Definite integration and summation are #P-hard

Leonid Gurvits & Warren D. Smith*
WDSmith@fastmail.fm, gurvits@lanl.gov

3 Sept 1998; revised for publication Jan 2005

Abstract — **We show that the common symbolic manipulation tasks of computing multiple partial derivatives, definite integration, and definite summation, are #P-hard, i.e., at least as hard as counting the accepting input strings for any Turing machine that halts in polynomial time. (The "multiple partial derivatives" part was previously known.)**

Keywords — multiple partial derivatives, definite integration, definite summation, symbolic manipulation tasks, #P-completeness, permanents, Ryser's formula.

1 Permanents, averaging, and partial differentiation

LET A be an $n \times n$ matrix

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}. \tag{1}$$

The "permanent" of A is defined by

$$\mathrm{per}A = \sum_{\sigma} a_{1,\sigma(1)} a_{2,\sigma(2)} \cdots a_{n,\sigma(n)} \tag{2}$$

where the sum is over the $n!$ permutations σ of $\{1, 2, ..., n\}$.

L.G.Valiant [13] showed that computing the permanent of a matrix whose elements lay in the set $\{-1, 0, 1, 2, 3\}$ was #P-complete.

We first note that

$$\mathrm{per}A = \begin{bmatrix} \text{coeff. of} \\ z_1 z_2 \cdots z_n \text{ in} \end{bmatrix} \prod_{k=1}^{n} (a_{1k} z_1 + a_{2k} z_2 + \cdots + a_{nk} z_n). \tag{3}$$

As a consequence, any "averaging process" on the variables z_1, z_2,..., z_n which serves to "cancel out" all the coefficients in the monomial expansion of the polynomial on the right of (EQ 3), *except* for the one coefficient we want, will lead to a formula for the permanent. Special cases of this observation have been discovered (and rediscovered) previously. For example, "Ryser's formula" [11]

$$\mathrm{per}A = \frac{1}{2^n} \sum_{q_1=0}^{1} \sum_{q_2=0}^{1} \cdots \sum_{q_n=0}^{1} \prod_{k=1}^{n} \left(\sum_{j=1}^{n} (-1)^{q_j + q_k} a_{jk} \right) \tag{4}$$

involving a sum over 2^n binary words $q_1 q_2 ... q_n$, arises by considering one such averaging process. This sum may be evaluated in $O(n2^n)$ arithmetic operations by traversing the binary words in a "Gray code" order in which any word differs from its predecessor at exactly one bit.

The point of the Gray code is that each of the n terms in the product may be updated, after changing one bit, in $O(1)$ steps. Note that in this algorithm, unlike the previous one, the storage requirements are only linear.

Incidentally, a further factor 2 saving in work may be garnered by only using the 2^{n-1} words with one of the bits *fixed.* The only monomials this averaging process could get wrong are ones in which z_1 (assuming the bit with

*Work done while both authors were at the now-defunct NEC Research Institute, 4 Independence Way, Princeton NJ 08540 USA. Gurvits is now at CCS-3 Modeling, Algorithms, & Informatics Group, Computer & Computational Sciences Division, Los Alamos National Laboratory. Non-electronic mail to Smith at 21 Shore Oaks Drive, Stony Brook NY 11790.

index 1 is the fixed bit) appears to an odd degree and all the other z_j appear raised to an even power. (Remember that $z_j^2 = 1$ if $z_j = \pm 1$.) But, such monomials cannot arise.

Other averaging processes based on multiple uses of Cauchy's residue theorem, integration over spheres or radially symmetric Gaussians, summation over roots of unity forming a regular p-gon (the above was $p = 2$), integrals over tori, and so forth could also be used, leading to a large number of possible permanent formulae. One useful trick when devising such formulae is to let $z_j = \exp(iq_j)$ and constrain q_n to equal $-\sum_{j=1}^{n-1} q_j$ inside the averaging integral or sum, which causes $\prod_{j=1}^{n} \exp(iq_j)$ to average to 1 and $\prod_{j=1}^{n} \exp(iq_j)^{m_j}$ to average to 0 if the m_j are unequal – precisely the desired effect. An alternative trick is to multiply the product in (EQ 3) by $z_1 z_2 \cdots z_n$ and then average $\vec{z}$ over any centrally symmetric probability distribution in $\mathbf{R}^n$. The term we want will average out to something positive since $(z_1 z_2 \cdots z_n)^2$ is nonnegative, but the other terms will average to 0 because at least one power of at least one z_j must be odd.

A different method of isolating the one coefficient we want is multiple partial differentiation. This leads to:

Theorem 1 (Previously known)
If $a_{jk} \in \{-1, 0, 1, 2, 3\}$, then evaluating the multiple partial derivative

$$\frac{\partial^n}{\partial z_1 \, \partial z_2 \cdots \partial z_n} \prod_{k=1}^{n} (a_{1k} z_1 + a_{2k} z_2 + \cdots + a_{nk} z_n)\big|_{\vec{z}=\vec{0}}. \tag{5}$$

at the point $z_1 = z_2 = \cdots = z_n = 0$ is #P-complete.

Proof. This is per$A$. □

2 The complexity of 1D definite integration

Let p be an integer with $|p| \geq 2$. Then we may express the permanent of an $n \times n$ matrix A as a 1D definite integral:

$$\mathrm{per} A = \frac{1}{2\pi} \int_0^{2\pi} \prod_{j=1}^{n} \left(\sum_{k=1}^{n-1} a_{jk} \exp(ip^{k-1}\theta) + a_{jn} \exp(-i\frac{p^n - 1}{p-1}\theta) \right) \mathrm{d}\theta. \tag{6}$$

The point is that this integral averages all the terms in the generating function (EQ 3) to 0, except for the one we want.

Observe: to make everything real, we can replace every occurence of $\exp(i\alpha)$ by its real part $\cos\alpha$ and use $\frac{1}{\pi}\int_0^{\pi}$ instead of $\frac{1}{2\pi}\int_0^{2\pi}$.

To avoid[1] the use of transcendental functions (e.g. cos) or of the transcendental number π as a limit of the integral, then by using the substitution $\theta = \arctan(2t)$, $\mathrm{d}\theta = 2/(1+4t^2)\mathrm{d}t$, $\cos\theta = (1-t^2)/(1+t^2)$, $\sin\theta = 2t/(1+t^2)$, $\exp(iq\theta) = [(1+it)/(1-it)]^q$ we may instead write the integral as a rational function of t integrated from $-\infty$ to $+\infty$ (for an appropriate choice of the branch of arctan). To avoid integrals over infinite intervals, by a further rational change of variables such as $t = s/(1-s^2)$, $\mathrm{d}t = (1+s^2)/(1-s^2)^2\mathrm{d}s$ we may transform to the finite range $-1 < s < 1$.

So, we conclude

Theorem 2 *Given a rational function of s (with Gaussian-integer coefficients and integer degrees) determining its integral over the range $-1 < s < 1$ is #P-hard. This is true even if the rational function integrand is expressed as a "straight line code" program, and even if we promise that the value is a positive integer only a polynomial[2] number of bits long, and even if we further promise that running this code at any rational point s in $(-1, 1)$ will run in polynomial time.* □

On the other hand, the promise problem[3] in the theorem is in fact soluble in polynomial time with the aid of an oracle for #P. Hence

[1] The reason why we go to extra trouble to make the theorem hold for *rational* functions is that they are the "simplest" class of functions that are not trivially integrable. Our point is that if rational functions are hard to integrate, presumably most everything else is at least as hard to integrate.

[2] "Polynomial" in the number of statements in the input straight-line program, which is a list of statements of the form $a \leftarrow b \& c$ where a, b, c are variable names or bounded-integer constants and & is one of the four operators $\{+, -, *, /\}$.

[3] "Promise problem" is a standard technical term introduced in [4] and further developed in [5]. A famous paper about a promise problem is [12]. It refers to the notion of allowing the solution procedures for some class of problems only to be required to work on the *sub*class of those problems which obey some "promise," e.g. that the output bit-length will be short. Obviously, solving the full class of problems is at least as hard as solving the promise-subclass.

Theorem 3 *The promise problem of the preceding theorem is #P-complete.*

Proof: We need only to show that it is in #P. We take advantage of the fact that integrating a Fourier mode around the unit circle (e.g. $\int_0^{2\pi} \sin(m\theta)\mathrm{d}\theta = 0$) may be done by summation at the vertices of a regular polygon, if the polygon has n vertices with $n \geq m$. In other words, for integrals of this sort, the trapezoid rule is exact if there are enough trapezoids.[4] Hence the integral (EQ 6) is expressible exactly as a finite sum at the vertices of a regular p^n-gon

$$\mathrm{per}A = \frac{1}{2\pi p^n} \sum_{m=0}^{p^n-1} \prod_{j=1}^{n} \left(\sum_{k=1}^{n-1} a_{jk} \exp(2\pi i m p^{k-1-n}) + a_{jn} \exp(-2\pi i m \frac{p^n-1}{(p-1)p^n}) \right). \tag{7}$$

We can ask our #P oracle to compute the number of summands at which bit b of the summand is 1, for all relevant b (also using bits on the fractional side of the decimal point) there being only a polynomial(n) number of bits that could matter (due to the promises). From these counts we may deduce the value of the sum (i.e. integral) accurate to the nearest integer (i.e. exactly). □

Again, since the integral of EQ 6 is expressible exactly as a finite sum, we conclude

Theorem 4 *Doing the definite integral of (EQ 6) (and its equivalent versions discussed above) is #P-complete; so is its summation version with p^n terms to be summed.*

The weaker result that determining whether an integral of the form

$$\int_0^{2\pi} \prod_{j-1}^{n} \cos(a_j\theta)\mathrm{d}\theta \tag{8}$$

equals zero is NP-hard, was previously shown by Plaisted [10]. Here the a_j are integers represented in binary. Plaisted's problem is soluble in "pseudo-polynomial time," i.e. in polynomial time if the a_j are input in unary.

In contrast, our integrals and sums remain #P-complete even if our a_{jk} are input in unary. However, we have other exponentially large integers (of order p^n) in our expression (EQ 6), which certainly couldn't be input in unary. These n numbers do have very simple radix-p representations, however. To be precise, the n numbers in question are $1, p, p^2, \ldots, p^{n-1}$ and $-(p^n-1)/(p-1)$. If these were written *symbolically* (as we've just written them) then *all* input numbers could be unary. Also, if these numbers were computed using straight-line code (or if exponentials y^k for our large integers k were computed using straight-line code) then the straight-line code could "handle the binary" and then only small numbers would need to occur explicitly, all of which could be input in unary. (And of course we recommend the simplest choice $p = 2$ for p.) Incidentally, these n numbers could instead have been chosen to be *any* set of n integers summing to 0 and such that the only way to pick n elements (with replacement) from this set, such that the selected multiset sums to 0, is to pick the entire set.

It remains possible that some other class of integrals than ours is even more difficult than #P-complete.[5]

3 How hard is it to approximate the permanent?

Consider the problem of approximating the number of satisfying assignments of a boolean formula (the approximate "#SAT problem"). This is "APX-complete" [2].

[4]This well known fact underlies the "Discrete Fourier Transform" (which is the same thing as the "fast fourier transform" FFT, except done slower), and the Euler-Maclaurin summation formula ([1]§23.1.30), and the discrete (i.e. sum) orthogonality relation obeyed by the Chebyshev polynomials ([1]§22.17). Although the FFT usually is attributed to J.W.Cooley and J.W.Tukey in 1965, it in fact had been published earlier by G.C.Danielson and C.Lanczos in 1942, and (much) earlier by C.F.Gauss in 1805, who indeed apparently even used it to fit the trajectories of the asteroids Pallas and Juno. For these historical facts see [6]. The orthogonality relation alone (i.e. the DFT, as opposed to the FFT – which suffices for our purposes) probably was known to J.B.J.Fourier (1768-1830) himself even earlier. A trivial proof is as follows: Because the m roots of the polynomial $z^m - 1$ lie at the vertices of a regular m-gon, by considering the fact that the coefficients of z^{m-k} in this polynomial are exactly 0 for each $k = 1, 2, \ldots, m-1$ we see successively that the sum of the kth powers of those roots plainly is exactly 0 for each $k = 1, 2, \ldots, m-1$. Another trivial proof is simply to observe the rotational symmetry: the kth powers of the polygon vertices are rotationally symmetrically distributed about 0 and hence sum to 0, except if k is a multiple of m, when they all are 1 and hence sum to m.

[5]In fact, the integral $\int_0^3 x^{N-1}\mathrm{d}x = 3^N/N$ has a value which requires exponentially many (as a function of the number of bits in N) bits to express in binary. This sort of triviality is best abolished with the aid of some sort of promise, which is in fact what we did. A slightly different approach would be, with the aid of the promise that the integral's value is a *rational number*, to ask for the value of that rational *modulo* some small prime. For example, it is trivial for a Turing machine to evaluate $3^N/N \bmod 5$ in a number of steps polynomial in the bit length of N (use modular arithmetic and "binary powering"). Furthermore, the value of a definite integral of a rational function between rational bounds is always a short rational linear combination of logarithms of rational numbers, and this fact, plus modular evaluation of the rationals, could be used to avoid any need for a promise *at all*. In contrast, prime-modular evaluation of integer permanents also is #P-complete [14], so that the comparable question about the integrals in theorem 2 is #P-complete.

We can build a boolean formula with n inputs such that the left $n/2$ inputs have exactly 0 or 1 satisfying assignments and such that the right half of the circuit is always satisfied (the full circuit ANDs the left and right halfs). Then the whole circuit will have either 0 or $2^{n/2}$ satisfying assignments. Thus approximating the number of satisfying assignments to within $2^{n/2}$ on an n-input SAT problem is at least as hard as determining whether a SAT problem on $n/2$ inputs has a solution, given the *promise* that it has exactly 1 or 0 solutions.

Valiant [13] showed that #SAT could be reduced in polynomial time to a single #3SAT computation, i.e. (in his notation from [14]) SAT$\leq$!3SAT. Valiant also found count preserving (except for an easily computed factor of proportionality) reductions to show that SAT$\leq$!HAMILTONIAN CIRCUITS and [14] that 3SAT$\leq$!$\{-1, 0, 1, 2, 3\}$PERMANENT.

These problems are complete over an interesting complexity class called "UP" (for promised Unique P). Clearly $P \subseteq UP \subseteq NP$. It is usual to conjecture that $P \neq UP$.

One reason to believe that $P \neq UP$ is that the following problem is in UP, but looks too difficult to be in P:
`PROMISE PROBLEM:` Does a product of two primes have a factor in a given range?
`INSTANCE:` An integer N, promised to be of the form $N = pq$, $p \leq q$ both prime, and an integer interval $[\ell, u]$.
`QUESTION:` Is $\ell \leq p \leq u$?

Even more convincingly, Valiant and Vazirani have shown [17] that RUP=NP, i.e. if one could detect unique solutions to NP problems in randomized polynomial time, then RP=NP.

There is some speculation [3] that $UP \neq NP$, i.e. that the randomness in [17] was essential.

Anyhow, the moral is that

Theorem 5 *There exists $c > 1$ such that it is impossible to approximate permanents with elements in the set $\{-1, 0, 1, 2, 3\}$ to within an additive factor $\pm c^L$ and a multiplicative factor c^L (where L is the input length) in random polynomial time, unless RP=NP.* $\Box$

However, it was recently shown [7] that there is a fully polynomial randomized approximation scheme for approximating permanents of matrices with *nonnegative* binary-integer entries to within any desired constant factor $1 + \epsilon$.

References

[1] M.Abramowitz & I.A.Stegun: Handbook of Mathematical Functions, US Nat'l Bureau of Standards 1964, and Dover reprint.

[2] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi: Complexity and Approximation. Combinatorial Optimization Problems and their Approximability Properties. Springer, Berlin, 1999.

[3] R. Beigel, H. Buhrmann, L. Fortnow: NP might not be as easy as detecting unique solutions, STOC 30 (1998) 203-208.

[4] S.Even & Y.Yacobi: Cryptography and NP-completeness, Proc. Colloq. Automata, Languages, and Programming 7 (1980) 195-207 (Springer LNCS #85).

[5] S.Even, A.S.Selman, Y.Yacobi: The complexity of promise problems with applications to public-key cryptography, Inform. & Control 61,2 (1984) 159-173. Also reviewed by D.S.Johnson in his column in J.Algorithms 6 (1985) 291-305.

[6] M.T. Heideman, D.H. Johnson, C.S. Burrus: Gauss and the history of the fast Fourier transform, IEEE Acoustics, Speech, and Signal Processing Magazine 1,4 (1984) 14-21, and Archive for History of Exact Sciences, 34,3 (1985) 265-277.

[7] Mark Jerrum, Alistair Sinclair, Eric Vigoda: A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries, ACM Sympos. Theory of Computing 33 (2001) 712-721. To appear in J. Assoc. Computing Machinery. I. Bezakova, D. Stefankovic, V. Vazirani, and E. Vigoda have recently claimed (in a preprint) to have reduced the runtime to approximate 0-1 permanents to $O^*(n^7)$ steps from $O^*(N^{26})$, but still do not regard their algorithm as practical.

[8] D.S.Johnson: A catalogue of complexity classes, chapter 2 pages 67-162 in Handbook of theoretical computer science (volume A), MIT Press 1990.

[9] David A. Plaisted: Sparse Complex Polynomials and Polynomial Reducibility, JCSS 14,4 (1977) 210-221.

[10] David A. Plaisted: Some Polynomial and Integer Divisibility problems are NP-Hard, SIAM J. Computing, 7,4 (1978) 458-464.

[11] H.Ryser: Combinatorial mathematics, (Carus math. monographs #14) Wiley 1960.

[12] Amit Sahai, Salil P. Vadhan: A Complete Promise Problem for Statistical Zero-Knowledge IEEE Symposium on Foundations of Computer Science FOCS 38 (1997) 448-457.

[13] Leslie G. Valiant: The complexity of enumeration and reliability problems, SIAM J. Comput. 8,3 (1979) 410-421.

[14] Leslie G. Valiant: The complexity of computing the permanent, Theor. Comput. Sci. 8,2 (1979) 189-201.

[15] Leslie G. Valiant: Completeness classes in algebra, STOC 11 (1979) 259-261.

[16] Leslie G. Valiant: Reducibility by algebraic projections, pp. 365-380 in Logic and Algorithmic: internat. sympos. held in honour of Ernst Specker, Monographies de l'Enseignement Mathematique 30 (1982).

[17] Leslie G. Valiant & Vijay V. Vazirani: NP is as easy as detecting unique solutions, Theor. Comput. Sci. 47,1 (1986) 85-93.