High Performance Computing Meets Experimental Mathematics*

David H. Bailey
Lawrence Berkeley National Laboratory, USA
David Broadhurst
Department of Physics, Open University, UK
Yozo Hida
University of California, Berkeley, USA
Xiaoye S. Li
Lawrence Berkeley National Labratory, USA
Brandon Thompson
University of California, Berkeley, USA

May 16, 2003

Abstract

In this paper we describe some novel applications of high performance computing in a discipline now known as "experimental mathematics." The paper reviews some recent published work, and then presents some new results that have not yet appeared in the literature. A key technique inovlved in this research is the PSLQ integer relation algorithm (recently named one of ten "algorithms of the century" by Computing in Science and Engineering). This algorithm permits one to recognize a numeric constant in terms of the formula that it satisfies. We present a variant of PSLQ that is well-suited for parallel computation, and give several examples of new mathematical results that we have found using it. Two of these computations were performed on highly parallel computers, since they are not feasible on conventional systems. We also describe a new software package for performing arbitrary precision arithmetic, which is required in this research.

1. Introduction

It is ironic that although modern computing had its roots in mathematics, with the work of Turing and von Neumann, nonetheless the field of mathematics has been relatively slow to adopt computing as a serious tool for the practicing researcher, compared with the

*This research was supported by the Director, Office of Science, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy under contract number DE-AC03-76SF00098.

aggressive strides taken in other fields of science and engineering. This situation is finally starting to change. Powerful, broad-spectrum mathematical software is now available, notably the Mathematica and Maple products, and a new generation of mathematicians is aggressively utilizing this software in serious mathematical research.

Almost all mathematical research computing done to date, however, has been performed on single-CPU systems, mostly personal computers, and has been restricted to relatively passive tasks, such as performing a particular symbolic manipulation prescribed by the mathematician. Recently however, novel mathematical applications have arisen that require large, high-performance computer systems. What's more, in these new applications the computer is taking an active role in discovering new results in mathematics.

In this manuscript, we will present a brief reprise of recent results in this arena, together with some new results that have not yet appeared in the literature. A common theme of our results is integer relation detection using the PSLQ integer relation algorithm. These computations require advanced numerical techniques, involving very high precision computation, implemented on high performance computer (HPC) systems. We present these results, in part, to draw the attention of researchers in the HPC community to this new and promising application of HPC technology.

Many, but not all, of the mathematical results that we have discovered numerically have subsequently been proven rigorously. In the following, the $\doteq$ notation is used in numerically discovered relations for which a formal proof is not yet known.

2. The PSLQ Integer Relation Detection Algorithm and Variants

Let $x = (x_1, x_2, \cdots, x_n)$ be a vector of real or complex numbers. x is said to possess an integer relation if there exist integers a_i, not all zero, such that

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n \;=\; 0$$

By an *integer relation algorithm*, we mean a practical computational scheme that can recover the vector of integers a_i, if it exists, or can produce bounds within which no integer relation exists.

The first general integer relation algorithm was found in 1977 [13]. At the present time, the most effective algorithm for integer relation detection is the "PSLQ" algorithm of mathematician-sculptor Helaman Ferguson [12]. PSLQ was recently named one of ten "algorithms of the century" by the publication *Computing in Science and Engineering* [3]. In addition to possessing good numerical stability, PSLQ is guaranteed to find a relation in a polynomially bounded number of iterations.

The basic PSLQ algorithm can be stated as follows [12, 5]: Let x be the n-long input real vector, and let nint denote the nearest integer function. Select $\gamma \geq \sqrt{4/3}$ (in our computations, we use $\gamma = \sqrt{4/3}$). Then perform the following operations:

Initialize:

1. Set the $n \times n$ matrices A and B to the identity.
2. Compute the n-long vector s as $s_k := \sqrt{\sum_{j=k}^{n} x_j^2}$, and set y to the x vector, normalized by s_1.

3. Compute the initial $n \times (n-1)$ matrix H as $H_{ij} = 0$ if $i < j$, $H_{jj} := s_{j+1}/s_j$, and $H_{ij} := -y_i y_j/(s_j s_{j+1})$ if $i > j$.

4. Reduce H: For $i := 2$ to n: for $j := i-1$ to 1 step -1: set $t := \mathrm{nint}(H_{ij}/H_{jj})$; and $y_j := y_j + t y_i$; for $k := 1$ to j: set $H_{ik} := H_{ik} - t H_{jk}$; endfor; for $k := 1$ to n: set $A_{ik} := A_{ik} - t A_{jk}$ and $B_{kj} := B_{kj} + t B_{ki}$; endfor; endfor; endfor.

Iterate until an entry of y is within a reasonable tolerance of zero, or precision has been exhausted:

1. Select m such that $\gamma^i |H_{ii}|$ is maximal when $i = m$.

2. Exchange the entries of y indexed m and $m+1$, the corresponding rows of A and H, and the corresponding columns of B.

3. Remove the corner on H diagonal: If $m \le n-2$ then set $t_0 := \sqrt{H_{mm}^2 + H_{m,m+1}^2}$, $t_1 := H_{mm}/t_0$ and $t_2 := H_{m,m+1}/t_0$; for $i := m$ to n: set $t_3 := H_{im}$, $t_4 := H_{i,m+1}$, $H_{im} := t_1 t_3 + t_2 t_4$ and $H_{i,m+1} := -t_2 t_3 + t_1 t_4$; endfor; endif.

4. Reduce H: For $i := m+1$ to n: for $j := \min(i-1, m+1)$ to 1 step -1: set $t := \mathrm{nint}(H_{ij}/H_{jj})$ and $y_j := y_j + t y_i$; for $k := 1$ to j: set $H_{ik} := H_{ik} - t H_{jk}$; endfor; for $k := 1$ to n: set $A_{ik} := A_{ik} - t A_{jk}$ and $B_{kj} := B_{kj} + t B_{ki}$; endfor; endfor; endfor.

5. Norm bound: Compute $M := 1/\max_j |H_{jj}|$. Then there can exist no relation vector whose Euclidean norm is less than M.

6. Termination test: If the largest entry of A exceeds the numeric precision level, then precision is exhausted. If the smallest entry of the y vector is less than the detection threshold (typically set near the precision "epsilon"), a relation has been detected and is given in the corresponding column of B.

The strategy of PSLQ is to reduce the entries of the H matrix by means of a sequence of operations wherein integer multiples of one row are subtracted from another. The particular choice of the two rows involved in the subtraction, and the technique of removing the resulting corner above the H matrix diagonal, are the keys to efficient and numerically stable operation. As one can see from step 5 of the iteration above, the norm bound increases in inverse proportion to the reciprocal of the largest H matrix diagonal entry. Thus if the original x vector possesses an integer relation (to within a certain numerical tolerance), it will be found once the H matrix diagonal entries are sufficiently small, provided numeric precision has not been exhausted. See [12] for full details.

Some efficient "multi-level" implementations of PSLQ are described in [5]. These new variants of PSLQ run many times faster than a naive implementation of PSLQ, particularly on large problems. Even with such accelerations, however, run times are painfully long for many applications. Thus one is led to consider implementations of PSLQ, both single-level and multi-level, on parallel computer systems. Unfortunately, iteration step 4 above exhibits a recursion that inhibits parallel execution. What's more, the multi-level schemes we just described have the perverse effect of removing other possible avenues for parallel computation.

Recently a "multi-pair" variant of PSLQ was discovered that is suitable for parallel computation [5]. The basic multi-pair PSLQ algorithm can be stated as follows. Here $\gamma \geq \sqrt{4/3}$ as before, and $\beta = 0.4$.

Initialize:

1. For $j := 1$ to n: for $i := 1$ to n: if $i = j$ then set $A_{ij} := 1$ and $B_{ij} := 1$ else set $A_{ij} := 0$ and $B_{ij} := 0$; endfor; endfor.

2. For $k := 1$ to n: set $s_k := \sqrt{\sum_{j=k}^{n} x_j^2}$; endfor; set $t = 1/s_1$; for $k := 1$ to n: set $y_k := tx_k$; $s_k := ts_k$; endfor.

3. Initial H: For $j := 1$ to $n-1$: for $i := 1$ to $j-1$: set $H_{ij} := 0$; endfor; set $H_{jj} := s_{j+1}/s_j$; for $i := j+1$ to n: set $H_{ij} := -y_i y_j/(s_j s_{j+1})$; endfor; endfor.

Iteration: Repeat the following steps until precision has been exhausted or a relation has been detected.

1. Sort the entries of the $(n-1)$-long vector $\{\gamma^i |H_{ii}|\}$ in decreasing order, producing the sort indices.

2. Beginning at the sort index m_1 corresponding to the largest $\gamma^i |H_{ii}|$, select pairs of indices $(m_i, m_i + 1)$, where m_i is the sort index. If at any step either m_i or $m_i + 1$ has already been selected, pass to the next index in the list. Continue until either βn pairs have been selected, or the list is exhausted. Let p denote the number of pairs actually selected in this manner.

3. For $i := 1$ to p, exchange the entries of y indexed m_i and $m_i + 1$, and the corresponding rows of A, B and H; endfor.

4. Remove corners on H diagonal: For $i := 1$ to p: if $m_i \leq n-2$ then set $t_0 := \sqrt{H_{m_i,m_i}^2 + H_{m_i,m_i+1}^2}$, $t_1 := H_{m_i,m_i}/t_0$ and $t_2 := H_{m_i,m_i+1}/t_0$; for $i := m_i$ to n: set $t_3 := H_{i,m_i}$; $t_4 := H_{i,m_i+1}$; $H_{i,m_i} := t_1 t_3 + t_2 t_4$; and $H_{i,m_i+1} := -t_2 t_3 + t_1 t_4$; endfor; endif; endfor.

5. Reduce H: For $i := 2$ to n: for $j := 1$ to $n - i + 1$: set $l := i + j - 1$; for $k := j + 1$ to $l - 1$: set $H_{lj} := H_{lj} - T_{lk} H_{kj}$; endfor; set $T_{lj} := \mathrm{nint}(H_{lj}/H_{jj})$ and $H_{lj} := H_{lj} - T_{lj} H_{jj}$; endfor; endfor.

6. Update y: For $j := 1$ to $n-1$: for $i := j+1$ to n: set $y_j := y_j + T_{ij} y_i$; endfor; endfor.

7. Update A and B: For $k := 1$ to n: for $j := 1$ to $n-1$: for $i := j+1$ to n: set $A_{ik} := A_{ik} - T_{ij} A_{jk}$ and $B_{jk} := B_{jk} + T_{ij} B_{ik}$; endfor; endfor; endfor.

8. Norm bound: Compute $M := 1/\max_j |H_{jj}|$. Then there can exist no relation vector whose Euclidean norm is less than M.

9. Termination test: If the largest entry of A exceeds the numeric precision level, then precision is exhausted. If the smallest entry of the y vector is less than the detection threshold (typically set near the precision "epsilon"), a relation has been detected and is given in the corresponding row of B.

The basic idea of the multi-pair PSLQ algorithm is that each iteration performs, in effect, up to βn independent row interchanges. Thus the number of iterations that must be performed in sequential order is reduced by a factor of roughly βn, while the amount of computation (and the opportunity for parallelism) involved in a single iteration increases by a factor of roughly βn. In addition, the reduction step (step 5) is performed in a diagonal sweep over the H matrix, thus avoiding the recursion that plagues standard PSLQ. This basic multi-pair PSLQ algorithm can be accelerated by multi-level implementation schemes (similar to those used to accelerate standard PSLQ), while still maintaining reasonable parallel efficiency.

3. A New Arbitrary Precision Computation Package

It should be emphasized that for almost all applications of PSLQ or any other integer relation algorithm, high-precision arithmetic (accurate to hundreds or even thousands of digits) must be used. Only a very small class of relations can be recovered reliably with the 64-bit IEEE floating-point arithmetic that is available on current computer systems. This follows from basic information theory considerations, wherein it can be seen that to recover a relation of length n, with coefficients of maximum size d digits, the input vector x must be specified to at least nd digits, and floating-point arithmetic accurate to at least nd digits must be employed in the integer relation detection algorithm. The PSLQ algorithm is quite efficient in this regard — it is typically successful in recovering relations when the input data and working precision are set as low as $1.1 \times nd$ digits.

High-precision arithmetic software is available from a number of sources. The commercial products Maple and Mathematica include multiple precision arithmetic facilities, and some "freeware" packages are available as well.

We take this opportunity to announce and briefly describe "arprec," a new software package for performing calculations with arbitrarily high precision. It consists of a revision and extension of Bailey's earlier "mpfun" package [2], enhanced with special IEEE numerical techniques described in an earlier paper by some of present authors [14]. In particular, this package:

- Is written in C++ for performance and broad portability.
- Includes C++ and F-90 translation modules that permit one to utilize the package with only minor changes to an existing C++ or F-90 program.
- Includes integer, floating and complex datatypes.
- Permits datatypes of different precision levels to be defined.
- Inter-operates with conventional integer and floating-point datatypes.
- Includes common transcendental functions (sqrt, exp, sin, etc).

- Includes quadrature routines (i.e. numerical integration).
- Includes PSLQ integer relation routines.
- Includes special routines for extra-high precision (> 1000 digits).

As noted above, the package is combined with translation modules that permit one to use the library routines with only minor changes to one's conventional source code (either C++ or Fortran-90). For example, the following simple Fortran-90 program

```
program main
use mpmodule
type (mp_real) a, b
a = 1.d0
b = cos(a)**2 + sin(a)**2 - 1.d0
call mpwrite (6, b)
stop
end program
```

verifies that $\cos^2(1)+\sin^2(1)-1 = 0$ to 1000 decimal digit accuracy (or any other specified level of accuracy). The arprec software is available from `http://www.nersc.gov/~dhbailey/mpdist/index.html`.

3. Finding Algebraic Relations Using PSLQ

A straightforward application of PSLQ is to determine whether or not a given constant t, whose value can be computed to high precision, is algebraic of some degree n or less (i.e., t satisfies an equation of the form $0 = a_0 + a_1 t + a_2 t^2 + \cdots + a_n t^n$ for integers a_i). This can be done by first computing the vector $T = (1, t, t^2, \cdots, t^n)$ to high precision and then applying PSLQ to the resulting $(n+1)$-long vector. If a relation is found for T, then this relation vector is precisely the set of integer coefficients of a polynomial satisfied by t (to within the numerical accuracy being used).

One of the first results of this sort was the identification of the constant $B_3 = 3.54409035955\cdots$, namely the third bifurcation point of the logistic map $x_{k+1} = r x_k(1 - x_k)$, which exhibits period doubling shortly before the onset of chaos. To be precise, B_3 is the smallest value of the parameter r such that successive iterates x_k exhibit eight-way periodicity instead of four-way periodicity. Computations using a predecessor algorithm to PSLQ found that B_3 is a root the polynomial

$$\begin{aligned} 0 \;=\; & 4913 + 2108t^2 - 604t^3 - 977t^4 + 8t^5 + 44t^6 + 392t^7 - 193t^8 - 40t^9 \\ & + 48t^{10} - 12t^{11} + t^{12} \end{aligned}$$

Recently, one of the present authors identified $B_4 = 3.564407268705\cdots$, the fourth bifurcation point of the logistic map, using PSLQ [5]. Some conjectural reasoning had suggested that B_4 might satisfy a 240-degree polynomial, and that the constant $\alpha = -B_4(B_4 - 2)$ might satisfy a 120-degree polynomial. In order to test this hypothesis, we applied PSLQ to the 121-long vector $(1, \alpha, \alpha^2, \cdots, \alpha^{120})$. Indeed, a relation was found, although 10,000 digit arithmetic and several hours of computation on a large Compaq computer system were required. The recovered integer coefficients descend monotonically

from 257^{30}, which is approximately 1.986×10^{72}, to one. Such regularity strongly suggests that the recovered relation is real, and not just an artifact of numerical computation.

4. A New Formula for Pi

For centuries mathematicians have assumed that there is no shortcut to computing just the n-th digit of π. Thus, it came as no small surprise when such an algorithm was recently discovered [4]. In particular, this simple scheme allows one to compute the n-th binary (or hexadecimal) digit of π, plus a few additional digits beginning at that position, without computing any of the first $n-1$ digits. This can be done using a simple algorithm that requires only minimal memory. This scheme is based on the following new formula, which was discovered in 1996 using PSLQ:

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

Since then, hundreds of additional results of this type have been found, including formulas for certain constants that arise in quantum field theory. All of these discoveries have been made, or at least suggested, by PSLQ computations. Just one example of many new results that could be cited is the following formula:

$$\begin{aligned} \tan^{-1}\left(\frac{4}{5}\right) = \frac{1}{2^{17}} \sum_{k=0}^{\infty} \frac{1}{2^{20k}} \Bigg(& \frac{524288}{40k+2} - \frac{393216}{40k+4} - \frac{491520}{40k+5} + \frac{163840}{40k+8} + \frac{32768}{40k+10} \\ & - \frac{24576}{40k+12} + \frac{5120}{40k+15} + \frac{10240}{40k+16} + \frac{2048}{40k+18} + \frac{1024}{40k+20} + \frac{640}{40k+24} \\ & + \frac{480}{40k+25} + \frac{128}{40k+26} - \frac{96}{40k+28} + \frac{40}{40k+32} + \frac{8}{40k+34} - \frac{5}{40k+35} \\ & - \frac{6}{40k+36} \Bigg) \end{aligned}$$

An updated compendium of these results is available in [1].

5. Implications for Normality

Until recently, the results of the previous section were regarded by some to be in the arena of "recreational mathematics" — amusing, but of no further consequence in mathematics or any other scientific discipline. To the contrary, it now appears that the existence of these formulas has deep implications for the centuries-old question of whether (and why) constants such as π and $\log 2$ are "normal" [7]. Here "normal" to a given base b means that all m-long base-b digit strings occur with a limiting frequency that is precisely what one would expect from random digits, namely b^{-m}. It is a true, if counter-intuitive consequence of measure theory that "almost all" real numbers are normal to a given number base b, and in fact to all bases b simultaneously. What's more, it appears from computational studies that all of the naturally occurring constants of mathematics, including π, e, $\sqrt{2}, \log 2, \sqrt{\pi}$, are normal. One can even conjecture that *every* irrational algebraic number is normal, as there are no known or apparent counter-examples. Nonetheless it is an embarrassing fact of modern mathematics that not a single one of the above-mentioned constants has ever been *proven* to be normal to any base. Indeed, until recently normality proofs had only been obtained for a handful of artificially contrived constants.

Recently progress was achieved on this front. In particular, it has now been proven that the normality of a certain class of mathematical constants, including π and $\log 2$, reduces to a plausible conjecture from the field of chaotic sequences. These new results are direct consequences of the recent computer-based discoveries mentioned in the previous section. A highly readable account of these developments is given in a recent *Science News* article [15].

In the latest development along this line, these same methods have now led to a rigorous proof that a certain infinite class of reals *is* normal. The simplest instance of this class is the constant

$$\begin{aligned} \alpha_{2,3} &= \sum_{k=1}^{\infty} \frac{1}{3^k 2^{3^k}} \\ &= \texttt{0.041883680831502985}07\ldots_{10} \\ &= \texttt{0.0AB8E38F684BDA12F684}\ldots_{16}\,, \end{aligned}$$

In this instance, it has been proven that $\alpha_{2,3}$ is normal base 2 (or, equivalently, base 16) — each m-long binary (or hexadecimal) digit string appears with a limiting frequency that is precisely 2^{-m} (or 16^{-m} in the case of hexadecimal digits). The normality of this particular constant was actually demonstrated in a little-known paper by Stoneham in 1973. However the recent results extend to a much larger class of similar constants. Full details of these latest results are given in the technical report [8].

6. Identification of Combinatorial Sum Constants

In another recent application of high performance computing to mathematical number theory, PSLQ has been used to investigate sums of the form

$$S(k) := \sum_{n>0} \frac{1}{n^k \binom{2n}{n}}$$

For small k, these constants are known to satisfy simple mathematical identities, such as $S(4) = 17\pi^4/3240$. For many years researchers have sought generalizations of these formulas for $k > 4$. Recently some more general formulas were found using PSLQ. In particular, the constants $\{S(k)|k = 5\ldots 20\}$ have now been evaluated in terms of multiple zeta values [9], which are defined by

$$\begin{aligned} &\zeta(s_1, s_2, \cdots, s_r) \\ &= \sum_{k_1>k_2>\cdots>k_r>0} \frac{1}{k_1^{s_1}\, k_2^{s_2} \cdots k_r^{s_r}} \end{aligned}$$

and multiple Clausen values [10] of the form

$$M(a,b) = \sum_{n_1>n_2>\ldots>n_b>0} \frac{\sin(n_1\pi/3)}{n_1^a} \prod_{j=1}^{b} \frac{1}{n_j}$$

A sample result, obtained using PSLQ, is the following:

$$\begin{aligned} S(9) \doteq\ & \pi\left[2M(7,1) + \frac{8}{3}M(5,3) + \frac{8}{9}\zeta(2)M(5,1)\right] - \frac{13921}{216}\zeta(9) \\ & + \frac{6211}{486}\zeta(7)\zeta(2) + \frac{8101}{648}\zeta(6)\zeta(3) + \frac{331}{18}\zeta(5)\zeta(4) - \frac{8}{9}\zeta^3(3) \end{aligned}$$

Recently $S(20)$ was evaluated with this methodology, which involved an integer relation problem of size $n = 118$. This solution required 5,000 digit arithmetic and several hours of computation on a high-end workstation. The solution is given explicitly in [5].

7. Identification of Euler-Zeta Sum Constants

One may define *Euler-Zeta sums* by [9]

$$\zeta\begin{pmatrix} s_1, & s_2 & \cdots & s_r \\ \sigma_1, & \sigma_2 & \cdots & \sigma_r \end{pmatrix} = \sum_{k_1>k_2>\cdots>k_r>0} \frac{\sigma_1^{k_1}}{k_1^{s_1}} \frac{\sigma_2^{k_2}}{k_2^{s_2}} \cdots \frac{\sigma_r^{k_r}}{k_r^{s_r}}$$

where $\sigma_j = \pm 1$ are signs and $s_j > 0$ are integers. When all the signs are positive, one has a multiple zeta value. Constants with alternating signs appear in problems such as computation of the magnetic moment of the electron.

One of the present authors had conjectured that the dimension of the space of Euler sums with weight $w := \sum_j s_j$ is the Fibonacci number defined by $F_{w+1} = F_w + F_{w-1}$, with $F_1 = F_2 = 1$. The first few Fibonacci numbers are $1, 2, 3, 5, 8, 13, 21, \ldots$. In an attempt to establish this conjecture, PSLQ was used to obtain complete reductions of all Euler sums to a basis of size F_{w+1} at weights $w \le 9$. At weights $w = 10$ and $w = 11$ the conjecture has been tested by application of PSLQ in more than 600 cases. At weight $w = 11$ these tests involve solving integer relations of size $n = F_{12} + 1 = 145$. These solutions required 5,000 digit arithmetic and many hours of processing on an IBM SP parallel computer system. In this instance the parallel processing was fairly straightforward, but the total cost of the computation (approximately 2,000 CPU-hours) is beyond the reach of what is reasonable to do on any single-processor system [5].

8. A Relation for a Root of Lehmer's Polynomial

The findings reported in this section, which are computationally the most demanding that we have attempted, arose from the numerical discovery (using PSLQ) by the one of the present authors that

$$\begin{aligned} \alpha^{630} - 1 &= (\alpha^{315} - 1)(\alpha^{210} - 1)(\alpha^{126} - 1)^2(\alpha^{90} - 1)(\alpha^3 - 1)^3(\alpha^2 - 1)^5(\alpha - 1)^3 / \\ &\quad [(\alpha^{35} - 1)(\alpha^{15} - 1)^2(\alpha^{14} - 1)^2(\alpha^5 - 1)^6\alpha^{68}] \end{aligned}$$

where $\alpha = 1.176280818259917\ldots$ is the larger real root of Lehmer's polynomial

$$0 = \alpha^{10} + \alpha^9 - \alpha^7 - \alpha^6 - \alpha^5 - \alpha^4 - \alpha^3 + \alpha + 1$$

Once discovered numerically, the relation above was rigorously proven by repeated substitution for α^{10}. The relation led this same author to believe that a valid ladder of polylogarithms exists at order $n = 17$, contrary to a suggestion in [16]. Indeed, we were able to find 125 non-zero integers a, b_j, c_k, up to 292 digits in size, such that the relation

$$a\,\zeta(17) \doteq \sum_{j=0}^{8} b_j\,\pi^{2j}(\log\alpha)^{17-2j} + \sum_{k\in D(\mathcal{S})} c_k\,\mathrm{Li}_{17}(\alpha^{-k})$$

holds to more than 50,000 decimal digits. Here the 115 indices k in $\mathrm{Li}_n(\alpha_1^{-k}) := \sum_{r>0} \alpha_1^{-kr}/r^n$ are drawn from the set, $D(\mathcal{S})$, of positive integers that divide at least one element of the

set $\{29, 47, 50, 52, 56, 57, 64, 74, 75, 76, 78, 84, 86, 92, 96, 98, 108, 110, 118, 124, 130, 132, 138, 144, 154, 160, 165, 175, 182, 186, 195, 204, 212, 240, 246, 270, 286, 360, 630\}$. The resulting set of integers can be obtained from [11].

This set of 125 integers was found in more than more way. One of these computations was performed on the NERSC Cray T3E, a highly parallel system at Lawrence Berkeley Laboratory. In this calculation one of the present authors employed a three-level version of the "multi-pair" PSLQ algorithm described in Section 2, implemented using MPI and Fortran-90. This computation required 50,000 decimal digit arithmetic, and approximately 44 hours on 32 CPUs of the T3E, completing after 236,713 iterations. Full details are given in the technical report [6].

In order to test the new arprec arbitrary precision computation package, we recently (July 2002) repeated this run, this time using the "seaborg" system at NERSC. This is a large IBM SP computer system, containing 184 nodes, each with 16 Power3 CPUs, for a total of nearly 2,944 CPUs. For this computation, 64 CPUs (four nodes) were used. The computation completed in 236,555 iterations (slightly different than before, due to minor numerical differences), requiring 16.6 hours. 50,000 digit arithmetic (using the arprec software) was used here, as in the previous computation.

In the latest run, the minimum and maximum y entries at the point of relation detection were approximately $1.6326 \times 10^{-49772}$ and $1.3489 \times 10^{-36401}$, respectively. The ratio of these two values, which can be thought of as a "confidence ratio" of the results, is approximately $8.2623 \times 10^{-13372}$. This very small value means that it is exceedingly unlikely that these results are merely a spurious artifact of numerical round-off error (although as before such computations cannot be taken as rigorous proof of these relations).

Because of both the large memory requirement and the CPU time requirement, this calculation is not feasible on any conventional single-processor computer system. We believe this to be the largest single integer relation problem ever solved.

9. Conclusion

For many years, mathematical researchers have dreamed of a facility that permits one to recognize a constant, whose numerical value can be computed, in terms of the mathematical formula that it satisfies. With the advent of efficient integer relation detection algorithms such as PSLQ and its variants, implemented on high performance computer systems, that time has arrived. Using these algorithms, numerous new facts of mathematics and physics have been discovered, and these discoveries have in turn led to valuable new insights. This is an excellent example of "experimental mathematics," namely the utilization of modern computer technology in the discovery of new mathematical principles. Because of the expected continuation in exponential improvement in computer performance, due to Moore's Law, together with a significantly greater familiarity with these tools on the part of the younger generation of mathematicians, experimental approaches are expected to play a much wider role in both pure and applied mathematics during the next century. We believe that increased dialogue between the mathematical community and the high performance computing community will help realize this dream.

References

[1] David H. Bailey, "A Compendium of BBP-Type Formulas for Mathematical Constants," manuscript Nov. 2000.

[2] David H. Bailey, "A Fortran-90 Based Multiprecision System", *ACM Transactions on Mathematical Software*, vol. 21, no. 4, 1995, pg. 379–387.

[3] David H. Bailey, "Integer Relation Detection," *Computing in Science and Engineering*, vol. 2, no. 1 (Jan-Feb. 2000), pg. 14–20.

[4] David H. Bailey, Peter B. Borwein and Simon Plouffe, "On The Rapid Computation of Various Polylogarithmic Constants", *Mathematics of Computation*, vol. 66, no. 218, 1997, pg. 903–913.

[5] David H. Bailey and David Broadhurst, "Parallel Integer Relation Detection: Techniques and Applications," to appear in *Mathematics of Computation*.

[6] David H. Bailey and David J. Broadhurst, "A Seventeenth-Order Polylogarithm Ladder," manuscript, 2001.

[7] David H. Bailey and Richard E. Crandall, "On the Random Character of Fundamental Constant Expansions," *Experimental Mathematics*, vol. 10, no. 2 (June 2001), pg. 175-190.

[8] David H. Bailey and Richard E. Crandall, "Random Generators and Normal Numbers," manuscript, Mar. 2002.

[9] Jonathan M. Borwein, David M. Bradley and David J. Broadhurst, "Evaluations of k-fold Euler/Zagier Sums: A Compendium of Results for Arbitrary k," *Electronic Journal of Combinatorics*, vol. 4, no. 2, 1997, #R5.

[10] Jonathan M. Borwein, David J. Broadhurst and Joel Kamnitzer, "Central Binomial Sums and Multiple Clausen Values (with Connections to Zeta Values)," available from `http://www.cecm.sfu.ca/preprints`.

[11] David H. Broadhurst, `ftp://physics.open.ac.uk/pub/physics/dbroadhu/lehmer/integers.txt`

[12] Helaman R. P. Ferguson, David H. Bailey and Stephen Arno, "Analysis of PSLQ, An Integer Relation Finding Algorithm," *Mathematics of Computation*, vol. 68, 1999, pg. 351–369.

[13] Helaman R. P. Ferguson and Rodney W. Forcade, "Generalization of the Euclidean Algorithm for Real Numbers to All Dimensions Higher Than Two," *Bulletin of the American Mathematical Society*, vol. 1, 1979, pp. 912–914.

[14] Yozo Hida, Xiaoye S. Li and David H. Bailey, "Algorithms for Quad-Double Precision Floating Point Arithmetic," *Proceedings of ARITH-15*, IEEE Computer Society, 2001.

[15] Ivars Peterson, "Pi al a Mode," *Science News*, vol. 160, no. 9 (Sept. 1, 2001), pg. 136–137.

[16] Don Zagier, "Special Values and Functional Equations of Polylogarithms", Appendix A in Leonard Lewin, editor, *Structural Properties of Polylogarithms*, Mathematical Surveys and Monographs, vol. 37, American Mathematical Society, Providence, RI, 1991.