MATHEMATICS OF COMPUTATION
VOLUME 53, NUMBER 188
OCTOBER 1989, PAGES 649–656

Numerical Results on Relations Between Fundamental Constants Using a New Algorithm

By David H. Bailey and Helaman R. P. Ferguson

Abstract. Let $x = (x_1, x_2, \ldots, x_n)$ be a vector of real numbers. x is said to possess an integer relation if there exist integers a_i not all zero such that $a_1x_1 + a_2x_2 + \cdots + a_nx_n = 0$. Beginning ten years ago, algorithms were discovered by one of us which, for any n, are guaranteed to either find a relation if one exists, or else establish bounds within which no relation can exist. One of those algorithms has been employed to study whether or not certain fundamental mathematical constants satisfy simple algebraic polynomials.

Recently, one of us discovered a new relation-finding algorithm that is much more efficient, both in terms of run time and numerical precision. This algorithm has now been implemented on high-speed computers, using multiprecision arithmetic. With the help of these programs, several of the previous numerical results on mathematical constants have been extended, and other possible relationships between certain constants have been studied. This paper describes this new algorithm, summarizes the numerical results, and discusses other possible applications.

In particular, it is established that none of the following constants satisfies a simple, low-degree polynomial: γ (Euler's constant), $\log\gamma$, $\log\pi$, ρ_1 (the imaginary part of the first zero of Riemann's zeta function), $\log\rho_1, \zeta(3)$ (Riemann's zeta function evaluated at 3), and $\log\zeta(3)$. Several classes of possible additive and multiplicative relationships between these and related constants are ruled out. Results are also cited for Feigenbaum's constant, derived from the theory of chaos, and two constants of fundamental physics, derived from experiment.

1. Introduction.

The problem of finding integer relations among a set of real numbers was first studied by Euclid, who gave an iterative algorithm which, when applied to two real numbers, either terminates, yielding an exact relation, or else produces an infinite sequence of approximate relations. The generalization of this problem for $n > 2$ has been attempted by Euler, Jacobi, Poincaré, Minkowski, Perron, Brun, Bernstein, among others. However, none of their iterative algorithms has been proven to work for $n > 3$, and numerous counterexamples have been found. In the case where the entries of a vector x have no exact integer relations, some of these algorithms provide a sequence of lattice approximations that converges to the line between the origin and x in the angular sense, but none produces a sequence that converges to the line in the absolute distance sense.

A breakthrough in this area occurred in 1979 with the discovery by one of us and R. Forcade of a recursive algorithm that is guaranteed to find an integer relation for a vector x of any length n if a relation exists [9], [10]. If the vector x does not satisfy an exact relation, then this algorithm produces a sequence of lattice approximations that converges to the line in the absolute distance sense, not just

Received June 6, 1988.
1980 *Mathematics Subject Classification* (1985 *Revision*). Primary 11–04, 11Y60, 11J82.

in the angular sense. Further, this algorithm provides a means of establishing firm lower bounds on the size of any possible relation. Later, some nonrecursive algorithms were discovered that share these properties [11]. It has been established that these algorithms have polynomial complexity [16].

An unfortunate feature of the above algorithms, that severely limits their practical application, is that they require enormously high numerical precision (and correspondingly long run times) in order to obtain meaningful results. For example, one of the calculations cited in [2] established that Euler's constant γ cannot satisfy any algebraic polynomial of degree eight or less and with coefficients of size 10^9 or smaller. This calculation, which employed one of the above algorithms, required 6,144-digit precision and two hours CPU time on a Cray-2 supercomputer. Such huge precision requirements utterly rule out the usage of these algorithms to study numbers obtained from physical measurements. Furthermore, even in cases where input values can be obtained to very high precision (such as mathematical constants), the computational expense of such high-precision calculations limits the degree and size of relations that can be practically explored.

Fundamental arguments indicate that such heroic levels of numerical precision should not be necessary in order to resolve the question of whether or not an integer relation exists among a set of real numbers. For example, let x be a unit vector in R^n. Then consider the set of sums $\{\sum a_i x_i \mid a_i \in [-10^d, 10^d]\}$. It is easy to show that except for a set of x vectors of small measure, the density of these sums in the vicinity of zero is of the order of $10^{d(n-1)}$. Thus, if the vector x is specified to dn digits or so, and if calculations are also performed to this precision, then one would not expect "at random" to find any of these calculated sums near machine zero (i.e., 10^{-dn}), unless of course that sum is exactly zero. When this reasoning is applied to the size of the bounds found in the calculation mentioned above, it follows that 100-digit arithmetic should in theory be more than sufficient to obtain those results. Therefore, it is plausible that algorithms much more efficient in their numerical precision requirement should exist. Since computer run time is roughly proportional to $m \log m$, where m is the number of words of precision, it is reasonable to expect that the run time of such algorithms would be correspondingly lower.

One step in this direction was reported by Kannan and McGeoch [15], who utilized the Lovász basis reduction algorithm to obtain bounds on any polynomial that could be satisfied by $\pi + e$ or $\pi - e$. Their technique was efficient enough that it could be run on a VAX 11/780. Recently, one of us discovered a new relation-finding algorithm, which includes guarantees of finding relations and establishing bounds similar to the previous algorithms, but which features vastly improved run time and numerical efficiency [11], [12]. For example, using this algorithm, a bound on degree-eight polynomials for γ similar to that mentioned above has been obtained with only 186-digit arithmetic, instead of 6,144-digit arithmetic. Further, this run required only 23 seconds CPU time instead of two hours. The ratios of both numerical precision and CPU time are over 300 to one. It is not known at the present time whether or not this new algorithm is guaranteed to halt in polynomial time. However, this does not affect either the usage of the algorithm or the validity of numerical results obtained with it.

This paper gives details of the implementation of this new relation-finding algorithm and gives the results of computations that employed this technique to search for relations between certain constants of mathematics. While no exact relationship was discovered in this process, bounds were obtained on the sizes of possible relations that are large enough to rule out any simple, low-degree relations. Some results are also cited for constants known only to modest precision, such as Feigenbaum's constant (from the theory of chaotic behavior) and two of the fundamental constants of physics. Distinct versions of these programs were run both on a Cray-2 supercomputer operated by the Numerical Aerodynamic Simulation System at NASA Ames Research Center and on a Silicon Graphics IRIS 4D workstation. By repeating equivalent problems on such dissimilar systems, using distinct multiprecision routines, a high degree of confidence can be attached to the computed results.

2. The Partial Sum of Squares (PSOS) Relation-Finding Algorithm. The new relation-finding algorithm, which will hereafter be referred to as the PSOS algorithm, can be briefly and completely stated as follows [12], [13]. Let $\operatorname{nint}(t)$ denote the nearest integer to t (for the case of half-integer values, the integer with the smaller magnitude is taken). For a given vector $z = (z_1, z_2, \ldots, z_n)$, let $S_k(z)$ denote the partial sum of squares of z:

$$S_k(z) = \sum_{j=k}^{n} z_j^2, \qquad 1 \le k \le n.$$

Given an arbitrary real row vector x of length n, set $X_0 = x$ and $A_0 = I_n$. Successive values of X_k and A_k are defined by $X_{k+1} = X_k B_k^{-1}$ and $A_{k+1} = B_k A_k$, where B_k and B_k^{-1} are computed with the following four-step procedure:

Step 1 (Sign). Let $X_k = (x_1, x_2, \ldots, x_n)$. If any $x_j = 0$, set the termination flag. In any event, set the diagonal matrix $T_{jj} = \operatorname{sign}(x_j)$, and set $y = xT$. Note that $T^{-1} = T$.

Step 2 (Sort). If any $y_i = y_j$, $i \ne j$, then set the termination flag. In any event, define P to be the permutation matrix such that $z = yP$, where $z_1 \ge z_2 \ge \cdots \ge z_n \ge 0$. Note that $P^{-1} = P^t$.

Step 3 (Reduce). Calculate the $n \times n$ matrices D and $E = D^{-1}$ from

$$D_{ij} = \begin{cases} 0, & i < j, \\ 1, & i = j, \\ \operatorname{nint}\left[\dfrac{z_j}{S_{j+1}(z)} \displaystyle\sum_{k=j+1}^{i} D_{ik} z_k\right], & i \ge j+1; \end{cases}$$

$$E_{ij} = \begin{cases} 0, & i < j, \\ 1, & i = j, \\ -\displaystyle\sum_{k=j+1}^{i} E_{ik} D_{kj}, & i \ge j+1. \end{cases}$$

Step 4 (Restore). Set $B_k^{-1} = TPD^{-1}$ and $B_k = DP^{-1}T$.

If the termination flag was set in Step 1, then the jth column of A_k^{-1} is a relation for x. If the termination flag was set in Step 2, then the difference between the ith and the jth columns of A_k^{-1} is a relation for x. If the algorithm has not terminated

in k iterations, then it has been established ([12], see also [10]) that the Euclidean norm of any relation R for x must satisfy

$$|R| \geq \max_{j \leq k} \left[\min_i \left(\frac{1}{|\text{row}_i A_j Q|} \right) \right],$$

where $Q = I_n - x^t x/(xx^t)$. The first "max" is included because the quantity inside the brackets does not necessarily decrease monotonically with successive iterations of the algorithm.

3. Multiprecision Techniques. Although the precision requirement of the PSOS algorithm is much less than for previous versions, it still requires multiprecision arithmetic in Step 3 above to obtain strong results for n greater than three or four. For this purpose, a package of high-performance multiprecision arithmetic routines was employed. These routines are similar to those described in detail in [1] and [2]. The main difference in the routines used for this application is the incorporation of an even faster complex FFT routine [3] at the heart of the multiprecision multiplication procedure. This new FFT, which employs a radix-4 version of an algorithm suggested by Swarztrauber [19], is presently the fastest known technique for performing power-of-two FFTs on the Cray-2. In fact, this routine has been adopted by Cray Research, Inc. as their library one-dimensional FFT routine for the Cray-2, after coding some loops in assembly language to further boost performance. Unfortunately, Cray's library version failed to preserve a key property essential for multiprecision computation, namely the ability to perform an FFT for any power-of-two size up to and including the size for which it is initialized. For this reason, the Fortran version of this FFT routine was employed for the Cray-2 calculations.

As will be discussed later, the calculations described below have been duplicated on a Silicon Graphics IRIS 4D workstation as a validity check. On this system, a somewhat modified multiprecision package was employed. It differs from the Cray-2 version mainly in the method for releasing carries (a scalar algorithm was employed) and in the FFT routine (a radix-2 version of Swarztrauber's FFT algorithm was used). These changes, together with the fundamental hardware differences in the floating-point operation of these two systems, resulted in differences in the trailing digits of the results of some operations.

The techniques used to compute the mathematical constants, including the evaluation of logarithms and exponentials, are generally the same as was described in [2], and so will not be discussed in detail here. It will suffice to mention that most of these calculations employed the quadratically and quartically convergent algorithms recently discovered by the Borweins [4], [5], [6]. The constants calculated for the present experiments that were not discussed in [2] include ρ_k (the imaginary parts of the zeros of Riemann's zeta function), $\varsigma(3)$ (Riemann's zeta function evaluated at 3), and Feigenbaum's constant. $\varsigma(3)$ was computed here using the formula

$$\varsigma(3) = \frac{7\pi^3}{180} - 2\sum_{k=1}^{\infty} \frac{1}{k^3(e^{2\pi k} - 1)}.$$

The constants ρ_k were not computed by the authors, but instead were obtained from Andrew Odlyzko of AT&T Bell Laboratories, who has performed extensive

computations with these numbers [17]. The value of Feigenbaum's constant (4.6692016091029) was obtained from [18].

As with previous relation-finding algorithms, tests for zero and tests for equality must be handled carefully, or else actual relations may be missed and false relations may be detected. Since equality can be checked by subtraction, each of these reduces to a test for zero. The multiprecision programs checked for zero by testing for numbers whose exponent is less than $\log_2 m - 2 - m$, where m is the number of mantissa words of precision. Since the radix used was 10^6, this corresponds to a decimal exponent of six times this value. This tolerance was found to be generous for this application, since in test cases where an actual relation was recovered, the actual exponent of a detected zero was never more than $2 - m$ and was usually $-m$ or $1 - m$.

4. Reliability of the Calculations. Whenever results of this sort are cited, the question of their reliability arises. There are of course many possible sources of error in these calculations. There could be programming errors in implementing the PSOS algorithm. There could be programming errors in either the basic multiprecision arithmetic routines or in the higher-level routines, such as those that evaluate π or extract natural logarithms. In spite of the high levels of precision employed, subtle numerical errors could have occurred that could have caused the programs to miss an actual relation. Compiler errors could have generated incorrect machine code. Finally, there is always the possibility that hardware errors have occurred, nullifying the results.

None of these possibilities can be absolutely ruled out, and thus these calculated results do not enjoy the certainty of a mathematical proof. However, measures have been taken to reduce the uncertainties inherent in these calculations to negligible levels. The most significant measure of this sort was to perform these calculations on two completely different computer systems: a Cray-2 supercomputer and a Silicon Graphics workstation. The one is a high-speed vector machine, while the other is a much simpler scalar system. Obviously, the Fortran compilers for the two systems are completely different. In addition, as was briefly mentioned earlier, different programs were employed for the IRIS calculations than on the Cray-2. The fundamental floating-point hardware differences in the machines, together with the program differences, resulted in discrepancies in the trailing digits of results of multiprecision operations. Just as performing a single-precision calculation on different computer systems with different floating-point hardware will disclose the extent to which numerical uncertainties are significant, by a similar argument performing multiprecision calculations on different systems will disclose the extent to which the results are numerically reliable. Furthermore, such duplicated calculations can effectively eliminate the possibility that a significant hardware or compiler error occurred in either calculation.

In addition to running the program on different computer systems, the numerical stability of these calculations was monitored by printing at each iteration the minimum absolute value and the minimum difference from Steps 1 and 2 above. In the normal running of this algorithm, these two values gradually decrease until a relation is recovered, at which time one of them drops precipitously to near machine zero (i.e., about 10^{-6m}). The possibility that an actual relation could be missed

because of a faulty zero test can thus be eliminated. The last measure taken to insure the reliability of these results was to run the final versions of the programs on numerous test cases, including several where actual relations are known to exist. In every case where an actual relation existed, the programs either recovered the relation or else exhausted precision before the relation bound exceeded the norm of the actual relation. In each case where precision was exhausted before the norm of an actual relation was achieved, repeating the test run with increased precision successfully recovered the desired relation.

5. Computational Results. The results of these calculations are listed in Table 1. Vectors of the type $(1, \alpha, \alpha^2, \ldots, \alpha^{n-1})$ in this list are attempts to discover algebraic numbers of degree $n - 1$. Several of the results in the table are not related to polynomial relations. Three consist entirely of natural logarithms of various constants. If a relation had been found between these logarithms, then a multiplicative relation would have existed between the arguments of the logarithm calculations.

The bounds listed are the minimum Euclidean norms of any integer relation that could be satisfied by the n-long x-vectors in the list, based on both the Cray-2 and IRIS 4D calculations. In each of these cases, the bounds and other information in the output of the Cray-2 and IRIS runs were identical except near the end, where numerical differences began to alter such aspects of the calculations as the sorted order of the y vector. The bounds listed in the table are the common bounds obtained by both programs up to the point where divergence occurred. After divergence, the separate programs each calculated higher bounds than those listed in the table. It might be noted that each of these runs terminated by recovering a "relation" when precision had been exhausted. The "relations" produced in such cases can be dismissed because of their very large norms, and because neither of the two minimum statistics mentioned above dropped precipitously as they should for a real relation.

The results listed in the table for Feigenbaum's constant demonstrate that some results can be obtained with the PSOS algorithm even if the input numbers are only known to a modest level of precision. This result suggests that these techniques might be applied to studying empirical constants, such as the fundamental constants of physics [7]. Unfortunately, it appears that physical constants are not yet known to sufficient precision to be able to yield strong bounds with the PSOS algorithm.

In fact, there is an interesting history of claims of relations between certain fundamental constants of physics and mathematical constants. As early as 1957, I. J. Good [14] noted that the proton-to-electron mass ratio $M = 1836.15152$ was close to $6\pi^5$. When the PSOS algorithm is applied to the vector $(\log M, \log 2, \log 3, \log \pi)$, indeed Good's relation is recovered by the algorithm as an intermediate result. Unfortunately, M is now known precisely enough to rule out Good's formula. Similarly, A. Wyler noted in 1969 [18] that the fine structure constant $\alpha^{-1} = 137.03604$ was close to $32\pi^3/9 \cdot (15/2\pi)^{1/4}$. When the PSOS algorithm is applied to the vector $(\log \alpha, \log 2, \log 3, \log 5, \log \pi)$, Wyler's relation is recovered as an intermediate result. In this case, however, the PSOS algorithm produces other relations of comparable complexity with even better accuracy. One of these is $\alpha^{-5} = 150\pi(6^5/5^2\pi^3)^8$.

TABLE 1
Computational Results Using the PSOS *Algorithm*

x Vector	n	Digits	Bound
$1, \gamma, \gamma^2, \ldots, \gamma^{n-1}$	9	768	9.2559×10^{36}
	13	768	1.0417×10^{11}
$1, \log\gamma, \log^2\gamma, \ldots, \log^{n-1}\gamma$	9	768	2.5175×10^{37}
	13	768	1.9622×10^{11}
$1, \log\pi, \log^2\pi, \ldots, \log^{n-1}\pi$	9	768	2.3232×10^{37}
	13	768	1.0064×10^{09}
$1, \rho_1, \rho_1^2, \ldots, \rho_1^{n-1}$	9	768	9.6055×10^{43}
	13	768	4.7194×10^{14}
$1, \log\rho_1, \log^2\rho_1, \ldots, \log^{n-1}\rho_1$	9	768	2.5332×10^{45}
	13	768	1.2413×10^{14}
$\rho_1, \rho_2, \ldots, \rho_n$	8	378	5.1278×10^{24}
$\log\rho_1, \log\rho_2, \ldots, \log\rho_n$	8	378	3.6408×10^{25}
$1, \varsigma(3), \varsigma^2(3), \ldots, \varsigma^{n-1}(3)$	9	768	2.0600×10^{47}
	13	768	5.3133×10^{11}
$1, \log\varsigma(3), \log^2\varsigma(3), \ldots, \log^{n-1}\varsigma(3)$	9	768	2.3691×10^{37}
	13	768	3.9526×10^{10}
$1, \pi, \varsigma(3), \gamma, \log(2), \pi^2, \pi^3,$ $\gamma^2, \gamma^3, \pi\gamma, \pi^2\gamma, \pi\gamma^2$	12	768	7.3887×10^{13}
$1, \log\varsigma(3), \log\pi, \log 2, \log\gamma,$ $\log\log 2, \log 3, \log 5, \log 7$	9	768	2.4536×10^{37}
$1, F, F^2, \ldots, F^{n-1}$	3	15	9.6681×10^{03}
	4	15	5.3105×10^{02}
$1, \log F, \log^2 F, \ldots, \log^{n-1} F$	3	15	6.0337×10^{03}
	4	15	3.8994×10^{02}
$\log F, \log\pi, \log 2, \log 3$	4	15	2.2978×10^{02}

Symbols:
γ: Euler's constant
ρ_k: Imaginary part of the kth zero of Riemann's zeta function
$\varsigma(3)$: Riemann's zeta function evaluated at 3
F: Feigenbaum's constant

Indeed, these results indicate that considerable caution should be employed in any attempt to find relationships involving empirical constants using purely numerical techniques.

The computer runs cited in the table provide an interesting comparison in performance between one of the world's most powerful supercomputers and a personal

workstation. Comparing the polynomial cases for $n = 9$, the average CPU time for the Cray-2 was 462 seconds (on one processor), corresponding to a performance rate of approximately 43 million floating-point operations per second (MFLOPS). The IRIS 4D runs on these same cases required an average of 31,700 seconds, or about 0.62 MFLOPS. The ratio of these performance rates is approximately 69.

Numerical Aerodynamic Simulation Systems Division
NASA Ames Research Center
Moffett Field, California 94035
E-mail: dbailey@orville.nas.nasa.gov

Supercomputing Research Center
17100 Science Drive
Bowie, Maryland 20715

1. D. H. BAILEY, "The computation of π to 29,360,000 decimal digits using Borweins' quartically convergent algorithm," *Math. Comp.*, v. 50, 1988, pp. 283–296.

2. D. H. BAILEY, "Numerical results on the transcendence of constants involving π, e, and Euler's constant," *Math. Comp.*, v. 50, 1988, pp. 275–281.

3. D. H. BAILEY, "A high performance FFT algorithm for vector supercomputers," *Internat. J. Supercomput. Appl.*, v. 2, 1988, pp. 82–87.

4. J. M. BORWEIN & P. B. BORWEIN, "The arithmetic-geometric mean and fast computation of elementary functions," *SIAM Rev.*, v. 26, 1984, pp. 351-365.

5. J. M. BORWEIN & P. B. BORWEIN, *Pi and the AGM—A Study in Analytic Number Theory and Computation Complexity*, Wiley, New York, 1987.

6. J. M. BORWEIN & P. B. BORWEIN, "Ramanujan and Pi," *Sci. Amer.*, v. 258, 1988, pp. 112–117.

7. J. V. DRAZIL, *Quantities and Units of Measurement: A Dictionary and Handbook*, Mansell Publishing, Ltd., London, 1983.

8. M. J. FEIGENBAUM, "Quantitative universality for a class of nonlinear transformations," *J. Statist. Phys.*, v. 19, 1978, pp. 25–52.

9. H. R. P. FERGUSON & R. W. FORCADE, "Generalization of the Euclidean algorithm for real numbers to all dimensions higher than two," *Bull. Amer. Math. Soc.*, v. 1, 1979, pp. 912–914.

10 H. R. P. FERGUSON, "A short proof of the existence of vector Euclidean algorithms," *Proc. Amer. Math. Soc.*, v. 97, 1986, pp. 8–10.

11. H. R. P. FERGUSON, "A non-inductive GL(n, Z) algorithm that constructs linear relations for n Z-linearly dependent real numbers," *J. Algorithms*, v. 8, 1987, pp. 131–145.

12. H. R. P. FERGUSON, *A New Integral Relation Finding Algorithm Involving Partial Sums of Squares*, Brigham Young University preprint, Sept. 1987.

13. H. R. P. FERGUSON, "A new integral relation finding algorithm involving partial sums of squares and no square roots," *Abstracts Amer. Math. Soc.*, v. 9, 1988, p. 214.

14. I. J. GOOD, "On the masses of the proton, neutron, and hyperons," *J. Roy. Naval Sci. Service*, v. 12, 1957, pp. 82–83.

15. R. KANNAN & L. A. MCGEOCH, *Basis Reduction and Evidence for Transcendence of Certain Numbers*, Springer Lecture Notes in Comput. Sci., vol. 241, 1986, pp. 263–269.

16. J. HASTAD, B. HELFRICH, J. C. LAGARIAS & C. P. SCHNORR, "Polynomial time algorithms for finding integer relations among real numbers," *SIAM J. Comput.* (to appear); A preliminary version has appeared in *Proc. STACS 86*, Springer Lecture Notes in Comput. Sci., vol. 210, 1986, pp. 105–118.

17. A. M. ODLYZKO & H. J. J. TE RIELE, "Disproof of the Mertens conjecture," *J. Reine Angew. Math.*, v. 357, 1985, pp. 138–160.

18. B. ROBERTSON, "Wyler's expression for the fine-structure constant α," *Phys. Rev. Lett.*, v. 27, 1971, pp. 1545–1547.

19. P. N. SWARZTRAUBER, "Multiprocessor FFTs," *Parallel Comput.*, v. 5, 1987, pp. 197–210.