JOURNAL OF COMPUTATIONAL PHYSICS 15, 413–420 (1974)

Note

Computation of a Class of Functions Useful in the Phase-Integral Approximation. II. Performance of TRIGMAN, SYMBAL and LISP

A previous paper [1] has set out the details of the phase-integral approximation and the symbolic computing (SC) of its special functions Y_{2n} through the 20th. order. Here we describe the characteristics of the computation itself, and make recommendations about how to present the results of any similar computation in the future. We consider this issue because, although the Y-function computation has aroused much interest [2]–[8] among designers of SC systems, comparison of the performances of different systems are difficult for the nonspecialist in the absence of agreed conventions of presentation. In particular, no two trials of different systems have been run for Refs. [2]–[8] in the same place on the same computer. We present comparisons of three systems which we have been able to run on a CDC 6600 computer (while noting that Sundblad [9] has recently completed an excellent two-computer study of several systems which reaches some conclusions similar to our own). We have chosen them to reflect three different approaches to SC which are implicit in the range of available systems and languages.

The phase-integral approximation [10] solves the differential equation

$$\frac{d^2\psi}{dz^2} + \frac{Q^2(z)}{\lambda^2}\psi = 0 \tag{1}$$

to any desired accuracy in the small parameter λ. Ref. [1] gives the details of the computation of each Y_{2n} from the recurrence relation which follows the substitution of Eqs. (2) and (3) into Eq. (1), where the two independent solutions of Eq. (1) have the form

$$q^{-1/2}(z) \exp\left(\pm i \int q(z)\, dz\right), \tag{2}$$

and

$$q(z) = \frac{Q(z)}{\lambda} \sum_{n=0}^{N} Y_{2n}(z). \tag{3}$$

In terms of

$$\epsilon_0 = \left(\frac{\lambda}{Q(z)}\right)^{3/2} \frac{d^2}{dz^2} \left(\frac{\lambda}{Q(z)}\right)^{1/2} \tag{4}$$

and

$$\epsilon_m = \left(\frac{\lambda}{Q(z)}\frac{d}{dz}\right)^m \epsilon_0 , \tag{5}$$

some of the low-order expressions are

$$Y_2 = \tfrac{1}{2}\epsilon_0$$
$$Y_4 = -\tfrac{1}{8}(\epsilon_0{}^2 + \epsilon_2)$$
$$Y_6 = \tfrac{1}{32}(2\epsilon_0{}^3 + 6\epsilon_0\epsilon_2 + 5\epsilon_1{}^2 + \epsilon_4)$$

Values up to Y_{20} have been published [1], but there are reports of computations through Y_{22} by Horowitz and Siegel [8], Y_{28} by Bourne [3], and Y_{32} by Brown and Hall [2].

The lengths $L(2n)$ of expressions for Y_{2n} grow rapidly as n increases. For the earlier paper [1], a symbolic program which fitted polynomials to difference tables produced an answer for the lengths (based on results through $n = 10$) in which the leading term was $\frac{2}{3}n^3$. However, this was quoted only as a programming achievement, and labelled "lower bound" because it did not display the exponential (rather than power-law) behaviour which has been characteristic of symbolic computations in physics. An exact determination of the length is not difficult: from Eqs. (4) and (5) it follows that ϵ_m is of order λ^{2m+2}, and the order of any term in Y_{2n} must be λ^{2n}, so that what is needed is a measure of all the partitions of $2n$ over the exponents, omitting 1 from the partitions because there is no factor that contributes exactly one power of λ. But the number of these latter partitions must be just $p(2n-1)$, because each such partition of $2n$ to be excluded can be formed if we add 1 to some partition of $2n-1$. Hence

$$L(2n) = p(2n) - p(2n-1) \tag{6}$$

An approximation to L good for all but small n is

$$L(2n) \approx (1/8\sqrt{3})\,[(c\sqrt{n}-1)/n(2n-1)]\,e^{2c\sqrt{n}}, \qquad c = \pi/\sqrt{3}. \tag{7}$$

The leading asymptotic value of $L(2n)$ quoted by Brown and Hall [2] is contained in Eq. (7). Bourne's discussion [3] of the problem implies that he is aware of Eqs. (6) and (7). Brown and Hall mention "about 1500 terms" in storage after the computation of Y_{32}: Eq. (6) gives 1507.

With these properties for the Y functions, it is necessary to be careful before selecting a method for practical computations to high values of n. Here, the history of improvements in algorithms may be of interest. The original recurrence [10] for Y_{2n} contained rational-number coefficients and a quadrilinear sum in the Y_{2m} $(m < n)$, both of which made strong demands on computing time and storage.

Sundblad [7] rewrote the recurrence to remove numerical denominators, and found an immediate improvement of 35 % in speed. Finally, Bourne [3] used two interlinked recurrences in which only bilinear factors occurred, and showed that the saving in time was about 52 %.

Based on our experiences, we now make several suggestions about the presentation of results for benchmark SC problem. We begin with two global suggestions, of which the first is justified by our comments immediately above.

(1.1) Following the proposal of a problem, there should be a relaxation time during which a fastest algorithm for computation is sought. Then, all systems are enabled to compete with the help of the same algorithm.

(1.2) The runs of all competing systems should be on a computer of the same type. This removes all complications which arise from the simple use of the memory cycle time as a means of normalizing computing times from different machines, as well as the complications based on differences in instruction sets and in the operation of hardware. Now that there are versions of all major SC systems for, e.g., the IBM 370/165 and CDC 6600 computers, the suggestion ceases to be Utopian.

Next, we propose some categories of data for reporting along with the solutions of test problems:

(2.1) The time t_0 taken for a program to run to a predetermined point (e.g. to a given value of N in (3)) is obviously basic to any discussion of performance. We recommend that intermediate times for the compilation or equivalent processing of the program, and for the completion of each order or stage, should also be recorded.

(2.2) Now that many systems are freely available to all comers, it is of importance for a newcomer to know how easy it is to get the best out of any given system. A collection of values of t_0 may not always be helpful for this purpose, because in the past almost all values of t_0 have referred to the performance of programs written by the designers of the systems. One possible way to identify hidden subtleties is to ask a programmer removed from the design of a system to write a program in that system, and to have it run to the same point as in (2.1). The resulting time, t_1, is a valuable piece of information, more so if it represents the average over the work of several programmers.

(2.3) A measure of the time spent in drafting, writing and debugging the program is a further independent measure of the performance of a system. Where it is possible to record this time for a given system and problem, ideally it should be quoted. For the same reason as in (2.2), separate times t_2 for a designer or one of his associates and t_3 for a mean nondesigner will be helpful.

(2.4) Even if (2.3) is too hard to organize, it may be easy to record an approximate measure of the preliminary effort in the form of the number of debugging runs required to achieve an error-free program which is ready for a final run to

generate the smallest possible demonstration value of t_0 or t_1. Again, separate values n_2 for a designer and n_3 for another programmer are desirable.

(2.5) At each stage of the computation for which an intermediate time is noted according to (2.1), the total number of words of storage occupied by the system, program and working space should also be noted.

(2.6) If the program for a given problem is not too long, a listing of the program can be of value. However, this is secondary to the question of how the program treats the unusual features (if any) of a problem. For example, in the present case, the compact notation of Eqs. (4) and (5) hides a small SC sub-problem. The Y functions are determined by differentiation, as Eq. (5) shows. Because of (5),

$$\epsilon_m' = \epsilon_{m+1} \tag{8}$$

and

$$(\epsilon_m{}^p)' = p\epsilon_m^{p-1}\epsilon_{m+1}\,, \qquad (\epsilon_m\epsilon_k)' = \epsilon_{m+1}\epsilon_k + \epsilon_m\epsilon_{k+1} \tag{9}$$

for any p, m and k. While (8) is a "derivative", it does not fit the usual definition well enough for the standard differentiation functions of most SC systems to be used. Thus one may be tempted to fall back on the systems' facilities for substitution. If this is done, however, the high-speed operation of the chain rule is lost, even though the repeated use of the chain rule of Eq. (9) is at the heart of the computation. We would extend the definition of "unusual features" to cover anything which is responsible for making t_0 significantly less than t_1

Our presentation of results is divided into the categories mentioned above. Computation in each case is based on the slow algorithm utilizing substitution of Eqs. (3) and (2) into Eq. (1), because this is the only algorithm common to the previous comments [2–4, 6–8] about performance. All information has been recorded on a CDC 6600 computer, unless otherwise stated. The presentation is preceded by remarks about each of the three systems or languages.

TRIGMAN [11] is specifically built to manipulate series such as the Y_{2n} functions. It represents a type of system which is specialized to the job in hand, in this case. Alternatively, it is an example of a system whose syntax resembles that of FORTRAN. (Indeed, to the casual user it "is" FORTRAN, with minor additions).

SYMBAL [12] is a general-purpose SC system, which the user must specialize to his needs by writing these into his own program. Alternatively, it is an exemplary ALGOL-like system.

Thirdly, we have taken the language LISP [13], which is not a "system" at all, but which acts as a base for several systems, including REDUCE [4] and SCRATCHPAD [6]. Thus we have to write everything, including the differentiation functions, from the ground up for this (and each) special case. We believe that LISP is so general for SC that it can always be used with success for problems which are intractable in the polynomial-handling systems.

(2.1) and (2.2): The timings for each of the three choices are shown in Table I. The values of t_0 refer to our own efforts, and t_1 to those of volunteers. We do not give a t_1 column for LISP, as (2.4) explains below. The relatively early version 1.1E of SYMBAL was used for t_0 and t_1. There is now a later version 2.D. Dr. M. Engeli has kindly run the problem in this version on the Control Data computer ($\approx$ CDC 6500) of the FIDES-Rechenzentrum in Zurich. The values t_{00} are the times for his computation divided by 2.2, a factor which describes well his experiences of the differences in SYMBAL between the Zurich and Texas computers.

TABLE I

Computing Times, in Seconds, for Each Stage of the Y Problem. The Values for Each Order are Cumulative Times from the Start of the Computation of the Y Functions, but Excluding Compilation, Which is Given First as a Separate Item.

STAGE	TRIGMAN		SYMBAL			LISP
	t_0	t_1	t_{00}	t_0	t_1	t_0
"Compile"	18.954	18.471	0.350	0.447	0.429	31.642
$n = 2$	0.278	0.259	0.091	0.069	0.099	0.106
$n = 3$	0.767	0.812	0.322	0.287	0.420	0.631
$n = 4$	1.463	1.944	0.945	0.888	1.183	1.522
$n = 5$	2.406	3.582	2.323	2.280	2.953	3.337
$n = 6$	3.828	5.603	5.929	6.040	7.930	6.814
$n = 7$	6.290	9.408	—	15.839	19.214	18.829
$n = 8$	10.666	16.032	—	44.571	57.681	86.190
$n = 9$	17.975	27.194	—	—	—	—
$n = 10$	30.991	45.897	—	—	—	—

Each computation except that reported by Dr. Engeli is cut off in Table I after the highest order reached without excessive garbage collections or automated arguments with our operating system. This number is in itself a measure of performance.

From Table I, it appears that "expert knowledge" increases the running speed of a SYMBAL program by 25 % and a TRIGMAN program by about 30 %. Even for unrelated problems, we observe that this percentage increases linearly with the amount of storage needed to hold a system. The estimate, which may be a useful rule of thumb, is supported by the idea that the number of idiosyncrasies which the designer can exploit best may be simply proportional to the length of his system. As a guide to the scale, the system TRIGMAN occupies about 14250 words of CDC 6600 storage.

(2.3) and (2.4): Unfortunately we had already finished the drafting and debugging of our first program before (2.3) occurred to us. As we were not able to unlearn what we had learned about the problem, we considered that any value of t_2 would have been too biased for comparison with volunteers' values of t_3. Therefore we have only recorded numbers for n_2 and n_3.

TABLE II

Numbers of Attempts Needed to Produce an Error-Free Final Program for the Y Problem in Each Case. The Number n_2 Refers to the Work of the Authors, and n_3 to the Work of Volunteers.

TRIGMAN		SYMBAL		LISP	
n_2	n_3	n_2	n_3	n_2	n_3
9	12	9	7	12	>20

Table II gives a reasonable impressionistic picture of the advantages of special knowledge about the three alternatives. The numbers for SYMBAL reflect an unhappy fixation of one of us in shortening the loop for quadrilinear sums. With Dr. Engeli's program also before us, we conclude that special knowledge of SYMBAL confers no advantage over the person who has recently learned SYMBAL in basic courses—a strong recommendation for the nonspecialist. The two numbers for LISP cannot be compared, because the sole volunteer gave up after 20 unsuccessful tries. Nevertheless, the finite value of n_2 shows that LISP is still a viable last resort if attempts to fit a problem into a higher-level SC system fail.

(2.5) In the absence of good diagnostics in the three cases, we report only total storage occupancy of system, program and working space. For $n = 8$, which is a representative example, the numbers of words required are: 29888 (TRIGMAN), 65536 (SYMBAL), 73728 (LISP). To improve the information available under this heading, we would like to have available for all systems the storage-measurement capabilities for CAMAL described by Fitch and Garnett [14].

(2.6) The "unusual feature" of the Y-problem is contained in Eqs. (8) and (9). In both TRIGMAN and SYMBAL, we have had to mix substitution with differentiation. Thus, to represent the effect of $\epsilon_4' = \epsilon_5$ on some Y, we write

E5*DERIV(Y,E4)

in TRIGMAN. Each specific ϵ-quantity must be written out separately. SYMBAL improves on this situation by allowing subscripted variables, e.g.

DY[M] := "FOR" J := 0:2*(M−1) "SUM" E(J+1) * "D" E(J) $ Y[M]

for a single derivative. In LISP, the lack of "system" superstructure allows us to make the optimum solution to the difficulty: we merely add one short line to our differentiation function to incorporate Eq. (8), e.g.

((EQ (CAR Y) (QUOTE E)) (CONS (CAR Y) (ADD1 (CDR Y)))),

and the chain-rule section of the function automatically deals with Eq. (9).

Our conclusions divide into two parts. For the designers of SC systems, the conclusions are expressed in the recommendations above. For intending users of SC systems, we have stated the three classes of tool presently available (special-purpose systems, general-purpose systems, "no system"—alternatively FORTRAN-like systems, ALGOL-like systems, SC languages), and have compared their performance in discussion and in the Tables. Although we have confined our attention to one computation here, our experience with other computations has been that they bring out exactly the same types of advantages and disadvantages among the three classes. Hence this note should also serve as a general guide to the alternatives open to casual users in their first encounters with SC.

Acknowledgments

We thank Dr. M. E. Engeli, Mr. A. Leonhardt, and volunteers from the courses C.S. 340 and C.S. 345 at the University of Texas for their help in contributing to the Tables. The work outlined here has been supported in part by the National Science Foundation under grant no. GJ-1069 for the development of specialized algebraic programs.

References

1. J. A. Campbell, *J. Comp. Phys.* **10** (1972), 308.
2. W. S. Brown and A. D. Hall, *SIGSAM Bull. ACM*, No. 24 (1972), 4.
3. S. R. Bourne, *SIGSAM Bull. ACM*, No. 24 (1972), 8.
4. A. C. Hearn, *SIGSAM Bull. ACM*, No. 24 (1972), 14.
5. D. Barton and A. C. Hearn, *SIGSAM Bull. ACM*, No. 24 (1972), 15.
6. R. D. Jenks, *SIGSAM Bull. ACM*, No. 24 (1972), 16.
7. Y. Sundblad, *SIGSAM Bull. ACM*, No. 24 (1972), 18.
8. E. Horowitz and M. Siegel, *SIGSAM Bull. ACM*, No. 24 (1972), 22.
9. Y. Sundblad, "One User's One-Algorithm Comparison of Six Algebraic Systems on the Y_{2n}-Problem," Royal Institute of Technology preprint, Stockholm (1973).
10. N. Fröman, *Ark. Fys.* **32** (1966), 541; N. Fröman and P. O. Fröman, *Nucl. Phys.* **A147** (1970), 606.
11. W. H. Jefferys, *Celest. Mech.* **2** (1970), 474; *Celest. Mech.* **6** (1972), 117.
12. M. E. Engeli, *Adv. Information Systems Sci.* **1** (1969), 117.

13. J. McCarthy *et al.*, "LISP 1.5 Programmer's Manual," MIT Press, Cambridge, Massachusetts, 1965.
14. J. P. Fitch and D. J. Garnett, Measurements on the Cambridge Algebra System, *in* "Proceedings of the Association for Computing Machinery International Computing Symposium, Venice," pp. 139–147, ACM Headquarters, 1972.

Received: November 7, 1973

J. A. Campbell

King's College Research Centre, King's College, Cambridge, England

William H. Jefferys

Dept. of Astronomy, University of Texas at Austin, Austin, Texas 78712