno. Analysis of PSLQ, an integer relation finding algorithm. *Math. Comput.*, 68(225):351–369, 1999.

[3] J. Håstad, B. Just, J. Lagarias, and C. Schnorr. Polynomial time algorithms for finding integer relations among real numbers. *SIAM J. Comput.*, 18(5):859–881, 1989.

[4] M. van Hoeij and A. Novocin. Gradual sub-lattice reduction and a new complexity for factoring polynomials. *Algorithmica*, 63(3):616–633, 2012.

[5] R. Kannan, A. Lenstra, and L. Lovász. Polynomial factorization and nonrandomness of bits of algebraic and some transcendental numbers. *Math. Comput.*, 50(181):235–250, 1988.

[6] A. Lenstra, H. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982.

[7] A. Meichsner. Integer Relation Algorithms and the Recognition of Numerical Constants. Master's thesis, Simon Fraser University, 2001.

[8] X. Qin, Y. Feng, J. Chen, and J. Zhang. A complete algorithm to find exact minimal polynomial by approximations. *Int. J. Comput. Math.*, 89(17):2333–2344, 2012.