Lew Randall

CAN YOU COUNT ON YOUR CALCULATOR?

by

W. Kahan and B. N. Parlett

Memorandum No. UCB/ERL M77/21

6 April 1977

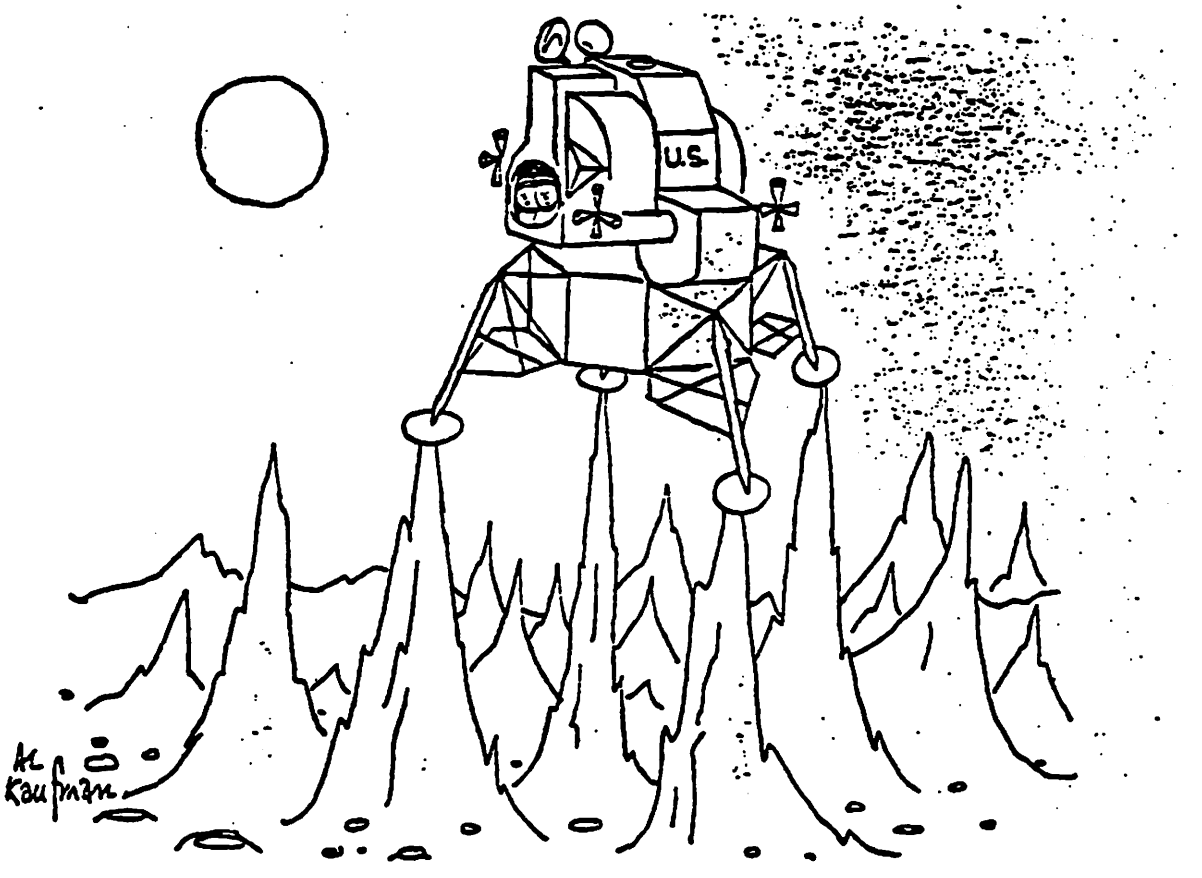
ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

CAN YOU COUNT ON YOUR CALCULATOR?

W. Kahan and B.N. Parlett

March 1977



Saturday Review

"You've got to hand it to those computers."

Both authors are professors of Mathematics and of Electrical Engineering and Computer Science in the University of California at Berkeley, California 94720. The first author is occasionally a consultant for Hewlett-Packard's Calculator Division.

CAN YOU COUNT ON YOUR CALCULATOR?*

W. Kahan and B.N. Parlett

Introduction

Do you care whether your hand held calculator occasionally, very occasionally, gives completely erroneous results for innocent looking problems? It can happen; and sometimes the false answers look quite plausible. The victim of such a malfunction, if he is aware of it, is apt to lose confidence in the results of subsequent computations. When will Misfortune strike again? A conscientious user may be driven to waste a lot of time checking answers which are, nearly always, quite correct.

Nevertheless, in certain cases incorrect answers seem practically unavoidable because calculators, and large computers too, work with numbers represented by a definite number of digits, frequently 10 or 12 decimals, and this constraint imposes intrinsic limitations on what can be achieved with reasonable expenditure of time. Consequently, as we take precautions against being misled by rare miscalculations, should we not marvel that they do not occur more often? Perhaps the risk of occasional bizarre results is unavoidable?

The problem is not so simple as may appear at first. We maintain that, despite its limitations a calculator need never deliver misleading answers. Calculators can be designed in such a way that if a user does encounter strange output he can be sure that it is not a consequence of anything capricious that his machine has done to him but must be attributed to his data, his problem, or his procedures. What is more such a calculator can be designed at a reasonable cost; but that story is for another day.

Our goal is first to alert the unscathed user to the fact that funny things can happen on even the best products currently available and, second

*Research supported by Office of Naval Research Contract N00014-76-C-0013.

to convey a particular point of view which is the prime tool needed to extirpate anomalies. Our method is to present three simple but perplexing examples from a whole file of mangled computations. The discussion of each example illustrates our approach.

At this point another difficulty arises: our remarks are critical. All too easily can we alienate those whom we are most eager to reach, owners and designers of the more powerful calculators. In fact we are reminded of the Viennese doctor Ignaz Semmelweiss and his Scottish follower Joseph Lister who, in the 1860's, suggested that surgeons should wash their hands before operating. Since cleanliness had not previously been acknowledged as a problem in medical practice, their suggestions were ill received. Medical men of those days, especially those philosophically inclined, proclaimed that the risk of death was intrinsic in surgical intervention and therefore unavoidable. How right they were. And, as they uttered this truth while avoiding the extra costs of sanitary precautions, they made the truth more true. Now washing hands and donning a clean coat and boiling medical instruments all do cost something, but surely we may say that the cost is negligible. The cost of cleaner arithmetic, cleaner than has been customary in big computers as well as little calculators, is negligible too. Unfortunately our analogy is imperfect because the benefits of sanitary precautions are now obvious in hospitals but still unobvious in calculators. Consequently our criticisms must be placed in the proper context. The advanced hand held calculators are triumphs of modern technology. The ones we mention below are among the best available today and are very good value for money. Next year's models could be much better.

Example 1. Non-Standard Deviation from the Strait and Narrow

Several calculators are pre-programmed to deliver the slope m and the intercept c of the straight line

$$y = mx + c$$

which best fits k data points $(x_1,y_1),(x_2,y_2),\ldots,(x_k,y_k)$ in the least-squares sense. Our example consists of three sample points drawn from the line

$$y = x - 666000 ;$$

the fact that the points happen to lie exactly on the straight line is not

what causes trouble. Many other examples could have been invoked to upset various calculators; our example's sole distinction is that it upsets so many simultaneously.

Data: k = 3

x	y
665999	-1
666000	0
666001	1

Expected Answer: m = 1, c = -666000

Answer Received: See Table 1.

Table 1

Calculator	Number of significant decimals carried by the calculator	Result instead of m = 1
Hewlett-Packard 22, 27, 91, ...	10	"Error"
Texas Instruments SR-51,51-II	12-13	Blinks 9's
Commodore SR 4190R, 5190R	12	"Error"
Texas Instruments Business Analyst	11	m = -0.02
Hewlett-Packard 65, 67, 97*	10	"Error" or Blinks
Texas Instruments SR-52,56*	12-13	Blinks 9's

*These are programmable calculators using the manufacturers' supplied software.

Discussion

> "Let him who has the mind for it calculate the number of the beast, for it is a man's number, and his number is six hundred sixty-six."
>
> Book of REVELATION ch. 13 v. 18.

Aside from this quotation we know no reason to fear the data points, especially when they are entered into calculators carrying at least twice as many digits as are specified in the data. Alas, the redundant 6's embarrass the textbook formula for the sample variance

$$\sigma_k^2 = \left(\sum_i^k x_j^2 - \left(\sum_i^k x_j\right)^2/k\right) / (k-1)$$

which is needed to calculate m. This formula is a natural one to use when sums like $\sum x_j$ and $\sum x_j^2$ are accumulated as the data is entered into the calculator. But in our example $\sum x_j^2 = 1{,}330{,}668{,}000{,}002$ must drop its last (13th) digit 2 if it is to fit into the 10-digit machines' registers, and consequently σ_3^2 must be calculated as 0 instead of the correct value 1. A 13-digit calculator appears to be needed to get the right answer for our example; but why do those 13-digit calculators fail? The TI SR 51 fails because it stores only 12 of the 13 digits it calculates for each entered datum. The TI SR 52 has no such excuse; it fails because its arithmetic occasionally mangles the thirteenth digit when operated as its manufacturer advises.

Our example usually elicits two contradictory reactions, sometimes both from the same person. The first reaction is to protest that unreasonable demands are being placed upon the accuracy of computation. After all, accuracy is only a means to another end, and no worthwhile end is served by this example except to demonstrate what happens when a calculator is applied to a problem outside its domain. We shall deal with this reaction later.

The second reaction is to point out tricks by which the correct answer could be coaxed from the calculator. For instance, all goes well if any of the redundant leading 6's are stripped off all the x-data and put back at the end; this amounts to a translation of the x-origin into the midst of the data before calculation and a translation back afterwards. But the trick is a mixed blessing. Although the trick can be proved mathematically to be reliable and has reappeared in the literature recently (Harms, 1976), it is not a trick we should expect calculator owners generally to know and trust since it is mentioned nowhere in texts nor is it touted as a "feature" of the calculators in their instruction manuals. Moreover, the trick entails the nuisance of extra thoughts, extra keystrokes, and extra opportunities to blunder. Did you remember to undo the translation at the end of the computation? What if the data (x_i,y_i) are generated by a program and cannot be scanned or predicted in advance? How does the trick handle the fit to a power curve $y = Cx^m$ when the x-data is closely clustered? All these questions can be answered; their answers miss the point.

The point of our example is that the foregoing formula fails on quite reasonable data. We would be free from troubles and tricks if the calculators had been microprogrammed to use a different and better algorithm like the one reproduced in Fig. 3. In other words, calculators which use the foregoing σ_k^2 formula are capable of solving reliably only those problems whose data lie in an artificially restricted domain.

Which brings us back to the first reaction: we should not use a function outside its domain. Where is the boundary of that domain? It is not discussed in texts nor in calculators' instruction manuals. It is too complicated. In Appendix 1 we describe that domain in order to persuade you that the information that has hitherto been denied you is information you would prefer not to have to know.

None of the troubles or tricks are necessary. As soon as statisticians and computer programmers realized that the troubles were caused by their algorithms and not by their data, they invented (and re-invented, from the 1930's onwards) better algorithms. One such algorithm is discussed in an elementary text by Forsythe et al. (1975), and we reproduce it in Fig. 3.

This alternative scheme costs only a few extra arithmetic operations per datum, but it delivers answers accurate enough for all statistical purposes no matter how many figures are specified in the data. If the mathematically correct answer is wanted (this will be more accurate than normally needed for statistical purposes) there are other schemes which cost only a few more storage registers and a few more arithmetic operations (clean, not dirty ones). The benefit that justifies these schemes' extra cost is that their user need not fear that his answer might be wrong because of roundoff. His answer may be wrong because of statistical or other methodological flaws, but not because of anything done to him by his calculator.

The mistake made by the designers of the calculators listed in Table 1 was their assumption that a standard formula found in many texts was the way to do the calculation. We know just how it feels to make that kind of mistake, so we won't laugh nor jeer. Besides, some of our colleagues have published "improvements" of that formula which have turned out to be worse. We do hope that our examples will stimulate designers to think about their calculators' other functions (logarithmic, trigonometric, ...) in a clearer light. Are bizarre results possible? If so, are they avoidable? If so, can they be avoided at a tolerable cost?

The foregoing example concerned the question of finding robust formulas. The next example concerns the care with which basic arithmetic operations and elementary functions are carried out.

Example 2. A Financial Miscalculation

Suppose n payments of $P apiece are deposited in a bank at the end of each of n periods during which interest accrues at the rate i% per period. Then these payments and their interest will accumulate to a "future value" $F which is given by

$$F = P(y^n-1)/(y-1) \quad \text{at} \quad y = 1+.01i \ .$$

Let us try the reasonable (though not nowadays realistic) values

$$n = 365 \text{ days}$$

$$P = 10{,}000.00$$

$$i = 3.6500364\% \text{ per annum compounded daily}$$

$$= \frac{3.6500364}{365}\%$$

Various results are given in Table 2. It is important to know that, according to their owners' manuals, "11 digits are carried for each result throughout all calculations" on the TI machine whereas the HP machine "always computes internally using 10 digits," although these calculators normally display 8 digits. Despite its extra digit of "accuracy" the TI machine has embezzled $27.93. This is not much in a sum of $3.7 million, but it is comparable with discrepancies through which gross computer-aided frauds have been uncovered. $27.93 is big enough to catch the bank examiner's eye and waste his time, big enough to undermine confidence that no bigger discrepancies lurk in those little calculators.

Although for larger interest rates the discrepancy would probably be smaller, for smaller rates it is often worse, and worse again on some earlier financial calculators; for instance, the obsolescent HP-80 would embezzle $37.78 here.

Table 2

Calculator (working accuracy)	Calculated Future Value $F
Texas Instruments Business Analyst (11 sig. dec.)	$3 717 213.88
Hewlett-Packard HP-22 or 27 (10 sig. dec.)	$3 717 241.83
$F correctly rounded to nearest cent	$3 717 241.81
Each interest payment rounded to nearest cent.	$3 717 241.85
Each interest payment chopped down to next cent.	$3 717 239.98

Discussion

Once again the point here is not to find a trick by which the correct answer could be teased out of a recalcitrant calculator. The point is that routine business calculations like these could have been completed satisfactorily by routine methods if the calculator's arithmetic had been implemented cleanly. Instead, avoidable arithmetic anomalies in the 11th figure have injected unnecessary noise into the 6th. Should the businessman trust all the figures displayed by his business calculator? If not all, then which? Must he take a college course to find out? The ghosts of Joseph Lister's medical colleagues might whisper (truly) that there are intrinsic and unobvious limits to what can be calculated with only 10 or 11 significant decimals, as if to suggest (falsely) that Example 2 lies beyond those limits. But today's calculator manufacturers will have none of that lest those whispers persuade businessmen to defer purchase until they can buy calculators carrying 16 or 20 or however many decimals are needed to get the right answers without reservations.

Will 16 or 20 digits be enough? We hope our example has engendered a certain uncertainty and wonderment. How did the calculator with the extra

digit of accuracy get the worse, the significantly worse, answer? It is not a fluke. The error in the TI calculator arises out of two sources that have been extirpated from recent HP calculators:

(i) Serious relative errors in logarithms of numbers very close to 1 undermine the calculations for large n of $y^n = \exp(n \ln(y))$ at $y = 1 + .01i$. For instance, this 11 digit calculator claims that

$$\ln(.99995) = -5.00016\ 00000 \times 10^{-5} \text{ instead of } -5.00012\ 50042 \times 10^{-5} .$$

An error smaller than this caused most of the embezzlement above.

(ii) Anomalies introduced into the tenth and eleventh figures during basic arithmetic operations (see three examples in Appendix 1) undermine the rest of the calculation.

Here we abandon Example 2 in favour of another which promotes the anomalies from the sixth significant decimal to the first. Because the next example concerns the same function as before, we hasten to assure the reader that we are not trying to get maximum mileage out of one troublesome function which we happen to have stumbled upon. This function was picked as the simplest way to show why neither intuition nor doctrine will protect us from the revenge that roundoff errors can take when they are dealt with in an arbitrary or slighting manner.

Example 3. A Good Formula Tainted by Dirty Arithmetic

The problem is to evaluate near $x = 0$ the function $f(x)$ which is given to us in composite form as $f(x) = g(y)$, $y = h(x)$ where

$$h(x) = (\tfrac{1}{3} - x^2)(3 + 3.45x^2)$$

is a tame 4th degree polynomial satisfying $h(x) \leq h(0)$, and where

$$g(y) = \begin{cases} (y^{127} - 1)/(y-1) & \text{for } y \neq 1, \\ 127 & \text{at } y = 1 \end{cases}$$

is another polynomial (of degree 126) despite appearances. Hence $f(x)$ is a polynomial (of degree 504) which behaves quite mildly for $-0.1 \leq x \leq 0.1$ as Fig. 1 shows.

Although the function $g(y)$ is perfectly smooth the formula given for $g(y)$ has a singularity at $y = 1$ which is apt to distract attention from the smooth behavior of $g(y)$ near $y = 1$. Therefore the function $h(x)$ has been devised solely to ensure that $y = h(x)$ will always be strictly less than 1 on every calculator. This is so because $\frac{1}{3}$ will be chopped or rounded down to 0.333...333. Consequently the formulas defining $f(x)$ never break down, although they may appear to flirt with disaster.

Table 3 shows values calculated for $f(0)$ by several calculators.

Discussion

Only the later HP calculators give the correct value 127; all others give very wrong answers but no warning. That behavior is typical of what happens with all tiny x. For no values x in $-1 \leq x \leq 1$ do the later HP calculators deliver fewer than 5 correct significant decimals; this cannot be said of the others.

Table 3

Calculator	Number of significant decimals carried by the calculator	Calculated f(0)
Hewlett-Packard HP 80	10	13
Hewlett-Packard HP 21,25,45,55,65	10	127
Hewlett-Packard HP 22,27,67,91,97	10	127
Texas Instruments Business Analyst	11	100
Commodore SR4190R, 5190R	12	12
Texas Instruments SR-50,50A, 51,51A	12-13	14
Texas Instruments SR-52,56,51-II	12-13	128
Monroe 326	13	12

Despite appearances, Example 3 was not designed as a commercial plug for HP calculators. The example was intended to discriminate between calculators that do and those that don't provide correctly rounded or chopped basic arithmetic and nearly correctly rounded logarithms and exponentials, and to discriminate independently of the number of significant figures carried.*

A natural reaction against Example 3 is to declare that it is worse than artificial, that it is pathological. Our financial example and the performance of the newer HP calculators should scotch that declaration. One reader suggested that the use of the formulas for f, g and h given above was asking for trouble, and that anyone misguided (or ignorant?) enough to use them deserved wrong answers. We will forego comment on the attitude revealed by such sentiments though we do regret that anyone should become so inured to dirty arithmetic as to accept it as natural law.

*We did not fully succeed. The earlier HP calculators listed in Table 3 do not round quite correctly, neither do they provide nearly correct logarithms and exponentials; but their errors appear to cancel fortuitously for our example. The Commodore SR4148R gets the correct answer too but via different and unlikely coincidences.

Another reaction is to devise alternative formulas for $g(y)$ which will survive dirty arithmetic. Here is one of its hidden costs, the needless exercize of ingenuity. Our example was not presented as a challenge to the reader to get the right answer somehow. Many an experienced programmer can circumvent the roundoff errors via different formulas for $g(y)$, but they all take much longer to evaluate than the natural formula and are more prone to key stroke errors. The simplest alternative is

$$g(y) = 1+y+y^2+\cdots+y^{125}+y^{126}$$

but it costs over 250 keystrokes. A less obvious formula is

$$g(y) = (1+y)(1+y^2)(1+y^4)(\cdots)(1+y^{32})(1+y^{64}) - y^{127} ;$$

but it is still much costlier than the natural formula

$$g(y) = (y^{127}-1)/(y-1)$$

using any of the following ways to evaluate y^{127}:

$$\begin{aligned} y^{127} &= y^{127} \text{ using the } y^x \text{ key} \\ &= \exp(127\ \ln(y)) \\ &= ((((((y^2)^2)^2)^2)^2)^2)^2/y \\ &= y(y(y(y(y(y(y^2))^2)^2)^2)^2)^2 . \end{aligned}$$

The surprising thing about this example, quite contrary to intuition, is that 10 clean digits do significantly better than 13 dirty ones, and you can prove it. With clean arithmetic and a reasonably reliable logarithm, you can <u>prove</u> that any of the four foregoing evaluations of y^{127} in the natural formula for $g(y)$ must yield values for $f(x) = g(h(x))$ correct to at least half the digits carried.* This is true both for tiny arguments

*The 12-figure TI SR-51-II performs this well only if y^{127} is calculated from one of the latter two evaluations without using y^x, exp or ln keys.

like $x \neq 10^{-8}$ and for not so tiny arguments like $x \neq 10^{-2}$, although different reasoning is needed in each case. For ways of attaining full accuracy in the evaluation of g see Appendix 2.

Let us glance briefly at what makes arithmetic dirty. The chief culprit in the production of many misleading answers, including those in Table 3, is lack of the internal guard digit needed to deliver correctly rounded results. Here is an example to show what happens without this guard; when $y = 0.99...99$ those calculators must calculate $1 - y$ as follows

$1 = 1.000...000$	...to the calculator's full precision
$-y = -0.999...999\not{9}$	...last 9 drops for lack of a guard digit
$1-y$ "=" $0.000...0010$	...is 10 times too big

To code around this kind of error the function $1-y$ may be replaced, for such calculators, by the expression $(0.5 - y) + 0.5$ or by $0.5 + (0.5 - y)$ except that only the latter expression works correctly on the old TI SR-52 (try them for $y = 3 \times \frac{1}{3}$). Different more complex tricks are needed to compensate for logarithms with low relative accuracy at arguments near 1. Multiplication and division without guard digits introduce phenomena (like $A \times B \neq B \times A$ or $\frac{9}{27} \neq \frac{1}{3}$) which matter only rarely, which is fortunate because the tricks required to compensate for those missing guard digits are prohibitively expensive.

A few disconcerting results soon teach the wily engineer to program defensively against the idiosyncracies of his calculator. He may engage in pyrotechnical programming or he may practice respectable mathematical analysis, but the energy he expends and the satisfaction he gains by his exploits represent a hidden cost to the company that employs him primarily to do something else.

The Moral of the Story

Our examples emphasize the contrast between calculators that do not and those that do try to perform their arithmetic tasks as cleanly as is mathematically possible despite that this cleanliness costs more to achieve. We have argued that clean arithmetic costs less to use, that it is more forgiving, that it may be used straightforwardly with better prospects of success, and that when things go wrong the reason will be more susceptible to analysis and repair, less an artifact of an inscrutable physical device. We claim that the benefits far outweigh their costs. We wish we could prove our claim like a mathematical theorem, but we cannot.

Alas, there does not exist any calculation that can be performed by clean arithmetic but not by dirty arithmetic; that is a theorem. The extra price that must be paid for always reliable computation with dirty arithmetic is the prior discovery by precise mathematical analysis of more than we wanted to know about our problem; this statement can be formulated as a theorem too. The cost of that analysis depends upon who does it, and how often, in ways that are best left unsaid. But this is how dirty arithmetic costs more to use than does clean.

Important to any fair discussion of complex devices like calculators is the distinction between (a) blunders in the implementation of a given performance specification, and (b) ill-conceived performance specifications. There is much to be said about both topics, but not here. A few slightly oracular comments on (a) are given in the next paragraph.

The thankless work of the Quality Assurance department will not likely improve in the near future as long as management sets so low a value on it as to begrudge the talent, the tools and the time for good work to be done. One obstacle to improvement is that quality assurance for calculators still

looks like a hardware job whereas in fact it involves difficult mathematical software issues, but in disguise. Blunders of type (a), unlike those of type (b), are sometimes acknowledged by the manufacturer when he recalls the calculator for remedial action. A recent instance is proclaimed in Table 4. As painful as such action may be for both customers and manufacturers, it reinforces rather than undermines confidence among present and future customers in the products' and producers' integrity, because acknowledged errors are the only errors from which we can learn to do better.

Table 4

x	$\sin^{-1}x$ radians correct to 10 sig. dec.	$\sin^{-1}x$ radians on HP 67 & 97
3×10^{-6}	3×10^{-6}	3.02×10^{-6}
4×10^{-6}	4×10^{-6}	4.10×10^{-6}
5×10^{-6}	5×10^{-6}	5.20×10^{-6}
6×10^{-6}	6×10^{-6}	6.40×10^{-6}
7×10^{-6}	7×10^{-6}	7.58×10^{-6}
8×10^{-6}	8×10^{-6}	8.92×10^{-6}

These six values x and their negatives, and the corresponding $\sin^{-1}$ and $\cos^{-1}$ values in radians, degrees and grads, are believed to be the only serious anomalies in early HP 67's and 97's. They were discovered by a customer, J.E. Dannenberg. The manufacturer is actively pursuing a way to resolve this problem.

This article has focussed on (b). It is a plea for what we call clean arithmetic; guard digits sufficient to yield correctly rounded algebraic operations, carefully evaluated elementary functions, and preservation of those mathematical relations whose violation would surprise a thoughtful customer (e.g. $a+b = b+a$). We do not seek an unattainble perfection;

the bibliography contains some items that show what can be done. And when we say that the added costs of clean arithmetic are negligible we refer to the performance penalty and to the added cost of production. The cost of designing cleaner arithmetic into calculators, especially the cost of prolonged delay during development, will not become negligible until the designers are familiar with the appropriate mathematical technology.

The necessary mathematical technology has been developed around some of the larger general purpose electronic computers. It is not so readily available as we would like because it is scattered among a few individuals and some relatively obscure publications; only some of the more sophisticated large computer installations benefit from this technology. Many a large computer suffers from arithmetic and elementary functions fully as dirty as the worst small calculator; cf. Kahan (1972). Moreover, most members of the mathematical community are ignorant of the technology needed by the calculator designer. Were he to seek advice indiscriminately he might well receive counsel of unattainable perfection. Worse, he could receive reaffirmation of those misconceptions which some of our colleagues taught him and still teach. An article in the Notices of the American Mathematics Society (December 1976) pp. 422-430, says

> It is inherent in any numerical computation on a machine retaining only a fixed number of digits, that subtraction of numbers agreeing in the first few digits is a very inaccurate operation,... . In numerical work, there is no real number zero.

These statements are misleading, the mischief coming from various meanings of the word "number". We do not wish to launch into mathematical niceties at this stage because all that needs to be said is that calculators work with digit strings of a prescribed length and these strings have no intrinsic meaning whatever. They can be interpreted in various ways and that is where the difficulties begin.

The assertions quoted above are natural consequences of interpreting each digit string in the machine not as a definite real number but rather as a sample from the interval of all those real numbers whose leading digits coincide with the given string. This interpretation is neither true nor false; it is simply a model, the so-called interval model. It leads its adherents to accept odd results ($ab \neq ba$) as inevitable; it is different enough from ordinary arithmetic that a business man will have to take time to absorb some of its implications if he is to accept a little embezzling as a predilection of certain calculators. The interval model does have its uses; the error lies in giving it the status of natural law.

An alternative interpretation, in widespread use among numerical analysts, regards each digit string in the machine as representing the unique rational number obtained by appending to the given string infinitely many 0's. The examples given above suggest that the choice of model is not simply a matter of taste; it provides the mental framework for formulating the crucial performance specifications which we have been discussing. We should choose the model which leads to the greater economy of thought.

Conclusion

"As you pass through life, brother,
Whate'er be your goal
Keep your eye on the donut
And not on the hole."
(Sign in a coffee shop)

We could fill these pages with a list of misconceptions and flaws revealed when trigonometric, hyperbolic, financial, statistical, navigational and numerous other functions are produced by various small electronic calculators. That would be a churlish way to acknowledge the calculator industry's magnificent accomplishment, especially when our intention is to help enhance it. We see the phenomenally widespread initial acceptance of these calculators as an unprecedented opportunity, and therefore obligation, to take some of the mystery and more of the tedium out of mathematics, and thus to improve one aspect of intellectual productivity. Most of that opportunity, and obligation, lies exclusively in the hands of a relatively small band of calculator designers. Any defects in their conception of finite precision arithmetic will be visited on all users for generations to come. Provided they are not rebuffed by ignorance and indifference in the marketplace, we trust they they will rise to the challenge once they know what it is.

Acknowledgment: We wish to thank our friends for lending us their calculators to be tortured and vilified.

Bibliography

(including some recommended reading)

Fike, C.T. (1968). Computer Evaluation of Mathematical Functions, Prentice-Hall, New Jersey.

Forsythe, Alexandra I., T.A. Keenan, E.I. Organick and W. Stenberg (1975). Computer Science--A First Course, 2nd ed., Wiley, New York. See esp. pp. 600, 675, 691.

Harms, D.W. (1976). "Improved Algorithms: Making $2^3 = 8$", in Session 32 'Advanced Pocket Calculators' of Electro 76, IEEE meeting in Boston, May 11-14, 1976. A Bowdlerized version appears on pp. 16-17 in the November 1976 Hewlett-Packard Journal, vol. 28, no. 3.

Kahan, W. (1972). "A Survey of Error-Analysis," Information Processing 71, Proceedings of 1971 IFIP Congress, Ljubljana; North-Holland Publishing Co., pp. 1214-1239.

Kahan, W. (1973). "Implementation of Algorithms," Parts I and II bound together and edited by David Hough, Technical Report 20, Computer Science Department, University of California, Berkeley. Now available from NTIS under DDC AD 769 124/9 GA.

Knuth, D.E. (1971). The Art of Computer Programming, vol. 2 (2nd printing), Ch. 4. Seminumerical Algorithms, Addison Wesley, Reading, Mass.

Osofsky, Barbara L. (1976). "Small Calculators for the Mathematician," Notices of the American Mathematics Society, vol. 23, no. 8 (December 1976), pp. 422-430.

Ris, F.N. (1976). "A Unified Decimal Floating-Point Architecture for the Support of High-Level Languages (extended abstract)," Report RC 6202 (#26651) (September 14, 1976), IBM Research Center, Yorktown Heights, NY. Also in ACM SIGARCH Newsletter, "Computer Architecture News," 1976.

Schmid, H. (1974). Decimal Computation, Wiley, New York.

Sterbenz, P.H. (1974). Floating-Point Computation, Prentice-Hall, New Jersey.

Walther, J.S. (1971). "A Unified Algorithm for Elementary Functions," Proceedings 1971 Spring Joint Computer Conference, pp. 379-385.

Appendix 1. The Domain of the Standard Formula for σ_k^2

For definiteness suppose $\sum_1^k x_j$ and $\sum_1^k x_j^2$ have been accumulated using 10 significant decimal arithmetic; for 12 significant decimal arithmetic replace 10^{-10} by 10^{-12} in what follows. Then roundoff could force the calculated value of σ_k^2 to suffer from a relative error as large as $5\times10^{-10}\sum_1^k x_j^2/\sigma_k^2$ roughly. In extreme cases, when that quotient exceeds 1 roughly, the calculated value of σ_k^2 might not be positive; otherwise the calculated values of slope m and intercept c can be no worse than if the x-data had been contaminated by not-quite-random noise whose standard deviation is roughly $5\times10^{-10}\sum_1^k x_j^2/\sigma_k^2$. Only when σ_k^2 is small compared with $\sum_1^k x_j^2$ can roundoff have a significant effect, but the appearance of squared terms here implies that the effect will be significant sooner than might intuitively have been expected.

Certain frequently encountered special cases should behave better than implied by the previous paragraph. When the data consists of at most, say, 100 pairs of at-most-4-decimal integers then the sums $\sum_1^k x_j$ and $\sum_1^k x_j^2$ are accumulated exactly; there is no roundoff. In these cases σ_k^2, m and c could be delivered correctly rounded provided they were calculated by aptly chosen algorithms. Unfortunately, there is evidence that some calculators' algorithms were not aptly chosen. For instance, try $x_j = 9966+j$, $y_j = j-1$ for $j = 1,2,\ldots,32,33$ $(k=33)$ on the Texas Instruments Business Analyst. Instead of $m = 1$, $c = -9967$ we get $m = .9989983305$ and $c = -9957.00033$. The discrepancies seem to be due to sloppy arithmetic; although the Owner's Manual (p. 3) says "All calculations are made to 11 digits...", the calculator says that $10 \div 3 = 3.3333333330$ and $(.99999999999)^2 = .99999999901$ and $1 - .99999999999 = .00000000010$ so it really carries only 9 or 10 digits correctly. Another example, this time for the Hewlett-Packard HP 91, is

$x_1 = 1971 \qquad x_2 = 1972 \qquad x_3 = 1973$

$y_1 = 300 \qquad y_2 = 325 \qquad y_3 = 350 \qquad (k = 3)$

which calculates for $\hat{y}(x) = mx + c$ the value $\hat{y}(1974) = 375.75$ instead of $\hat{y}(1974) = 375$ as correctly calculated by the HP 27 and others. See Fig. 2. The discrepancy here arises when an expression $a \cdot b + c \cdot d$ is evaluated as $(a \cdot b/c + d) \cdot c$. Neither of these algebraically equivalent expressions is appreciably more vulnerable to roundoff than the other except when all values happen to be integers and only the first is evaluated exactly. The calculator's designer chose the second expression because he had only two internal registers to work with and the first expression needs three; his manager had, in effect, asked him to fit a size 13 foot into a size 10 shoe.

Appendix 2. Attaining Full Accuracy for g

We cannot rejoice at the salvation of half the digits carried without mourning the other half's loss, even though so severe a loss almost never happens. For almost all the values of n and y that matter in financial calculations, the expression

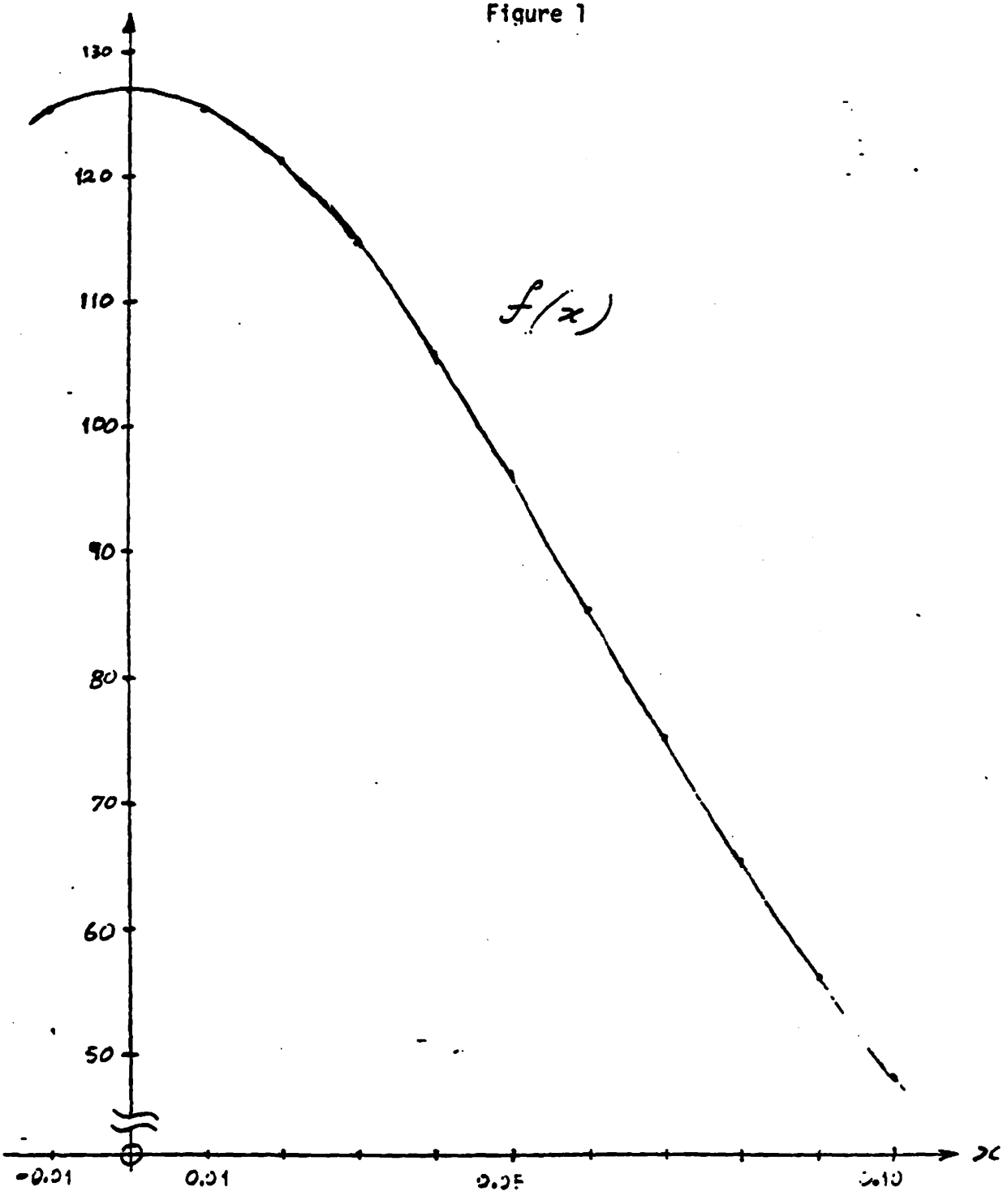
$$g_n(y) = (y^n-1)/(y-1)$$

is capable of producing correctly the 7 or 8 significant decimals displayed by the TI Business Analyst provided it be calculated with cleanly implemented 11 significant decimal arithmetic. (Perhaps the TI engineer who chose that expression did so under the illusion that the arithmetic would be cleanly implemented.) Better accuracy (e.g. the error confined to the last digit) requires either higher precision or a different formula. One different formula is displayed in Fig. 4; it calculates $g_n(y)$ correct in all but the last digit carried for all n and $y > 0$ provided reasonably clean arithmetic, logarithm and exponential functions are used. Whether Fig. 4 is a trick or a treat we leave to the reader's judgment after he has sought some better procedure. Neither Taylor series nor the expansion $g_n(y) = 1+y+y^2+\cdots+y^{n-2}+y^{n-1}$ can easily be exploited to match Fig. 4's performance, especially when n is huge; and if dirty arithmetic must be used the obstacles grow formidable. Besides, whether Fig. 4 is trickier than Taylor series depends on what you are used to.

Of all the calculators mentioned in our article, only the HP 22, 27, 67, 91 and 97 enjoy arithmetic clean enough to guarantee correct results from Fig. 4's procedure. Besides being far simpler to use than any other we know, it can be proved correct in less time than is needed merely to figure out, via Taylor series and other methods, the diverse procedures

that attain comparable performance on calculators with dirty arithmetic. (Watch out when $|n(y-1)| \gg 1 \gg |y-1|$!) This example is just one of many where clean arithmetic costs significantly less than dirty to use.

Figure 1



$$f(x) = g(h(x)) = f(-x)$$

$$h(x) = (\tfrac{1}{3} - x^2)(3 + 3.45x^2)$$

$$g(y) = (y^{127} - 1)/(y - 1)$$

Figure 2

Linear Prediction with HP 91

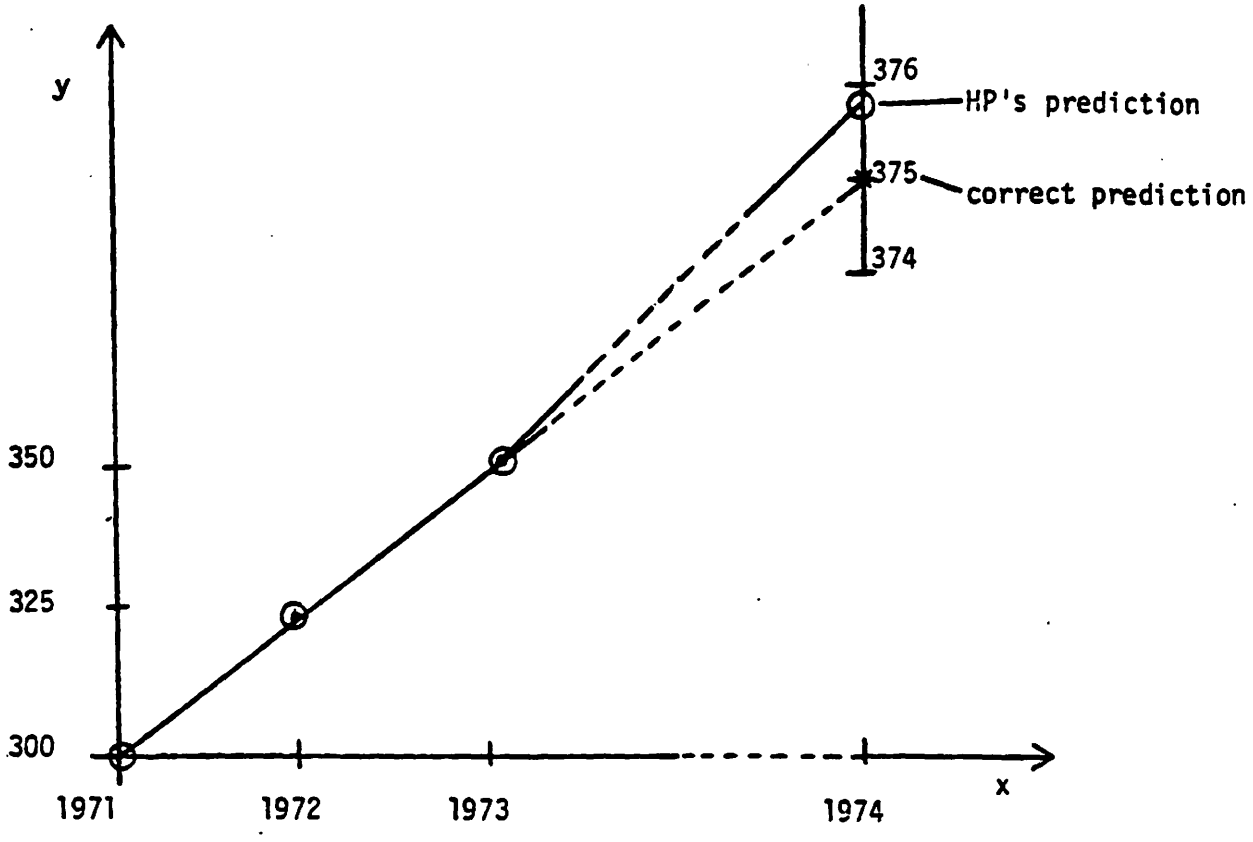


Figure 3

More Accurate Mean and Standard Deviation

Instead of accumulating $\sum x_j$ and $\sum x_j^2$, accumulate

$$M_k = \sum_1^k x_j/k \quad \text{(mean value), and}$$

$$Q_k = \sum_1^k (x_j - M_k)^2 = \sum_1^k x_j^2 - (\sum_1^k x_j)^2/k \ ,$$

via the recurrences

$$M_1 = x_1 \ , \quad M_k = M_{k-1} + (x_k - M_{k-1})/k \ , \quad k > 1 \ ,$$
$$Q_1 = 0 \ , \quad Q_k = Q_{k-1} + (x_k - M_{k-1})^2 (k-1)/k \ , \quad k > 1 \ ,$$

until all data has been entered, and then get

$$\sigma_k^2 = Q_k/(k-1) \ .$$

Figure 4

A procedure to calculate

$$g_n(y) = (y^n-1)/(y-1)$$

for any n and $y > 0$, with reasonably clean arithmetic.

1. Set $v = y^n$. If y^n underflows reset $v = 0$; if y^n overflows skip to step 6.
2. If $v = 1$ then $g_n(y) = n$ and all subsequent steps must be skipped. Otherwise (if $v \neq 1$) ...
3. Set $G = (v-1)/(y-1)$. If G overflows so must $g_n(y)$.
4. If $|v-1| \geq 0.1$ then $g_n(y) = G$ except for its last digit, and time may be saved by skipping all subsequent steps. However, the next step does no harm.
5. $g_n(y) = G \cdot (n \log(y)/\log(v))$ except for its last digit; if overflow occurs here (or in step 3) it is practically unavoidable.
6. This step is needed only to cope with the unlikely possibility that y^n may overflow although $g_n(y)$ does not. This can happen in only one way; if $n > 0$ then set $g_n(y) = y^{n-1}/(1 - 1/y)$. Otherwise overflow is practically unavoidable.