Applications of Integer Relation Algorithms

Jonathan M. Borwein
and
Petr Lisoněk

Centre for Experimental and Constructive Mathematics
Department of Mathematics and Statistics
Simon Fraser University
Burnaby, BC
Canada V5A 1S6

e-mail: `jborwein@cecm.sfu.ca`, `lisonek@cecm.sfu.ca`

November 18, 1997

Submitted to the Proceedings of the 9th International Conference on Formal Power Series and Algebraic Combinatorics (FPSAC'97), July 1997, Vienna, Austria, to appear as a special issue of Discrete Mathematics.

Abstract

Let a be a vector of real numbers. By an integer relation for a we mean a non-zero integer vector c such that $ca^T = 0$. We discuss the algorithms for finding such integer relations from the user's point of view, by presenting examples of their applications and by reviewing the available software implementations.

1 Introduction

Integer relation algorithms are core tools for computer-assisted mathematics. Their main purpose is the "reverse engineering" of real numbers in the sense that they help to recover a formula that evaluates to a given real number.

This paper is written from our perspective as frequent users of integer relation algorithms, mainly by presenting research that we have done recently. The background of the algorithms is described only to the extent to which average users would like to be able to modify integer relation code available to them. Those who want to learn more about the theory of these algorithms, including all proofs and complexity analyses, are referred to the excellent survey [23].

Two types of integer relation algorithms and certain general applications of them are introduced in Section 2. Examples of more specific applications are discussed in detail in Section 3. Implementations available to us at the present time are briefly surveyed in Section 4.

1.1 Integer Relations

Let $a \in \mathbb{R}^n$ be a given vector. We say that the vector $c \in \mathbb{Z}^n$ is an *integer relation* for a if

$$\sum_{i=1}^{n} c_i a_i = 0 \tag{1}$$

with at least one c_i different from zero. It is the purpose of *integer relation algorithms* to search for a non-zero vector c satisfying (1), possibly with a certain level of confidence.

Of course, a numerical discovery of a relation by a computer does not in general constitute a proof of this relation; one of the reasons being that the computer operates on rational approximations of numbers that in many applications are likely to be transcendental. In many cases, however, the relations we first discovered numerically subsequently received rigorous mathematical proofs. Moreover, many complicated relations probably would never have been dreamed of without the assistance of the computer. In Section 3 we show examples of such phenomena.

1.2 An Example

As an introductory example, let us consider the definite integral

$$V = \int_0^\infty \frac{\sqrt{x}\,(\ln x)^5}{(1-x)^5}dx. \tag{2}$$

Some computer algebra systems are able to evaluate V in a closed form while others are not. Most of these systems will nevertheless yield a numerical approximation to V to a few dozens of decimal digits very quickly; this value is

$$V \approx -16.699473719229070496187243400 7\cdots$$

to the precision of 30 decimal digits.

Suppose that we knew in advance that the closed form evaluation of V was a rational linear combination of even powers of π. If moreover we knew that K is the maximal power of π that possibly can enter the expression for V, we could consider the integer relation problem (1) with $a_1 = V$, $a_2, a_3, \ldots, a_n = 1, \pi^2, \ldots, \pi^{2(n-2)}$, where the value of n is determined by $2(n-2) = K$.

In this "toy" example we can use any available implementation of the integer relation algorithms to obtain a solution. For example, using an LLL-based algorithm implemented in Maple, an initial trial for $K = 10$ (thus $n = 7$) with all numerics performed at the precision of 30 decimal places gives within two seconds an answer

$$(c_1, c_2, c_3, c_4, c_5, c_6, c_7) = (24, 0, 120, 140, -15, 0, 0)$$

or in other words

$$V = \frac{5}{24}\pi^2(3\pi^4 - 28\pi^2 - 24). \tag{3}$$

We must have some reasonable idea about the form of the result sought if we hope to keep an integer relation search within reasonable limits (Section 1.3). In the case of the integral V, its closed form follows from the theory presented in [24], which contains a general symbolic algorithm for a family of definite integrals of the type (2). Indeed the example (4.8) of [24] is exactly our equation (3).

Similarly, multiplicative relations can be sought by taking logarithms and including logarithms of small primes, from which an eventual rational coefficient can be composed. Such ideas are discussed in detail by Bailey and Plouffe in [5].

1.3 Limits of the Search

Two limitations on the model of a relation that we may pursue are evident: (i) the length of the vector a and (ii) the precision to which the entries of the vector a are known. As a rule of thumb, if a has n entries and D is the maximal number of digits among the entries of c, then (as an easy counting argument shows) we have to know all entries of the vector a to the precision somewhat greater than nD digits and computations have to be performed using at least this level of numeric precision, if we hope to be able to distinguish between bogus and significant relations c.

Since different integer relation algorithms (and their implementations) can differ by as much as two orders of magnitude in their running times on the same input, there are no globally valid limitations on n and D. In Section 4 we include timings and the required precision for an example with $n = 31$, $D = 6$ and running times ranging between dozens of seconds and dozens of minutes for various implementations. The largest example that we have run was with an input vector a of length 171, given with a precision of 2000 decimal places; a total of 56 linearly independent integer relations (with few digit coefficients) were recovered in 7 days of CPU time on a DEC Alpha station (Section 3.2).

2 Integer Relation Algorithms

We recommend [23] for a comprehensive survey of integer relation algorithms. That paper also presents a unifying view of several types of algorithms that at the first glance seem to be different in nature. Since our paper is application oriented, we restrict our brief presentation to the two types of algorithms for which implementations are accessible to us (and which then are surveyed in Section 4).

Since an implementation of the LLL (Lenstra, Lenstra, Lovász) algorithm is available in almost any computer algebra system, we discuss the LLL-based methods in more detail (Section 2.1) because it is conceivable that the reader would like to play with the respective integer relation code. In the case of the PSLQ algorithm, whose implementations are much more efficient but less directly available, we restrict ourselves to stating its most important features (Section 2.2) and we refer the reader to appropriate sources for a more technical discussion.

2.1 Lattice Basis Reduction Methods

By the *length* of the vector $v \in \mathbb{Q}^n$ we mean its Euclidean length, that is, $|v| = \left(\sum_{i=1}^n v_i^2\right)^{1/2}$.

Suppose we have a set of k vectors $B = \{v^{(1)}, \ldots, v^{(k)}\} \subset \mathbb{Q}^n$, linearly independent over $\mathbb{Q}$. By the *lattice* spanned by B we mean the set

$$L = \left\{ \sum_{i=1}^{k} r_i v^{(i)} \;\middle|\; r_i \in \mathbb{Z} \right\} \subset \mathbb{Q}^n.$$

This lattice has *dimension* n and *rank* k. The set B is called a *basis* of the lattice. The basis of course is not unique, and given a lattice, we may look for bases with some distinguished properties.

The problem of finding the shortest vector in a given lattice L is believed to be NP-complete [23]. There are, however, approximate solution algorithms for this problem that run in polynomial time, such as the famous LLL lattice basis reduction algorithm [25] which, given a basis for a lattice L, finds a *reduced basis* for L that is entirely composed of vectors with length at most $2^{n-1} \cdot l$, where l is the length of the shortest vector in L and n is the dimension of L.

Lattice basis reduction algorithms can be employed to search for integer relations in the following way ([25], p. 535). Suppose that we are given a vector of real numbers $(a_1, \ldots, a_n)$ and let $(a'_1, \ldots, a'_n)$ be a vector of their rational approximations. Let us consider the lattice L spanned by the $(n+1)$-dimensional vectors $v^{(i)}$ $(1 \le i \le n)$ where $v^{(i)}_j = \delta_{i,j}$ (the Kronecker delta) for $1 \le i, j \le n$ and $v^{(i)}_{n+1} = C \cdot a'_i$ for $1 \le i \le n$, where C is a large constant. Now, if w is a short vector in a reduced basis for L,

$$w = (w_1, \ldots, w_n, C \sum_{i=1}^{n} w_i a'_i),$$

then $|\sum_{i=1}^n w_i a'_i|$ is small and thus $(w_1, \ldots, w_n)$ is a good candidate for an integer relation for the vector $(a_1, \ldots, a_n)$. Of course, with increasing C we improve our chances of correctly determining which vectors in the reduced basis correspond to an integer relation for $(a_1, \ldots, a_n)$, and which do not.

A technical detail is worth mentioning: Many software systems have LLL-based routines that search for minimal polynomials (Section 2.3) but correspondingly lack a general integer relation routine. It was the purpose of this section to give the reader sufficient insight in the LLL-based integer relation search so that (s)he can adapt the minimal polynomial routine as a general integer relation routine.

2.1.1 Rational Inputs

In the special case when the vector $a = (a_1, \ldots, a_n)$ consists entirely of rational numbers and we are taking $a_i = a_i'$ for all i, we of course know in advance that the dimension of the lattice of integer relations for a is equal to $n-1$ (assuming a non-zero). If C is taken sufficiently large, then upon completion of the lattice reduction algorithm, all numbers $|\sum_{i=1}^n w_i a_i'| = |\sum_{i=1}^n w_i a_i|$ will be equal to 0 for *any* vector w in the reduced basis, since $w_1, w_2, \ldots, w_n$ are integers and, by the definition of a reduced basis,

$$(2^n \cdot l)^2 \geq \sum_{i=1}^{n+1} w_i^2 \geq w_{n+1}^2 = C^2 \cdot \left(\sum_{i=1}^{n} w_i a_i' \right)^2$$

where l is the length of the shortest vector in the lattice under consideration.

In Section 3.3 we present an application of finding integer relations for integer input vectors.

2.2 The PSLQ Algorithm

The first integer relation algorithm working reliably on vectors of non-trivial length was discovered by Ferguson and Forcade in 1977. After a number of improvements, in 1991 a new algorithm, known as the PSLQ algorithm, was developed by Ferguson [19]. More recently a much simpler formulation of this algorithm was introduced [20]. The papers [19] and [20] are available on David Bailey's WWW page at
`http://science.nas.nasa.gov/Groups/AAA/db.webpage/dbailey.html`

Because this algorithm employs a numerically stable matrix reduction procedure, it is free from the numerical difficulties that plague other integer relation algorithms. Furthermore, its stability allows for an efficient implementation with lower run times on average than other algorithms currently in use.

A distinguishing feature of the PSLQ algorithm is that, besides recovering an integer relation (if one exists), it can also produce a *negative* (non-existence) result of the form "if a relation exists, then its Euclidean norm is greater than M (an explicit numerical bound)". One can show, on using a working precision that is only slightly higher than that of the input data, that these bound results obtained from computer runs are reliable. Results of this type, supporting conjectures about transcendence of certain constants, are to be found in [2, 4]. (See also Section 2.3 on minimal polynomials.) For example, Bailey and Ferguson [4] used PSLQ to establish, that if Euler's constant γ satisfies an integer polynomial of degree 50 or less, then the Euclidean norm of the coefficients must exceed $7 \cdot 10^{17}$.

In the context of these exclusion bounds it is important to realize that, even for very "similar" numbers, the Euclidean norms of their relations can differ greatly. For example, if x satisfies an integer polynomial of degree d, norm s and k is an integer, then kx satisfies an integer polynomial whose norm may become as high as roughly $|k|^d s$. The lesson that we learn here is that, if we can, we should "normalize" our input so that the resulting integer relation is of low Euclidean norm.

2.3 Finding Minimal Polynomials

If α is a real number, then by definition α is algebraic exactly if, for some r, the vector

$$(1, \alpha, \alpha^2, \ldots, \alpha^r) \tag{4}$$

has an integer relation. The integer coefficient polynomial of lowest degree, whose root α is, is determined uniquely up to a constant multiple; it is called the *minimal polynomial* for α. Integer relation algorithms can be employed to look for minimal polynomials in a straightforward way by simply feeding them the vector (4) as their input. Some computer algebra systems have an LLL-based procedure for finding minimal polynomials but lack a general procedure for finding integral relations; based on the explanation in Section 2.1 the reader should be able to convert those minimal polynomial routines into general integral relation routines.

2.4 Basis Perturbation

Suppose that we want to find *many* short vectors in a given lattice L. One situation where we may wish to do this is the case when we are given a vector a that possesses more than one integer relation, and the lattice L is formed by integral vectors c such that $ca^T = 0$. Short (simple) vectors from L are usually easier to understand, hence we might wish to have a larger collection of them.

Let $SL(n,\mathbb{Z})$ be the special linear group of the $n \times n$ matrices over $\mathbb{Z}$ with determinant equal to 1, where n is the dimension of the lattice L. If B is a basis for L, then for any $M \in SL(n,\mathbb{Z})$, the image $M \cdot B$ is also a basis for L. By running the basis reduction algorithm on many different bases of L we increase our chances of finding more short vectors of L. A method for generating the matrices from $SL(n,\mathbb{Z})$ with some degree of randomness is needed here; in our computations we have chosen to use long products of matrices of the form of the $n \times n$ identity matrix with an $SL(2,\mathbb{Z})$ submatrix inserted at the intersections of the i-th and j-th row and column, for two randomly chosen $i, j \in \{1, 2, \ldots, n\}$.

3 Applications

In Sections 3.1 through 3.3 we discuss in some detail examples in which we have successfully applied integer relation algorithms. More applications are then briefly listed (with appropriate references) in Section 3.4.

3.1 Cubic Singular Values

An interesting test case for the application of integer relation methods to the task of finding minimal polynomials as in Section 2.3 arises in the setting of modular or theta functions. [9], [18]. Consider

$$a := a(q) = \sum_{m,n=-\infty}^{\infty} q^{m^2+mn+n^2}, \qquad b := b(q) = \sum_{m,n=-\infty}^{\infty} \omega^{n-m} q^{n^2+mn+m^2},$$

where $\omega = e^{2\pi i/3}$ and

$$c := c(q) = \sum_{m,n=-\infty}^{\infty} q^{(n+1/3)^2+(n+1/3)(m+1/3)+(m+1/3)^2}.$$

These three functions lie on the Fermat curve $a^3 = b^3 + c^3$ and one has the lovely parameterization of a hypergeometric function:

$${}_2F_1(\frac{1}{3}, \frac{2}{3}; 1; \frac{c^3}{a^3}) = a.$$

If we set $q := e^{-2\pi\sqrt{N/3}}$ it follows from the theory in [9], [18] that, for N rational, $s_N := \frac{c}{a}$ is an algebraic number (expressible by radicals). For a positive integer N, s_N is called the N-th *cubic singular value*—in analogy to the classical case of the complete elliptic integral of the first kind and quadratic singular values k_N ([7], p. 139). While with more work it is possible to be more explicit about the exact form of s_N, it is interesting to ask what may be determined entirely by our computational methods.

In terms of the classical *theta functions*

$$\theta_3(q) := \sum_{n=-\infty}^{\infty} q^{n^2} \qquad \theta_2(q) := \sum_{n=-\infty}^{\infty} q^{(n+\frac{1}{2})^2}$$

one has the following highly lacunary representations for a, b and c

$$a(q) = \theta_3(q)\theta_3(q^3) + \theta_2(q)\theta_2(q^3)$$

$$b(q) = (3a(q^3) - a(q))/2 \qquad c(q) = (a(q^{1/3}) - a(q))/2$$

which allows one to compute many hundreds or thousands of digits of s_N almost instantly.

This then provides a rather good test of the ability of a program to determine the algebraic value at hand. For example:

$$\begin{aligned}
s_1 &= 1/\sqrt[3]{2} \\
s_2 &= 0.52710011025237060710\cdots = (1/2 - 1/4\sqrt{2})^{1/3} \\
s_3 &= 0.36602540378443864697\cdots = (\sqrt{3} - 1)/2 \\
s_4 &= 0.26625264629019611364\cdots = (1/2 - 5/18\sqrt{3})^{1/3}.
\end{aligned}$$

From the first few cases, and in analogy with the classical (quadratic) case[1], we were led to look instead at

$$G_N := (1/2 - s_N^3)^2 \qquad \text{or} \qquad g_N := \frac{3s_N}{1 - s_N^3} \tag{5}$$

for $N \equiv \pm 1$ or $N \equiv 0$ mod 3 respectively. This has the effect of reducing the degree of the polynomial sought significantly and so made it practical—with refinements to PSLQ—to obtain the first 100 values of s_N. Thus, we obtain $s_{14} = (1/2 - 1/250\sqrt{14} - 99/500\sqrt{6})^{1/3}$. In addition, suppose we compute

[1] Where one distinguishes $4k^2(1 - k^2)$ or $(1 - k^2)/2k$ for N respectively odd or even.

g_{14} to 100 places and then obtain the potential polynomial from, say, a 50 digit computation. If the putative root is then verified to near full precision we are almost certainly right.

To extract the radical representation of s_N for a given N, we compute P_N, the minimal polynomial for G_N or g_N (depending on N mod 3). Then we try factoring P_N over different quadratic number fields until we get a factor which is (essentially) of degree 4 or less, solve it in radicals and use a transformation inverse to (5) to get s_N in radical form. The least value of N for which this approach does not seem to work is $N = 53$. Computing G_{53} to 200 places and using 100 digits in Maple's 'minpoly', we obtain

$$P_{53}(x) := -1468655558161 - 9100986507260x + 23965205326688x^2$$
$$+672242305831040x^3 - 3033916937098496x^4 + 3679773552653312x^5$$

as a polynomial we hope satisfies $P_{53}(G_{53}) = 0$. We also verify that $P_{53}(G_{53}) = -1.9 \cdot 10^{-185}$. Extra comfort is provided by 'galois($P_{53}(x)$)' which returns

$$+D5, 10, (25)(34), (12345)$$

telling us that the polynomial indeed has a solvable Galois group of order 10. Now, Daniel Lazard has written Maple code which returns a radical for any solvable quintic. The procedure, which works well on tame examples, returns a radical with 7508 symbols. Kevin Hare at CECM has greatly simplified the radical expression to one with only 860 symbols which, as Maple can now happily verify symbolically, solves P_{53} and which ultimately will become something attractive.

Most reassuringly, our evaluations of s_N can often be verified via the equation

$$J_2(k_{3N}) = J_3(s_N) \tag{6}$$

which holds for any rational N. Here, k and s are as above, while

$$J_2(x) = \frac{4}{27}\frac{(1-x^2(1-x^2))^3}{x^4(1-x^2)^2}$$

is Klein's *absolute invariant* ([7], p. 115) and

$$J_3(x) = \frac{1}{64}\frac{(9-8x^3)^3}{x^9(1-x^3)}$$

its cubic counterpart[2].

The identity (6) follows from Proposition 5.8 in [7], p. 185 (equation (5.5.26)) and [8]. Using known values of k_{3N} (see [7], pp. 139, 162 and many other places) the corresponding s_N can be verified. Maple easily checks the identity (6) symbolically using its 'radnormal' procedure for $N \leq 10$; for larger N the complexity of intermediate results becomes a limiting factor and the computation requires human guidance. Still, after adequate effort we were able to verify our value of s_{70} symbolically, using the evaluation of k_{210} (see [7], p. 141) that Hardy celebrated as "one of the most striking of Ramanujan's results." (See [22], p. 229.)

It is interesting to note that the s_N values can be also verified by a mixture of symbolic and numerical computing. For example, using a Liouville type bound on the minimal polynomials for k_{210} and our minpoly-guessed value of s_{70} (let us call it S), we can prove that either (i) $J_2(k_{210}) = J_3(S)$ or (ii) $|J_2(k_{210}) - J_3(S)| > 10^{-6400}$. It takes just 9 minutes of CPU time on an SGI workstation to prove numerically that (ii) cannot hold, again using Maple. Thus, $G_{70} =$

$$\frac{735540276593}{3996969003000} + \frac{3413048639}{222053833500}\sqrt{2}\sqrt{3}$$
$$+\frac{357407303}{55513458375}\sqrt{3}\sqrt{5}\sqrt{7} - \frac{8992317139}{1998484501500}\sqrt{2}\sqrt{5}\sqrt{7}.$$

We have found similar proofs and bounds for other larger values of N, such as 110 and 154.

3.2 Euler Sums

Euler sums (also called "multiple zeta functions" or "multiple harmonic sums") are a useful generalization of the classical zeta function: For positive integers $i_1, i_2, \ldots, i_k$ we define

$$\zeta(i_1, i_2, \ldots, i_k) := \sum_{n_1 > n_2 > \ldots > n_k > 0} \frac{1}{n_1^{i_1} n_2^{i_2} \cdots n_k^{i_k}} \tag{7}$$

and we note that $i_1 > 1$ is both necessary and sufficient condition for convergence. Thence we get the mapping $\zeta : (\mathbb{N}^+)^k \to \mathbb{R} \cup \{\infty\}$. The integer

[2] We note that $J_3(x) = f\left(2(1-x^3)^{1/3}\right)$ where f is the Belyi function of the tetrahedron, see [26], p. 376.

k is called the *depth* of the sum, and $i_1 + i_2 + \cdots + i_k$ is called the *weight* of the sum. The definition (7) can be extended to alternating sums. Euler sums recently found very interesting interpretations in high energy physics and knot theory.

Euler sums satisfy many striking identities, of which $\zeta(2,1) = \zeta(3)$ is the simplest one. (Let us remark that Euler himself suggested and partially proved theorems about reducibility of depth 2 zetas to depth 1 zetas.) Recent work on integral representations of Euler sums has led us to a fast algorithm for their high precision evaluations [13], thus allowing us to thoroughly examine them using integer relation algorithms.

The example that we want to single out for presentation is motivated by the identity conjectured by Zagier [27] and proved first by Broadhurst [13], after extensive empirical work in [12]:

$$\zeta(\{3,1\}_n) = \frac{1}{2n+1}\zeta(\{2\}_{2n}) \quad (= \frac{2\pi^{4n}}{(4n+2)!})$$

where $\{s\}_n$ denotes the string s repeated n times. Let us mention in this context that, in all known "non-decomposable" identities involving Euler sums, all ζ-terms have the same weight. This fact of course is of great importance for guided integer relation searches—as it dramatically reduces the size of the search space.

Broadhurst and Lisoněk recently used Bailey's fast implementation of the PSLQ algorithm (Section 4.3) to search for possible generalizations of Zagier's identity. Soon it appeared that the values

$$Z(m_1, m_2, \ldots, m_{2n+1}) := \zeta(\{2\}_{m_1}, 3, \{2\}_{m_2}, 1, \{2\}_{m_3}, 3, \ldots, 1, \{2\}_{m_{2n+1}})$$

participate in many such identities. We then started to form the PSLQ input vectors from all Z values of fixed weight ($2K$, say) along with the value $\zeta(\{2\}_K)$; and were rewarded by detecting many identities, from which we were able to infer general patterns, and conjecture (among many other things) the following generalization of Zagier's identity [14]:

$$\sum_{i=0}^{2n} Z(C^i S) = \zeta(\{2\}_{M+2n}) \tag{8}$$

where S is a string of $2n+1$ non-negative integers whose sum is M, and $C^i S$ denotes the cyclic shift of S by i positions. Zagier's identity is a special case of (8) with all entries of S being zeros.

The symmetric form of (8) highlighted that Zagier-type identities have a lot of combinatorial content, and in the cases $M = 0, 1$ we were able to reduce (8) to evaluation of certain combinatorial sums, yielding in this way combinatorial proofs [14] whose only analytic component is the classically known representation of Euler sums as iterated integrals. In the cases $M \geq 2$ we do not have such proofs yet, but we have strong numerical evidence supporting (8) in these cases also.

The (now partially proved) conjecture (8) would never have been identified without extensive use of an integer relation algorithm. Not only did we infer new identities as results of sophisticated PSLQ runs, but we even received a different perspective on the entire situation and new avenues and proof styles opened up for us.

3.3 The Prouhet-Tarry-Escott Problem

The *Prouhet-Tarry-Escott Problem* (PTE Problem) is an old unsolved problem in number theory. Extensive accounts on it are available in [21] or [15].

In its most general setting the PTE Problem asks for two multisets of integers (or, equivalently, of rational numbers) $X = \{x_1, \ldots, x_s\}$ and $Y = \{y_1, \ldots, y_s\}$ such that

$$\sum_{i=1}^{s} x_i^e = \sum_{i=1}^{s} y_i^e \qquad e = 1, 2, \ldots, s-1. \tag{9}$$

Of course the system (9) is satisfied if X and Y are equal as multisets; such solutions we call *trivial.* Also, if X, Y are multisets satisfying the system (9) and $f : t \mapsto at+b$ is a linear transformation with rational coefficients, then the multisets $f(X), f(Y)$ satisfy (9). We then say that X, Y and $f(X), f(Y)$ are *equivalent* solutions. In what follows we consider only non-trivial solutions up to equivalence.

Solutions to the PTE Problem are known only for $s = 1, \ldots, 10$, and in all these cases (except $s = 9$) it is known that there are infinitely many non-equivalent solutions.

Due to its large underdeterminacy as a polynomial equation system, the PTE Problem is often considered in its more restrictive, *symmetric version.* For s odd, the symmetric version takes the form

$$\sum_{i=1}^{s} z_i^e = 0 \qquad e = 1, 3, \ldots, s-2. \tag{10}$$

Clearly, if $z_1, \ldots, z_s$ satisfy (10), then $x_1 = z_1, \ldots, x_s = z_s$ and $y_1 = -z_1, \ldots, y_s = -z_s$ is a solution of (9). Such PTE solutions we call *odd symmetric solutions* [15].

In the case $s = 9$, only two solutions to the PTE Problem are known, and they are both odd symmetric solutions. They are, up to equivalence, generated by

$$\{z_1, \ldots, z_9\} = \{-98, -82, -58, -34, 13, 16, 69, 75, 99\} \tag{11}$$

and

$$\{z_1, \ldots, z_9\} = \{-169, -161, -119, -63, 8, 50, 132, 148, 174\}. \tag{12}$$

Since the entries of (11) and (12) are small integers, we of course expect that there exist many relations for them with small integer coefficients. There is, however, one type of relations whose frequency is striking in the output of an integer relation algorithm: it is

$$z_{i_1} + z_{i_2} + z_{i_3} + z_{i_4} = z_{j_1} + z_{j_2} + z_{j_3} + z_{j_4} + z_{j_5} = 0 \tag{13}$$

where $\{i_1, i_2, i_3, i_4\} \cup \{j_1, j_2, j_3, j_4, j_5\}$ is a disjoint partition of $\{1, 2, \ldots, 9\}$. (Recall from (10) that $\sum_{i=1}^{9} z_i = 0$. Thus (13) represents just one relation, in spite of being written as two equations.) While there is one relation of the form (13) for the vector (11), there are as many as four linearly independent relations of the form (13) in the case of the vector (12), far more than would be suggested by a simple counting argument: merely assuming that (12) consisted of nine random integers that sum up to 0.

This finding suggested forcibly that split relations of the type (13) are a frequent feature of odd symmetric PTE solutions, and led us to check the other odd symmetric solutions we knew for the presence of this phenomenon. Interestingly, of the more than 230 odd symmetric solutions of size 7 that we computed in several weeks of CPU time [16], more than 85% have a relation analogous to (13), that is $z_{i_1} + z_{i_2} + z_{i_3} = z_{j_1} + z_{j_2} + z_{j_3} + z_{j_4} = 0$. (Here, analogously, the i and j index sets are a disjoint partition of $\{1, 2, \ldots, 7\}$.)

While we do not yet have a rigorous explanation for this phenomenon, we have apparently discovered a new fact about the symmetric PTE solutions that had not been noted previously in the literature.

3.4 More Examples

There is a growingly vast number of results in which discoveries by integer relation algorithms played an important role. We briefly list three more of these. In *all* cases, developing the right model (i.e. the form of the relations to be searched for) was absolutely key, but the machine discovery of relations holding in the model was an equally essential prerequisite to success.

1. Borwein and Bradley [10, 11] considered the well-known formula for $\zeta(3)$, used by Apéry in his irrationality proof, and recalled that there are no immediately analogous formulae for $\zeta(5), \zeta(7), \ldots$. In other words, if there are relatively prime integers p and q such that

$$\zeta(5) = \frac{p}{q} \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k^5 \binom{2k}{k}},$$

then q is astronomically large.

However, a less known formula for $\zeta(5)$ due to Koecher suggested generalizations for $\zeta(7), \zeta(9), \ldots$, with the coefficients in these formulae determined by an LLL integer relation algorithm. For example,

$$\zeta(7) = \frac{5}{2} \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k^7 \binom{2k}{k}} + \frac{25}{2} \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k^3 \binom{2k}{k}} \sum_{j=1}^{k-1} \frac{1}{j^4}.$$

The authors were able—by finding LLL based relations for $n = 1, 2, \ldots, 10$—to encapsulate the formulae for $\zeta(4n+3)$ in a single conjectured generating function [10, 11]: For any complex z, we have

$$\begin{aligned} \sum_{n=1}^{\infty} \zeta(4n+3) z^{4n} &= \sum_{k=1}^{\infty} \frac{1}{k^3 (1 - z^4/k^4)} \\ &= \frac{5}{2} \sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{k^3 \binom{2k}{k}} \frac{1}{(1 - z^4/k^4)} \prod_{m=1}^{k-1} \frac{1 + 4z^4/m^4}{1 - z^4/m^4}. \end{aligned} \tag{14}$$

Borwein and Bradley found many reformulations of (14), including a finite sum:

$$\sum_{k=1}^{n} \frac{2n^2}{k^2} \frac{\prod_{i=1}^{n-1} (i^4 + 4k^4)}{\prod_{i=1,\, i \neq k}^{n} (k^4 - i^4)} = \binom{2n}{n}.$$

This identity was subsequently proved by Almkvist and Granville [1], thus finishing the proof of (14) and giving a rapidly converging series for any $\zeta(4n+3)$ where n is positive integer.

2. Bailey, Borwein and Plouffe [3] discovered a new series for π (and for some other polylogarithmic constants) which allows one to compute hexadecimal digits of π without computing the previous digits. The algorithm needs very little memory, does not need multiple precision and the running time grows only slightly faster than linearly in the order of the digit being computed. The key formula, discovered using the PSLQ algorithm, is [3]

$$\pi = \sum_{k=0}^{\infty} \left(\frac{1}{16}\right)^k \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6}\right).$$

The authors knew that an algorithm would follow from such a formula and spent several months hunting for one. In September 1997, Fabrice Bellard [6] used a variant formula to compute 152 binary digits of π, starting at the trillionth position (10^{12}). This computation took 12 days on 20 workstations working in parallel over the Internet.

3. The *Catalan constant* is defined by

$$G = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)^2}. \tag{15}$$

Bradley [17] considered the following series for the Catalan constant, due to Ramanujan (see [7], p. 386)

$$G = \frac{\pi}{8}\log(2+\sqrt{3}) + \frac{3}{8}\sum_{k=0}^{\infty} \frac{1}{(2k+1)^2\binom{2k}{k}}$$

and showed that this series is a member of an entire family of expansions accelerating (15). Another example of a series from this family is

$$G = \frac{\pi}{8}\log\left(\frac{10+\sqrt{50-22\sqrt{5}}}{10-\sqrt{50-22\sqrt{5}}}\right) + \frac{5}{8}\sum_{k=0}^{\infty} \frac{L(2k+1)}{(2k+1)^2\binom{2k}{k}} \tag{16}$$

where $L(n)$ are the *Lucas numbers.* At the heart of proving the general formula for this family of series are certain transformations for log tangent integrals, discovered by an LLL integer relation algorithm. For example, the expansion (16) is proved using the transformation

$$2\int_0^{\pi/4} \log(\tan\theta)\,d\theta = 5\int_0^{3\pi/20} \log(\tan\theta)\,d\theta - 5\int_0^{\pi/20} \log(\tan\theta)\,d\theta.$$

In contrast to the original series, it is easy to hunt numerically for relations between different $\int_0^{\frac{p}{q}\pi} \log(\tan\theta)\,d\theta$.

4 Implementations

We briefly mention some software implementations of integer relation algorithms known to us (the list is by no means complete). We also report performance results for these algorithms when they are used to find the minimal polynomial of a specific algebraic number that we chose as a test case.

4.1 A Test Example

Consider $A = 2^{1/6}+3^{1/5}$. The advantage of picking a simple algebraic number for our test is that we can easily precompute the minimal polynomial (using resultants) and thus have a correctness check for the results coming from integer relation algorithms.

The minimal polynomial for A is

$$\begin{aligned} M(x) &= \mathrm{res}_y(y^6-2,(x-y)^5-3) \\ &= x^{30}-18x^{25}-10x^{24}+135x^{20}-7380x^{19}+40x^{18}-540x^{15} \\ &\quad -135540x^{14}-56160x^{13}-80x^{12}+1215x^{10}-336420x^9 \\ &\quad +538380x^8-43920x^7+80x^6-1458x^5-102060x^4-98280x^3 \\ &\quad -20520x^2-1440x+697. \end{aligned}$$

We have tested different implementations of integer relation algorithms by asking them to recover the polynomial M for different input precisions at which A was numerically specified. In consecutive runs we were incrementing the precision by 5 decimal digits. The results are summarized and compared in Sections 4.2 through 4.4.

4.2 LLL-based Algorithms

An LLL-based minimal polynomial algorithm is found in most computer algebra systems. We have received quite comparable results for the following three systems. For all three systems the tests were performed on an R4600 SGI workstation running at 132 MHz. Maple and Mathematica are leading commercial computer algebra products. PARI-GP can be obtained by anonymous ftp to `megrez.math.u-bordeaux.fr`.

software system	precision needed	time needed	function used
Maple 5.5	235(310)*	27 min*	`minpoly`
Mathematica 3.0	170	33.5 min	`NumberTheory‘Recognize‘`
PARI-GP 1.39	175(265)*	27.5 min*	`lindep2`

Comments:

* The notation $d_1(d_2)$ means that the first correct result is received at the precision of d_1 decimal digits but a steady stream of correct results (when incrementing the precision by 5 digits in one step) only starts at d_2 digits. In this case, the timing in the third column is for the run at the precision of d_1 digits.

4.3 The PSLQ Algorithm

software system	precision needed	time needed
MPFUN (Fortran)	220	14 sec

The tests leading to this result were performed on a 125 MHz DEC Alpha station, using Bailey's fast three-level implementation of PSLQ written in his Fortran multiprecision arithmetic library MPFUN. Fortran sources for both MPFUN and PSLQ are available on Bailey's WWW page (see Section 2.2).

4.4 The Comparison

A detailed comparison of these (and many other) implementations of integer relation algorithms would require a much larger collection of experiments and would be a topic for a paper on its own. From the limited data that we have presented one can derive the following observations:

Numerical stability. PSLQ shows more stability, by having a clear-cut line between precisions at which the relation is not found and those where the relation is found. On the other hand, two of the three LLL-based algorithms that we have tried out were quite unstable in this sense.

Running time. Our tests were carried out on two different computers, which nonetheless are comparable in speed and performance.

Interestingly, theoretical results predict an asymptotically similar number of arithmetic operations for both types of algorithms (in both cases n is the dimension of the input vector):

- If B is a basis for the lattice L and b is the maximal length of vectors in B, then the LLL algorithm will find a reduced basis for L using $O(n^4 \log b)$ arithmetic operations [25].
- If M_a is the minimum length among all existing integer relations for a given vector a, then the PSLQ algorithm will find a relation for a using $O(n^4 + n^3 \log M_a)$ arithmetic operations [20].

The large difference between the timings observed in Sections 4.3 and 4.2 can be partially accounted for by the fact that Bailey's PSLQ implementation is highly customized and compiled with a fast multiprecision library, whereas the LLL-based methods are implemented in (mostly interpreted) general purpose computer algebra systems. Another fact that apparently plays a great role here is that the numerical stability of PSLQ admits an efficient multi-level implementation using a combination of double precision and multiple precision arithmetic, at two or even three distinct levels of working precision.

Our general experience is that Bailey's PSLQ indeed runs much faster than the LLL implementations, and we definitely recommend using PSLQ for larger input vectors and/or higher numerical precision. For small cases, the LLL-based implementations are sufficient and probably easier to use, as they do not require installation of special software but rather come as parts of widespread mathematical software packages.

Acknowledgment

We are indebted to David Bailey, Greg Fee and Chris Pinner for valuable comments.

References

[1] G. Almkvist, A. Granville, Borwein and Bradley's Apéry-like formulae for $\zeta(4n+3)$. Manuscript.

[2] D. Bailey, Numerical results on the transcendence of constants involving pi, e, and Euler's constant. *Math. Comp.* **50** (1988), 275–281.

[3] D. Bailey, P. Borwein, S. Plouffe, On the rapid computation of various polylogarithmic constants. *Math. Comp.* **66** (1997), 903–913.

[4] D. H. Bailey, H. R. P. Ferguson, Numerical results on relations between numerical constants using a new algorithm. *Math. Comp.* **53** (1989), 649–656.

[5] D. H. Bailey, S. Plouffe, Recognizing Numerical Constants. *In:* The Organic Mathematics Project Proceedings, `http://www.cecm.sfu.ca/organics`, April 12, 1996. Hardcopy version, Canadian Mathematical Society Conference Proceedings, **20** (1997), 73–88.

[6] F. Bellard, The pi challenge. `http://www-stud.enst.fr/~bellard/pi-challenge/` .

[7] J. M. Borwein, P. B. Borwein, *Pi and the AGM.* John Wiley & Sons, New York, 1987.

[8] J. M. Borwein, P. B. Borwein, A cubic counterpart of Jacobi's identity and the AGM. *Trans. Amer. Math. Soc.* **323** (1991), 691–701.

[9] J. M. Borwein, P. B. Borwein, F. G. Garvan, Some cubic identities of Ramanujan. *Trans. Amer. Math. Soc.* **343** (1994), 35–47.

[10] J. Borwein, D. Bradley, Empirically determined Apéry like formulae for zeta(4n+3). *Experiment. Math.,* in print.

[11] J. Borwein, D. Bradley, Searching symbolically for Apéry-like formulae for values of the Riemann zeta function. *SIGSAM Bulletin Communications in Computer Algebra* **30** (Issue 116, June 1996), 2–7.

[12] J. M. Borwein, D. M. Bradley, D. J. Broadhurst, Evaluations of k-fold Euler/Zagier sums: a compendium of results for arbitrary k. The Wilf Festschrift. *Electron. J. Combin.* **4** (1997), no. 2, R5, 21 pp.

[13] J. Borwein, D. M. Bradley, D. J. Broadhurst, P. Lisoněk, Special values of multidimensional polylogarithms. Manuscript.

[14] J. Borwein, D. M. Bradley, D. J. Broadhurst, P. Lisoněk, Combinatorial aspects of Euler sums. Manuscript.

[15] P. Borwein, C. Ingalls, The Prouhet-Tarry-Escott Problem revisited. *Enseign. Math.* **40** (1994), 3–27.

[16] P. Borwein, P. Lisoněk, Computational investigations of the Prouhet-Tarry-Escott Problem. Manuscript in preparation.

[17] D. Bradley, A class of series acceleration formulae for Catalan's constant. *The Ramanujan Journal,* in print.

[18] H. H. Chan, W. C. Liaw, On Russell-type modular equations. *Canad. J. Math.* (submitted).

[19] H. R. P. Ferguson, D. H. Bailey, A polynomial time, numerically stable integer relation algorithm. RNR Technical Report RNR-91-032, NASA Ames Research Center, Moffett Field, CA. December 1991.

[20] H. R. P. Ferguson, D. H. Bailey, S. Arno, Analysis of PSLQ, an integer relation finding algorithm. NAS Technical Report NAS-96-005, NASA Ames Research Center, Moffett Field, CA. April 1996.

[21] A. Gloden, *Mehrgradige Gleichungen.* P. Noordhoff, Groningen, 1944.

[22] G. H. Hardy, *Ramanujan.* Chelsea, New York, 1940.

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

[24] K. S. Kölbig, Explicit evaluation of certain definite integrals involving powers of logarithms. *J. Symb. Computation* **1** (1985), 109–114.

[25] A. K. Lenstra, H. W. Lenstra Jr., L. Lovász, Factoring polynomials with rational coefficients. *Math. Ann.* **261** (1982), 515–534.

[26] N. Magot, A. Zvonkin, Belyi functions for Archimedean solids. *In:* Proceedings, Formal Power Series and Algebraic Combinatorics, Vienna, July 1997, Volume 3, 373–389.

[27] D. Zagier, Values of zeta functions and their applications. First European Congress of Mathematics, Volume II, Birkhäuser, Boston, 1994, pp. 497–512.