JOURNAL OF APPROXIMATION THEORY **50**, 193–199 (1987)

Reduced Complexity Evaluation of Hypergeometric Functions

PETER B. BORWEIN*

Mathematics Department, Dalhousie University, Halifax, Nova Scotia, B3H 4H8, Canada

Communicated by Paul G. Nevai

Received August 6, 1984

DEDICATED TO THE MEMORY OF GÉZA FREUD

Fast Fourier transform-like techniques can be employed to reduce the complexity of the evaluation of standard approximations to hypergeometric functions and the gamma function. This leads to algorithms that provide n digits of these functions for $O(\sqrt{n}\,(\log n)^2)$ arithmetic operations. The usual methods require $O(n)$ operations for comparable accuracy.

1. INTRODUCTION

Most standard methods for evaluating the various elementary and special functions provide $O(n)$ digits for n arithmetic operations. This is the case for evaluating, by Horner's rule, the partial sums of the Taylor expansion of $\log(1-x)$ (or virtually any other non-entire analytic function). The Taylor polynomials are locally optimal polynomial approximants on discs in the comlex plane and if one merely measures rate of convergence against the degree of the approximant, then, for most familiar functions, there is little gain to be made in pursuing the matter further. However, there are situations where high-degree approximants can be generated using relatively few arithmetic operations.

For example, Newton's method for calculating $\sqrt{x}$ is the iteration

$$x_{n+1} := \tfrac{1}{2}\,(x_n + x/x_n), \qquad x_0 = 1.$$

The $(n+1)$st iterate, which is in fact the $(2^n, 2^n - 1)$ Padé approximant to $\sqrt{x}$ at 1, is a rational function in x of degree 2^n which provides roughly 2^n digits of $\sqrt{x}$ and can be evaluated for only $O(n)$ operations.

* Research partially supported by NSERC of Canada.

0021-9045/87 $3.00

Various reduced complexity algorithms for log and exp are presented in [3, 5–8]. These are based on the arithmetic-geometric mean iteration for elliptic integrals. They can be used to construct rational approximations that provide n digits of log for $O((\log n)^2)$ arithmetic operations.

These approaches provide a remarkable improvement in complexity—n operations are reduced to $(\log n)^k$ operations. The functions for which similar reduction is known include the Jacobian elliptic functions, the elementary transcendental functions, and algebraic functions. Unfortunately, no comparable methods are known for most of the other special functions.

Our primary goal is to show that the hypergeometric functions and the gamma function can be calculated to n digit accuracy in $O(\sqrt{n}\,(\log n)^2)$ arithmetic operations. These algorithms based on FFT related methods, while far slower than the $O((\log n)^2)$ algorithms for log, are nonetheless, from a complexity point of view, a considerable improvement over currently employed methods.

2. Main Results

We need the following FFT related lemmas. Proofs may be found in any of [2, 4, 9].

Lemma 1. *Given the coefficients of any two polynomials of degree n the coefficients of their product can be calculated using $O(n \log n)$ arithmetic operations.*

Lemma 2. *Any polynomial of degree n (given by either its roots or its coefficients) can be evaluated at any n distinct points using $O(n(\log n)^2)$ arithmetic operations.*

We say that a function f is *hypergeometric* if

$$f = \sum_{n=0}^{\infty} a_n z^n \qquad \text{where } a_n/a_{n-1} = R(n) \text{ and } a_{-1} = 1$$

for some fixed rational function R. This is a more general definition than is sometimes employed. We require that f has non-zero radius of convergence. We additionally assume that R has rational coefficients. This is to ensure the easy evaluation of R. We could for most purposes replace this restriction by the assumption that the coefficients of R are precomputed to desired accuracy. We observe that $\log(1-z)$, $\exp(z)$, and $\sin\sqrt{z}$ are all hypergeometric by this definition as are Gaussian hypergeometric series ($F(a, b; c; z)$ with a, b, c rational) and Bessel functions of integer order [1].

THEOREM 1. *The first n digits of $f(z)$ for any hypergeometric function f (with the additional assumption above) can be calculated using $O(\sqrt{n}\,(\log n)^2)$ arithmetic operations.*

Before proving Theorem 1 some comments are in order. By computing n digits of $f(z)$ we mean estimating $f(z)$ with error bounded by 10^{-n}. The order estimate in Theorem 1 is independent of z provided we restrict ourselves to compact regions in the interior of the region of convergence of the expansion of f; it is not, however, independent of f.

The arithmetic operations in question are addition, substraction, multiplication, and division all performed to precision $O(n)$. Thus, the bit complexity (the number of single digit operations) is $O_{\mathrm{BIT}}(\sqrt{n}\,(\log n)^2\, M(n))$ where $M(n)$ is the bit complexity of multiplying two n digit integers together. With a fast multiplication this produces $O_{\mathrm{BIT}}(n^{3/2}(\log n)^4)$ algorithms for any hypergeometric function. (See [2], [4], or [9].)

Proof of Theorem. We have

$$f(z) = \sum_{k=0}^{\infty} a_k z^k, \qquad |z| < \delta$$

where

$$a_k = \prod_{i=0}^{k} R(i).$$

We assume that $|z| < \varepsilon < \delta$ so that for large n

$$S_n(z) := \prod_{i=0}^{n^2-1} a_i z^i$$

differs from f by $O(\rho^{n^2})$ for some fixed $\rho < 1$ independently of z. It now suffices to show that we can calculate S_n in $O(n(\log n)^2)$ arithmetic steps. This constitutes the remainder of the proof. Let

$$T(k) := \prod_{i=0}^{k-1} R(i)$$

and let

$$Q(k) := R(k) + R(k)R(k+1)z + \cdots + [R(k)R(k+1)\cdots R(k+n-1)]z^{n-1}.$$

Then $a_{k-1} = T(k)$ and

$$S_n = z^0 Q(O)T(O) + z^n Q(n)T(n) + \cdots + z^{n(n-1)}Q(n(n-1))T(n(n-1))$$

and we have reduced the evaluation of S_n to the evaluation of a simpler function $Q \cdot T$ at n points.

First observe that $z^0, z^n, z^{2n}, \cdots, z^{n(n-1)}$ can all be calculated using $O(n)$ steps.

The second observation is that $T(0), T(n), \cdots, T(n(n-1))$ can all be calculated using $O(n(\log n)^2)$ steps. To see this consider the rational function V of degree $n \cdot$ degree (R) defined by

$$V(x) := \prod_{i=0}^{n-1} R(i+x).$$

Then

$$T(kn) = V(0) \cdot V(n) \cdots V((k-1)n)$$

and by Lemma 2, given the coefficients of $V(x)$, the $V(i)$ can be evaluated at $0, n, \ldots, (n-1)n$ in $O(n \log n)^2)$ steps. That the coefficients of $V(x)$ can be calculated with similar dispatch is straightforward. Let $C(m)$ be the number of operations required to evaluate the coefficients of a product of m terms $R(j+1+x)R(j+2+x) \cdots R(j+m+x)$. Then by dividing the product into two products of half the size and recombining the halves using a fast polynomial multiplication, as in Lemma 1, we see that

$$C(2m) \leqslant 2C(m) + O(m \log m)$$

whence

$$C(m) = O(m(\log m)^2),$$

and we can expand $V(x)$ as required. Thus, we can calculate the $V(i)$ and hence the $T(kn)$ with operational complexity $O(n(\log n)^2)$.

The third part necessitates showing that $Q(0), Q(n), \ldots, Q(n(n-1))$ can all be calculated with complexity $O(n(\log n)^2)$. Note that Q is a rational function of k of degree bounded by $n \cdot$ degree (R). We observe that the evaluation of the coefficients of 0 can be split recursively since

$$R(k) + R(k)R(k+1)z + \cdots + [R(k)R(k+1) \cdots R(k+2n-1)]z^{2n-1}$$
$$= \left(z^n \prod_{i=k}^{k+n-1} R(i)\right) \cdot (R(k+n) + \cdots$$
$$+ [R(k+n)R(k+n+1) \cdots R(k+2n-1)]z^{n-1})$$
$$+ (R(k) + \cdots + [R(k)(k+1) \cdots R(k+n-1)]z^{n-1}).$$

If $D(m)$ is the number of operations required to evaluate the coefficients of a sum of running products of the above type of length m then

$$D(2m) \leqslant 2\mathrm{D}(m) + O(m \log m).$$

The final term comes from the multiplications and additions needed to recombine the two pieces. Note that we can compute all the $\Pi R(i)$ terms in $O(n(\log n)^2)$ operations. The recursive inequality for D solves as

$$D(m) = O(m(\log m)^2)$$

and by Lemma 2 we can calculate all the $Q(kn)$ with complexity $O(n(\log n)^2)$.

The three parts above combine to finish the proof by showing that S_n can be evaluated using $O(n(\log n)^2)$ operations. ∎

In a similar vein we have the following theorem for the gamma function.

THEOREM 2. *The first n digits of* $\Gamma(x)$ *can be calculated using* $O(\sqrt{n}\,(\log n)^2)$ *arithmetic operations.*

Proof. We can construct a uniform estimate for $s \in [1, 2]$. From

$$\Gamma(s) := \int_0^\infty e^{-t} t^{s-1}\, dt$$

we have on splitting the integral at N and expanding

$$\Gamma(s) = N^s \sum_{k=0}^{\infty} \frac{(-1)^k}{k!} \frac{N^k}{s+k} + \int_N^\infty e^{-t} t^{s-1}\, dt.$$

One easily derives for $s \in [1, 2]$ and $n := 6N$ that if

$$g_n(s) := N^s \sum_{k=0}^{6N} \frac{(-1)^k N^k}{k!(s+k)}$$

then

$$|\Gamma(s) - g_n(s)| \leqslant 2Ne^{-N}.$$

The result now follows basically as in Theorem 1. We observe that the incomplete gamma function

$$\sum_{k=0}^{\infty} \frac{(-1)^k x^k}{k!(s+k)}$$

is a hypergeometric function in x and is amenable to a similar analysis as that provided by Theorem 1 (with some extra care to ensure the uniformity in s). The approximation $g_n(s)$ requires the computation of N^s which, by Theorem 1, can be performed in $O(\sqrt{n}\,(\log n)^2)$ operations.

3. Comments

(A) For rational z within the region of convergence of a hypergeometric f we can calculate $f(z)$ with bit complexity

$$O_{\text{BIT}}((\log n)^2 M(n)).$$

This bound is no longer independent of z. (We must also make some minimal assumptions about the multiplication underlying $M(n)$. It sufficies, for example, to assume M increasing and $M(2n) \geqslant 2M(n)$). This estimate is achieved by taking advantage of the observation that most of the operations required to calculate $Q(0)$ recursively, as in Theorem 1, can be performed to a reduced precision. If the rational function R associated with a hypergeometric f has a zero at infinity then the partial sums converge rapidly enough to ensure an

$$O_{\text{BIT}}((\log n) M(n))$$

bit complexity method for evaluating f at a rational z. This will be the case exactly when f is entire. See [7] for further details concerning calculating exp in this fashion.

(B) Let

$$s_n(z) := \sum_{k=0}^{n} \frac{z^k}{k!}.$$

For $|z| \leqslant 1$

$$|e^z - (s_n(z/2^n))^{2^n}| \leqslant \frac{1}{n!\, 2^{n^2}}.$$

This already provides a method of evaluating exp using only $O(\sqrt{n})$ arithmetic operations. We can couple this with an FFT reduced evaluation of $s_n(z/2^n)$, as in Theorem 1, to provide an algorithm that calculates n digits of exp using

$$O(n^{1/3}(\log n)^2)$$

arithmetic operations. Similar algorithms exist for log and the trig functions [6, 7]. Of course, very much faster algorithms for the elementary functions based on the transformations of elliptic integrals are known [3, 5–8].

(C) These algorithms are fairly complicated and are of little practical utility even for the not very practical problem of unlimited precision evaluation of special functions. They do, however, offer further evidence that traditional measures of efficiency of approximation are perhaps not always the most appropriate.

References

1. M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions, National Bureau of Standards, Washington, D. C., 1970.
2. A. Aho, J. Hopcroft and J. Ullman, "The design and Analysis of Computer Algorithms," Addison-Wesley, Reading, Mass., 1974.
3. M. Beeler, R. W. Gosper and R. Scroeppel, "Hakmem," MIT Artificial Intelligence Lab., MIT, Cambridge, Mass., 1974.
4. A. Borodin and I. Munroe, "The Computational Complexity of Algebraic and Numeric Problems," Amer. Elsevier, New York, 1975.
5. J. M. Borwein and P. B. Borwein, The arithmetic-geometric mean and fast computation of elementary functions, *SIAM Rev.* **26** (1984), 351–366.
6. J. M. Borwein and P. B. Borwein, Reduced complexity calculation of log, to appear.
7. R. P. Brent, The complexity of multiple-precision arithmetic, *in* "Proceedings, Seminar on Complexity of Computational Problem Solving," Queensland Univ. Press, Brisbane, 1975.
8. R. P. Brent, Fast multiple-precision evaluation of elementary functions, *J. Assoc. Comput. Mach.* **23** (1976), 242–251.
9. D. Knuth, "The Art of Computer Programming," Vol. 2, "Seminumerical Algorithms," Addison-Wesley, Reading, Mass., 1969.