THE TABLE-MAKER'S DILEMMA

AND OTHER QUANDARIES

W. Kahan Univ. of Calif. at Berkeley

" *A little learning is a dang'rous thing ;*
Drink deep, or taste not the Pierian spring :
There shallow draughts intoxicate the brain ,
And drinking **largely** *sobers us again .* "

from "An essay on criticism"
Alexander Pope (1688-1744)

Pieria : a district in N. Thessaly, around Mt. Olympus, reputed home of the Muses, with springs whose waters were said to nourish the arts , etc.

The attached is a verbatim transcript of a speech delivered by Velvel Kahan at the Mathematical Software II Conference held at Purdue University on 29-31 May 1974. I have transcribed it from a tape made by Dick Jenks, and Prof. Kahan has graciously supplied me with the original foils used in his talk. Ideally, his colorful drawings should not be reproduced in black-and-white, but as they contributed so much to the spirit of the talk, I have felt it better to try to reproduce them (at somewhat of a reduction) rather than attempting to redraw them. The only editorial function I have performed is to indicate a certain amount of context in square brackets. This version should be regarded as draft only in which Prof. Kahan retains all rights.

The written contribution appears in the preliminary proceedings and, briefly summarized, explores the following two problems:

- The Table-Maker's Dilemma: Suppose I wish to compute the value of some function correctly rounded to the nearest integer. As I compute to 3 decimal places I obtain N + .499; to 6 places I obtain N + .500000; to 10 places, N + .4999999999; each correct to a unit or two in the last place computed. My dilemma is deciding when to stop computing to higher and higher precision and start trying to prove a theorem that the result is exactly N + 1/2. The counterpart for the person writing a similar function in software on a fixed-precision computer is to know how much intermediate precision he must carry in order to guarantee a correctly-rounded result at reasonable cost. "It is important to realize that Accuracy, unlike Virtue, is not its own reward, but a means to another end. That end is achieved, in numerical computation, through the conservation of mathematical relationships that vary widely from one application to another. When we do not know which mathematical relationships must be preserved and which can be abandoned in any particular application, we try to preserve as many relationships as possible; this is what accuracy is good for."

- The Error Analyst's Quandary: When should the error in a computational scheme be summarized in a simple way? Do so too soon, and the result may be too weak to be useful. Do so too late, and the result may be too complicated to be comprehended. And there is no guarantee that a gap exists between 'too soon' and 'too late'."

F. N. Ris
17 July 1974

First, I would ask you to please do unto me as I would do unto you. Interrupt me at the slightest provocation. Secondly, I would like to say that I believe what is to be found written in the proceedings represents accurately enough the way I feel about it, despite the fact that it was written one and two years ago.

Because it does not seem possible in under half an hour to explain what you ought to do instead of what you are now doing, I have chosen to give instead a brief sermon on the text which you see there by Alexander Pope. Before you get too insulted, I want you to understand that I do not believe that you are any more ignorant than I; but rather that collectively, by shooting from the hip or giving advice off the top of our heads, we have so confused the people who make hardware and software, that we have gotten computer systems just what we deserve. They are capricious, they involve enormous, staggering costs in order to produce what is called "portable" software, they are riven with misconceptions of every conceivable kind, and I suppose the best I could do now would be to try to address what I consider to be the worst of the misconceptions, and then perhaps make a few comments about the way I fear things may drift in the future.

The principal misconception, I think, concerns the concept of value in a numerical calculation; that is, calculations tend to be valued wrongly for their accuracy, when in fact what we ought to do is value them for how big we know the error isn't.

To illustrate this point, look here at two evaluations of pi. [Fig. 1] Now which of those do you prefer? I imagine that most of you would prefer the second, unless you happen to remember that the first is correct to all of the sixteen figures cited. And now when I tell you the first is correct to all the fifteen figures cited--thereby, of course, overwriting this piece [First error bound] of information—now which do you prefer? Of course. And that is really the problem; that you only prefer the better value when you know that it's better.

Now, at this point, I would have liked to digress to discuss portable software. I think it was the Indiana state legislature; I will save that for a different sermon on another day.

Now most people think about uncertainty in probabalistic terms, and although I find that style of thought uncomfortable, I will try to address the issue here with an example that makes the same kind of point as the previous one did.

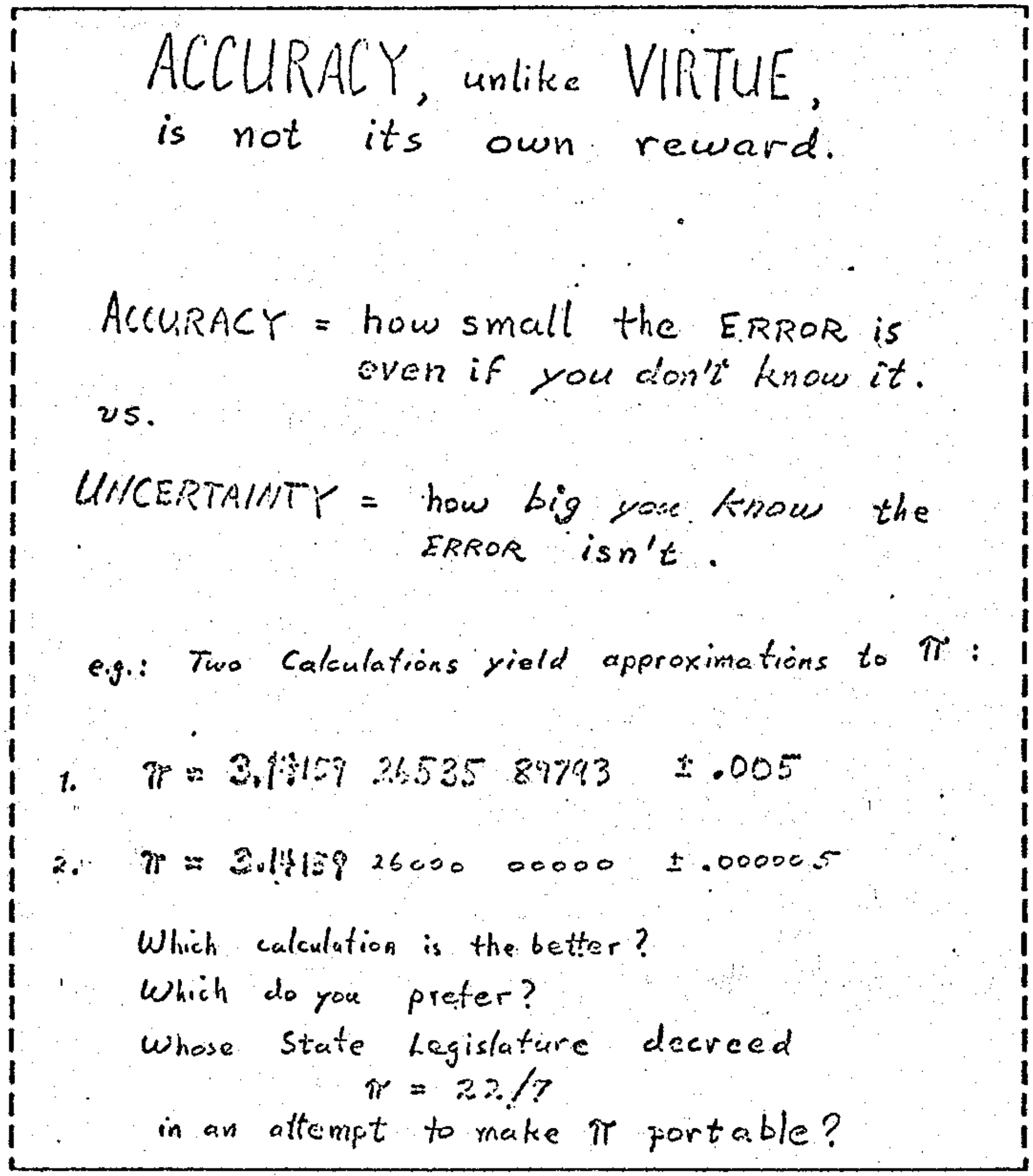

Figure 1

[Fig. 2] If you think of the error that comes from a calculation as having a probability distribution with the density shown here, and compare it with the competing calculation whose probability density is there, I imagine that most people would prefer the former, because the error appears to be smaller on the average. In fact, I should have perferred to have exaggerated this so that the spike in here would be very, very narrow--like a nail--compared with this lump. However, I don't think that one can apply a set of criteria in quite this way and get a satisfactory answer.

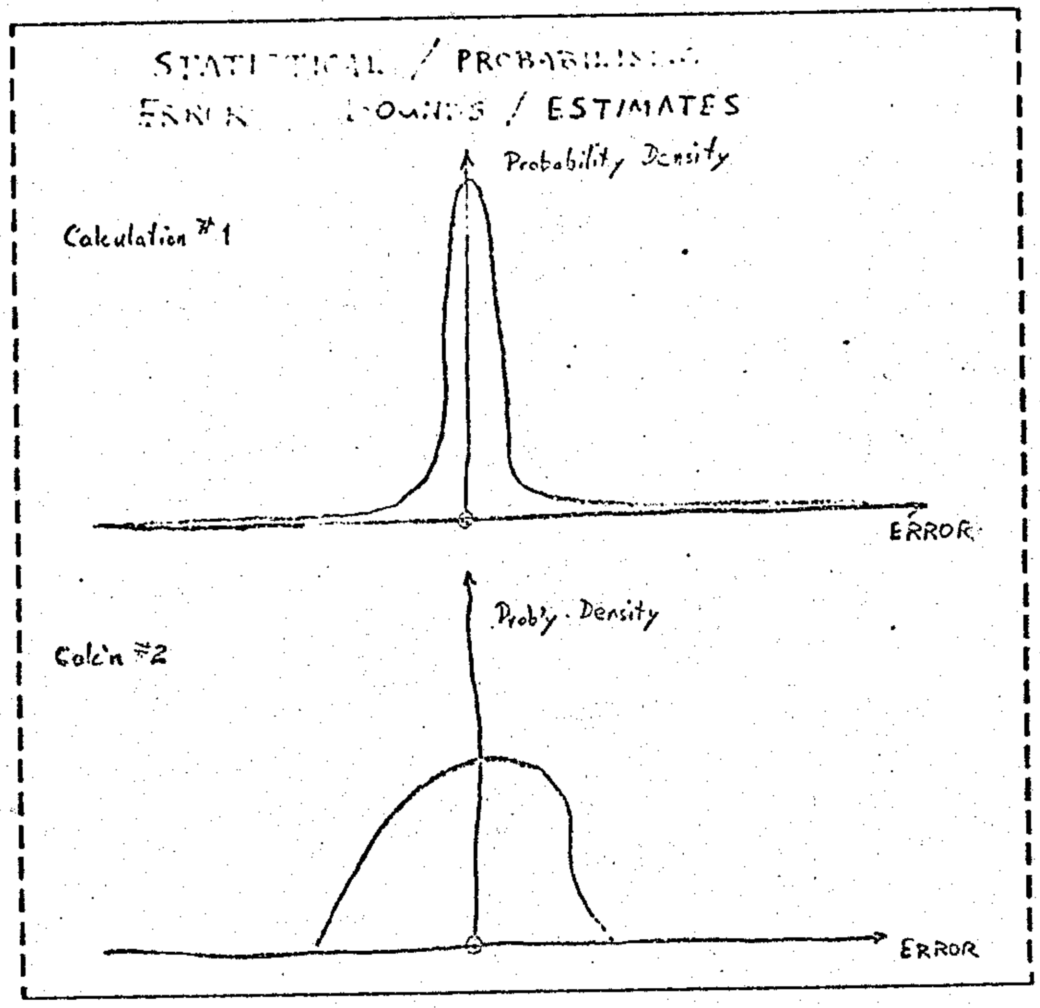

Figure 2

We have to apply criteria having to do with cost. It's not merely how likely is an error, but how much would it cost? Vyssotsky quoted a price of a hundred million; alas, I am not in a position to cite chapter and verse where any such error did cost a hundred million, because most people who suffer from numerical errors try to keep that a secret known only to their wastebasket. [Fig. 3] However we observe that in those cases where error is costly, the cost rises extremely rapidly once you get past a certain point.

Now I do not know where that point is, in general. It is my belief that at least one tenth, and probably one third of the results that are normally accepted by ordinary people as correct when they see them are much more wrong than they think, and they would probably come to different conclusions at least part of the time if they knew how wrong these answers were.

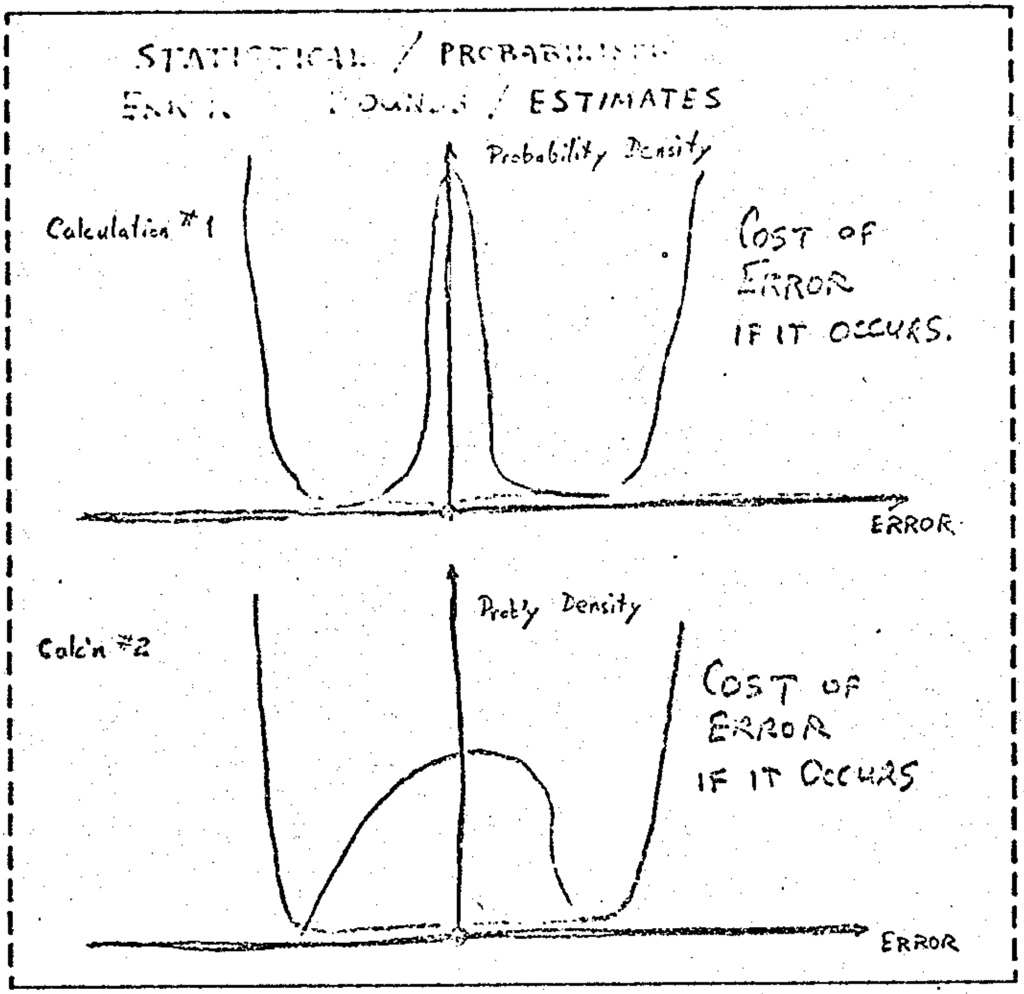

Figure 3

The reason I believe this is because I am a busybody, and I look over the shoulders of various people, and when I see the numbers I say to them, "How do you know that that's correct?" And then they give me an argument, and if the argument sounds like something that came off the top of their head, then I ask myself, "I wonder if I can teach them something, because that is my job." And so I'll run through the calculation, and I usually get a rather different answer; and this is true whether the person over whose shoulder I've looked is one of my students (who actually is responsible for teaching me things), or one of my collagues (whom I can't tell anything anyway).

The point is, that although I don't know what the cost is, I am convinced that it rises very steeply--past a certain point--in all those calculations in which cost matters. Maybe there aren't very many calculations where cost does matter, since nobody else seems to observe that their answers are wrong; so I guess maybe they needn't have

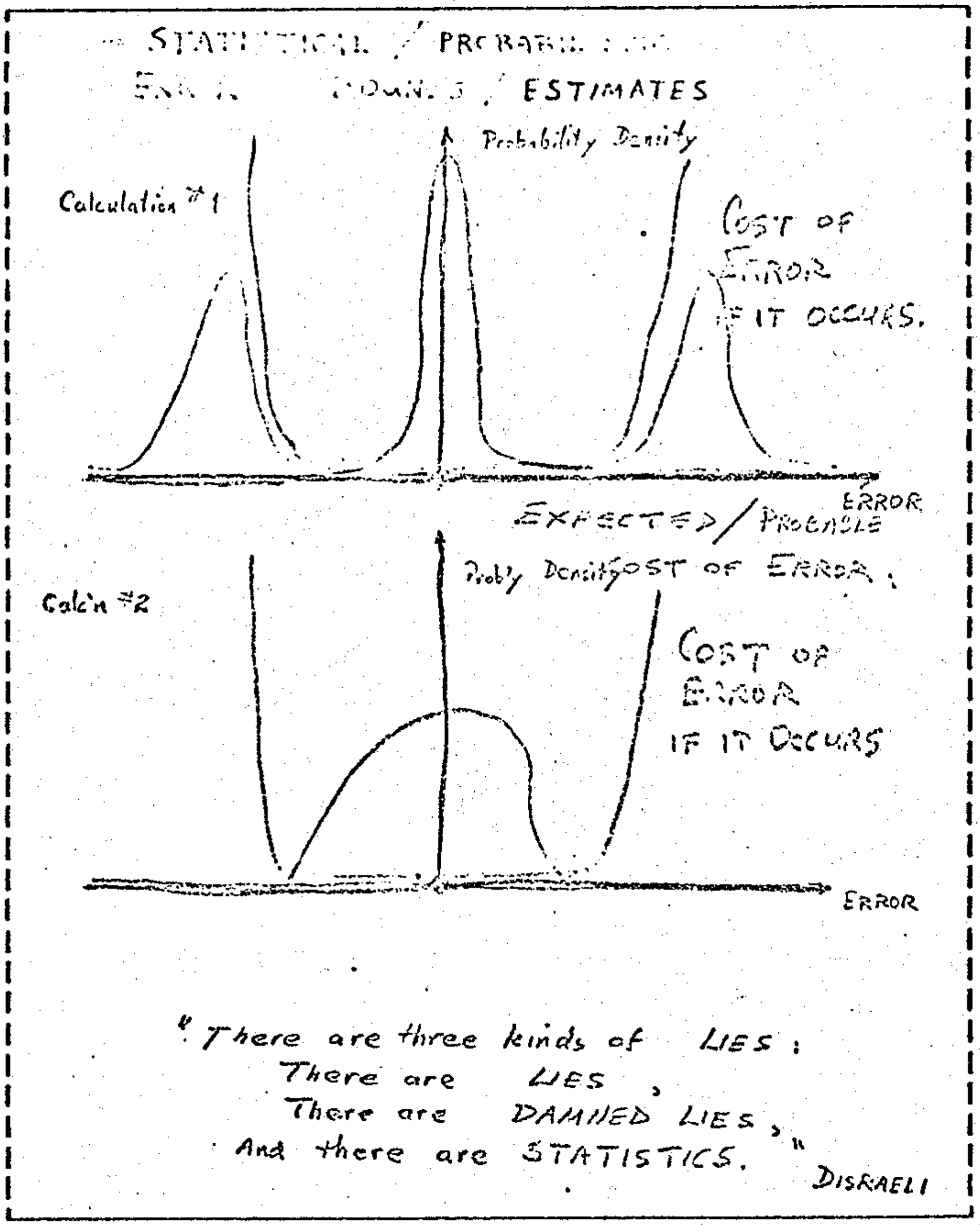

Figure 4

bothered to do them in the first place. Well, when we multiply cost times probability density, [Fig. 4] then you see the first calculation could be rather more costly a kind of calculation than the second, because the average cost of error would be rather greater. Well, as I said, I feel somewhat uncomfortable with statistical arguments, and for reasons that go back quite a ways [revealing Disrali quote]. However, I do hope that you recognize one thing, and that is that rather than talk merely about error, I am concerned about costs. I do not believe that aesthetic considerations

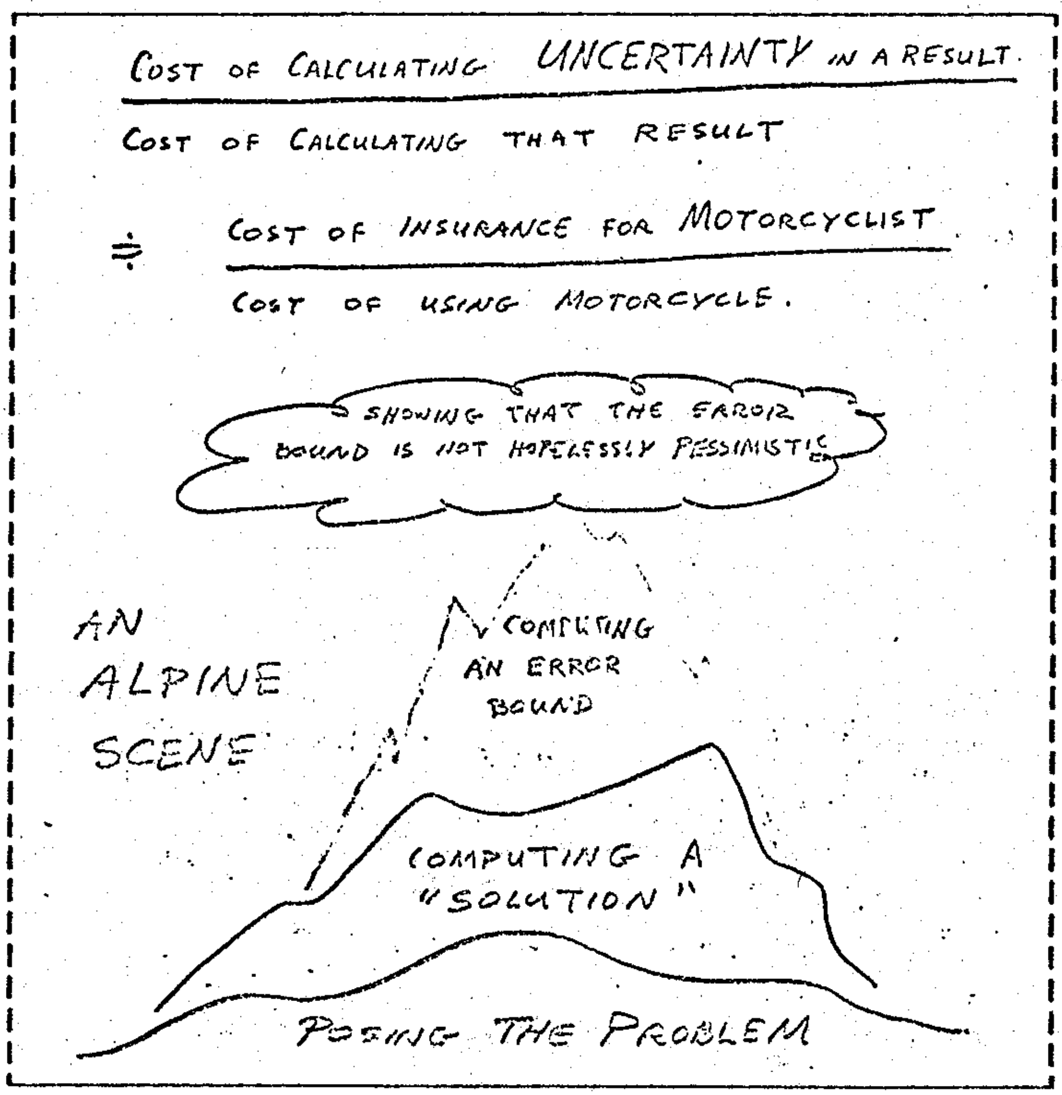

Figure 5

suffice to discuss these issues.

[Fig. 5] Now, if the issue, then, is not so much a matter of calculating an accurate answer as of calculating an answer whose uncertainty is known to be small, then we are entitled to ask: "How much more does the uncertainty estimate cost than the estimate whose uncertainty worries us?" And people who drive motorcycles know what I mean. It is not uncommon to find insurance premiums approaching a thousand dollars for a man whose motorcycle runs on perhaps a half a cent a mile for gasoline.

And indeed not only are the costs higher, so indeed is the intellectual effort involved in trying to establish error estimates. It's not too hard to pose a problem in the numerical domain. It's a little bit harder to compute a solution; hard enough that we feel proud of ourselves if we succeed. It tends to be extremely difficult, rocky, and unrewarding to give error bounds; and then to show that the error bound is not outrageously pessimistic is frequently beyond the scope of even the best mathematician. If somebody asks me for a cheap error bound, I merely say, "Somewhere between minus infinity and plus infinity."

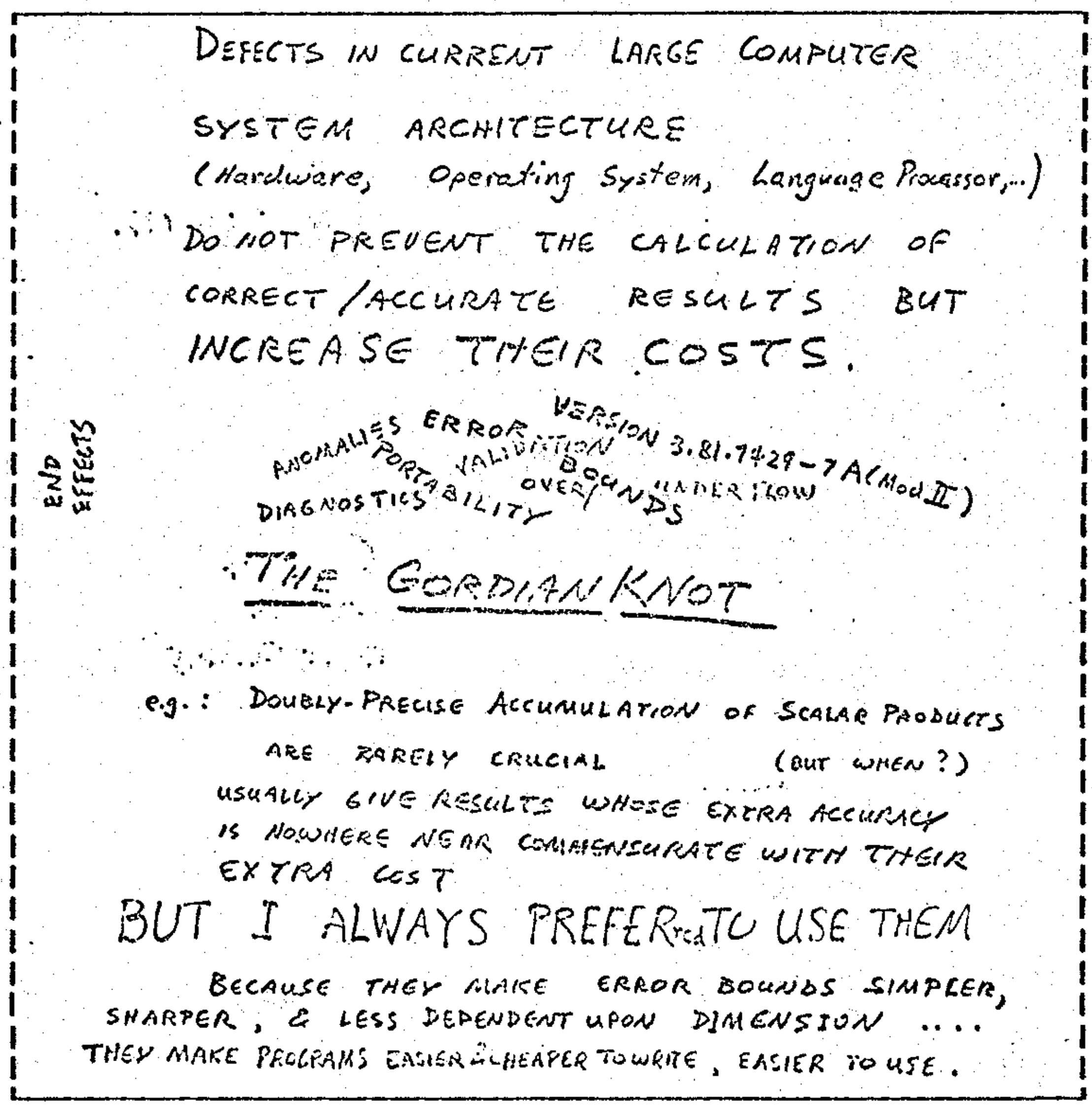

Figure 6

[Fig. 6] Now the trouble with the current computer system architectures is not so much that they prevent us from getting results. Anybody who walks out of this computing center with an armload of paper knows that there is no problem in getting results. The trouble is that finding out how good the results are can be terribly costly.

Let me give you one example to give you an indication of the way in which various factors interlock and contest for our attention. We know about doubly precise accumulation of scalar products. We know that they are alleged to be beneficial. The Europeans and Englishmen use them all the time because, as it happens, their calculators in many cases still reflect the state of mind appropriate to the man who uses a desk calculator. And I envy them. But we don't have that advantage. How much do we lose? Well, it is very rare that the loss is crucial. Unfortunately, I cannot easily tell you how to know.

It is very rare that you lose very much by failing to accumulate inner products to double precision. In fact, if you are doing Gaussian elimination, for example, or any other linear equation solver based on Gaussian elimination, or triangular factorization, or Crout or the Doolittle scheme or their descendants in numerical software, you are unlikely to benefit from doubly precise accumulation unless the following conditions are satisfied: 1) Your machine has a rather short word length. That means IBM 360 and 370 short arithmetic. 2) Your matrix is of <u>very</u> large dimension. Like 100. And 3) The elements in your matrix are uncertain by only a unit or two in the last place cited. If you do not satisfy those three criteria, then there is very little to be gained by using doubly precise arithmetic. Very little accuracy to be gained.

However, the point I'm trying to make is that the issue is not how <u>accurate</u> your results are, but how inaccurate you know they <u>aren't</u>. And that's where double precision pays off enormously. Because, by accumulating products to double precision, one can make the error bounds easier to find, they are sharper (that is, they are more nearly reflective of the true state of affairs), they depend less upon the dimensionality, the program is in consequence easier to write, because there are less fudgy, kludgy little things you have to stick in to prevent bad things from happening. Consequently, the program is also easier to use, because the relationship between intention and accomplishment is generally closer.

That's why I always liked to use doubly precise accumulation on the 7094 where the cost was negligible. However, machine design has not altogether advanced since then, and now the cost of implementing doubly precise accumulation in those cases where it would be useful is extremely great. We've heard about procedures that have been proposed by Lawson, Hanson, and others that might diminish those costs, though with side-effects concerning which I've already had arguments with them, and I don't think now is the time to repeat the arguments. But these issues are important, and their resolution is not necessarily going to be straightforward. Unfortunately, in this subject, every decision that you make seems to have ramifications, but perhaps that's why it's mathematics, where we know that small changes in axioms can lead to very large consequences.

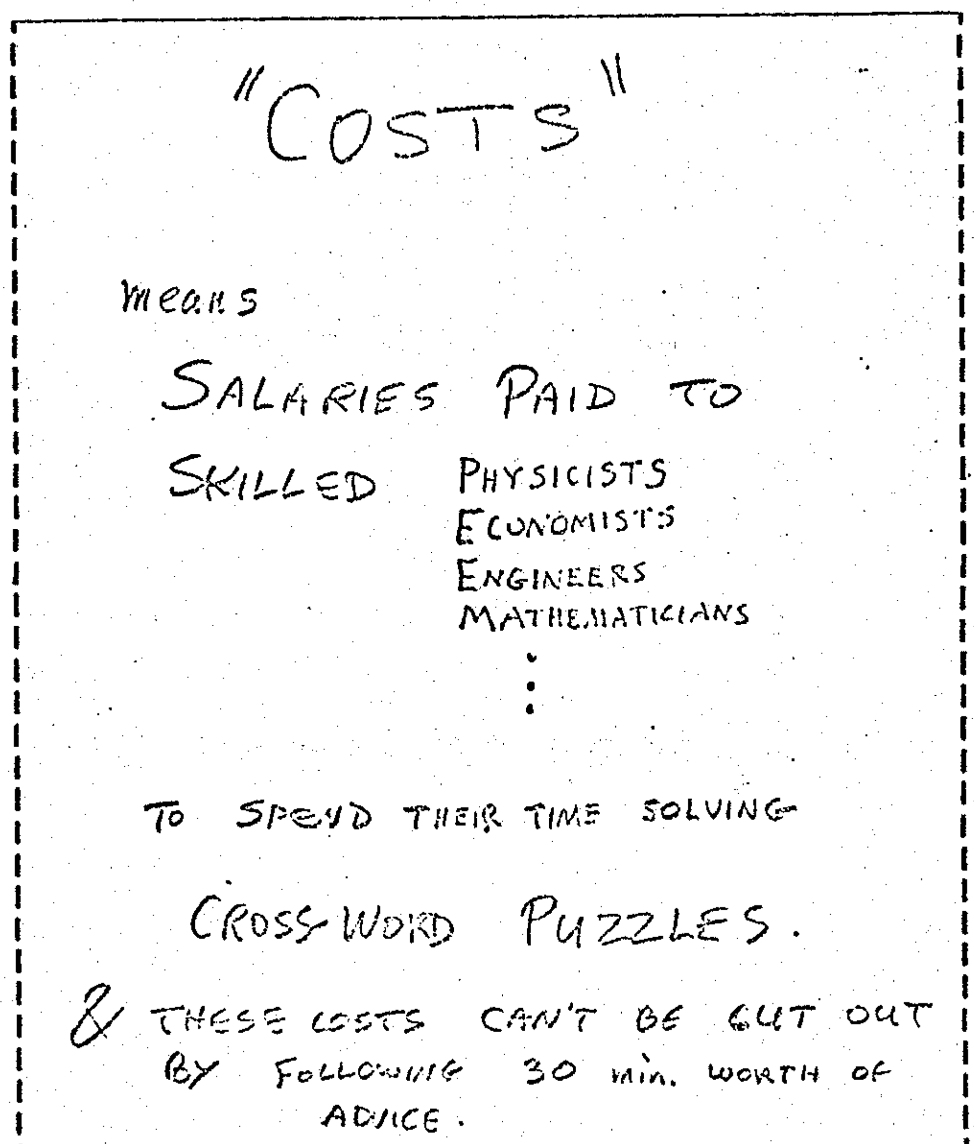

Figure 7

[Fig. 7] Now I've been talking about costs, and I should make clear that what I mean by "costs" are not things that will appear as line items in the budgets of most people who pay for computers, except as salaries where these costs get hidden. You are obliged to pay clever people who have been educated at great expense to play with the crossword

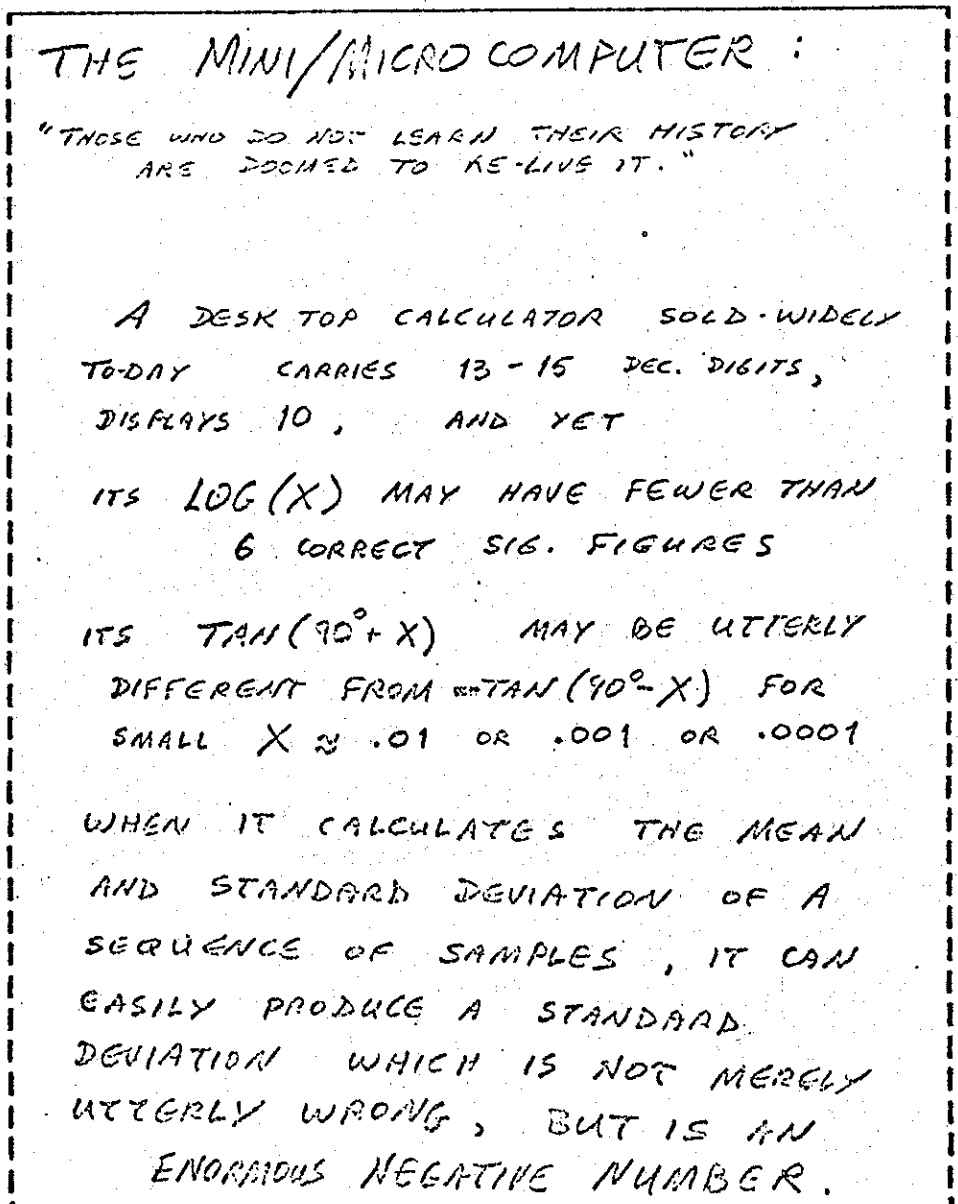

Figure 8

puzzles that are sometimes described as "outwitting the compiler", or "ensuring portability of code." Now I regret that, standing here, I cannot tell you in a short and snappy way how to get rid of those things, but they are things that difficulties have to be gotten rid of if we are to bring the cost of computation down to the point where we can afford to do it, never mind the vast multitude, who, we are told, are going to constitute the new market for the new range of computers. Perhaps we should look at that and speculate for a while.

DESPITE INNUMERABLE ARTICLES IN JOURNALS AND EXAMPLES IN TEXTS,

<u>ALL</u> HAND-HELD PROGRAMMABLE CALCULATORS NOW ON SALE WITH STANDARD "SOFTWARE" TO SOLVE THE QUADRATIC EQUATION

$$Ax^2 + Bx + C = 0$$

GIVE UTTERLY WRONG ANSWERS FOR ONE OF THE ROOTS OF

$$x^2 - 3\times 10^n x - 1 = 0$$

FOR n = 4 OR 5 OR 6.

The correct answers are innocuous & not uncommon:

n=4	30 000.0	.0000 333 333
5	300 000.0	.00000 333 333
6	3 000 000.0	.000000 333 333

Figure 9

[Fig. 10] Well, I'd like to come back to this one. You may feel it's a little bit outrageous, so let's consider an example.

[Fig. 8] Now, one day a salesman came in and asked whether I'd like to see a calculator which had been selling very widely on campus and of course I said yes. And as I played with it funny things happened and he looked over my shoulder and said, "Would you do that again?" It was clear that he was getting less and less happy, and I was getting perplexed myself, because I knew it was probably going to be bad, but I didn't see how it could be <u>that</u> bad. Yes, we ran in a sequence of numbers and got this incredible business where the standard devitation was a enormous negative number, and I can speculate on why that happens.

I must tell you, however, this computer is one of the most widely sold in the country. It is a desk calculator which you plug in, and it has a nice, bright display and very convenient keyboard and lots of buttons for statisticians or engineers to push. And, because it carries a very large number of digits, almost any test you can think of will give you accurate results. However, I must warn you that they are somewhat uncertain.

Well, what about the programmable calculators? They're interesting, because in them things can go on which, once again, you will not be able to see. In the good old days of the desk calculators, at least you could see your blunder as soon as you made it. But now, here's an interesting situation. [Fig. 9] I bring this up principally because there are people who keep on chastising me for not writing down what I know and publishing it, and they are right to chastise me, but <u>in defense</u> I might point out that there are at least three times I have written about quadratic equation solvers, at least two of them have been published, there are at least five other people who have published articles about quadratic equation solvers and we all agree pretty well on what you ought to do, but somebody hasn't done it. I suspect that what happened was that in the course of producing the software, management discovered that one of their engineers or somebody like that had not dealt too successfully with his assignments so far and they thought, "Well, what the hell, let's give him this job."

[Fig. 10] It seems to be crucial for the economic well-being of a large number of computer manufacturers that the computer market grow extremely rapidly, by factors ranging between ten and a hundred. The only way in which this market can grow is by bringing the power of computing, for good or for ill, to a very large number of people who, however well they have been trained in their chosen discipline, have not perhaps been trained as well in numerical analysis as some of us. They may become somewhat disgruntled. I can't predict that consumerism will spread from the ranks of breakfast foods and the automobile industry into the computing industry; I just don't know.

I do know, however, that the nature of the concept of warranty changes with time. It has changed in the law discernably in this country. Whereas at one time many warranties could be superseded by disclaimers, there are amply many areas where the disclaimers are ignored by the courts. They tend to have the following characteristic when they are ignored. Namely, if it is believed by the courts that a reasonable man who bought a given article and who expected that article to behave very much as other like

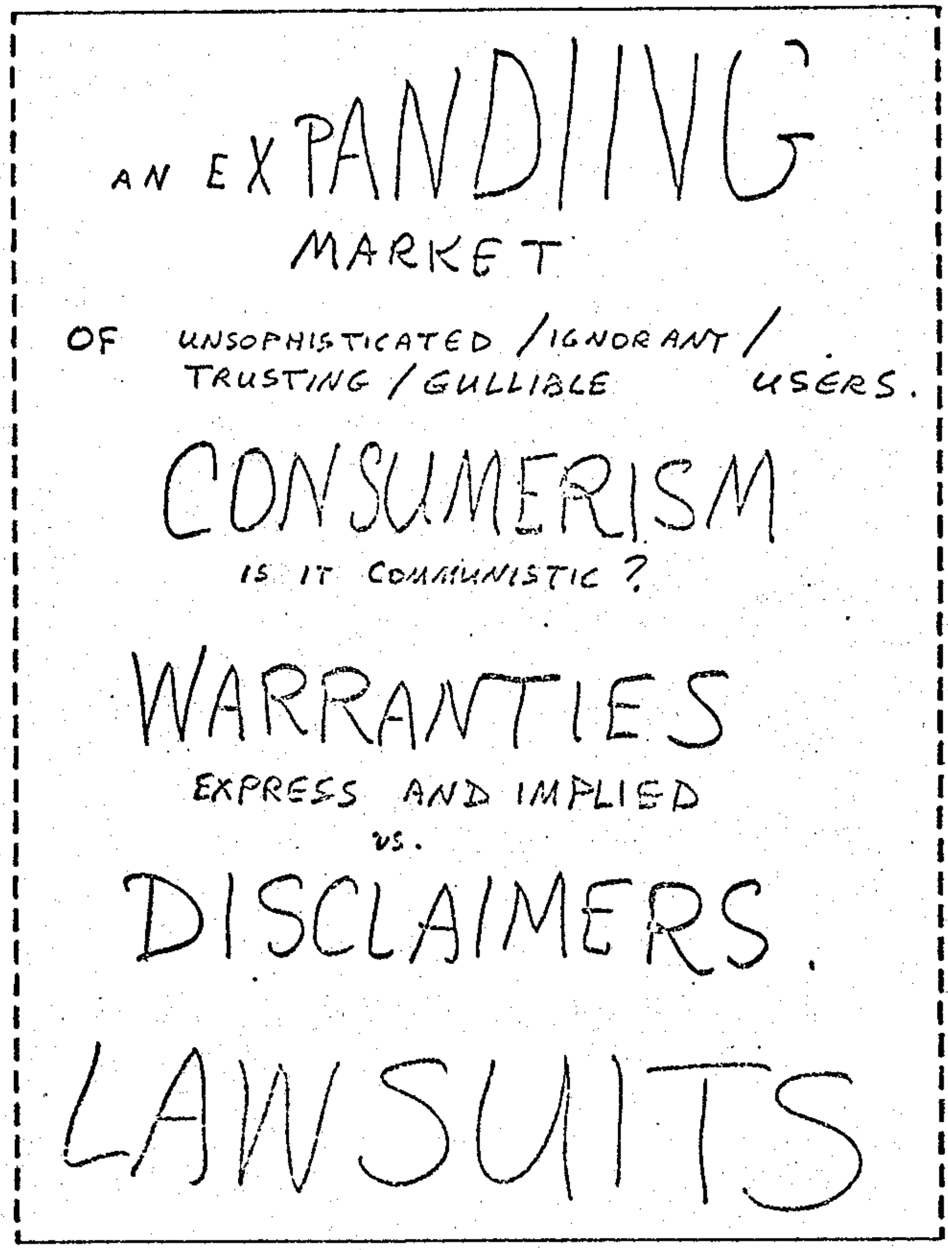

Figure 10

articles produced by other manufacturers behaved, and if he was disappointed, and without warning, then he has a just case.

An example would be the case against General Motors, concerning the Corvair, which would tuck its rear wheels in from time to time. Now that case appeared to be heading towards settlement against General Motors--I believe that was settled out of court. And the reason appeared to be that General Motors had neglected to issue a warning to potential purchasers of the car that it might from time to time tuck its wheels under. I imagine that would have done something to their market.

If you feel that this epic lawsuit is too extreme, I should point out that I know of some hand-held calculator types who are engaged in lawsuits now, probably relatively trivial ones so far, but we could speculate on the date in the future--I don't know when--when we might be greeted with the following kind of scenerio.

[Fig. 11] A businessman who flies his own light plane sets out from Sacramento late one afternoon and crashes in the evening in the hills of San Bernardino, only a little bit short of his destination. The crash is not a terribly serious one. He is killed, but out of the wreckage intact they find his shirt pocket calculator--still functioning--his scratchpad, his little pack of programs including the navagational programs he bought with the calculator, and they also find that the navagational equipment on the aircraft still functions, the dials are still set, and he'd probably done correctly. Alas, the calculator gave him a course which has brought him thirty miles off where he should have been, and so he hit a hill instead of the runway.

The widow, of course, is dismayed, and hires a clever lawyer who observes that if a different manufacturer's calculator, which is also programmable and also comes with software for handling the navigational problem, if <u>that</u> calculator had been used and if the same data had been entered, the man would have been off course again, but eight miles in the opposite direction, and would have landed ultimately successfully. Naturally, the widow is advised to sue, and does so.

The scene in court. The plaintiff brings forth witnesses, most of them employed by the other calculator company, to explain that in their calculator they carry thirteen digits, though they only display ten, and although they cannot always guarantee the correctness of the tenth, <u>usually</u>, because they carry thirteen, the results are more accurate. And, as far as this particular calculation is concerned, the experiment has already been performed and one can see what would happen.

The calculator company in defense brings <u>its</u> witnesses. For example, it can bring a prestigious numerical analyst who will explain, to the interested judge and jury, by using graphs with hyperbolas, that because of the intrinsic uncertainly in the man's data--that is, the last digit that he read could easily have been off one way or the other--and because the hyperbolas intersect in a grazing fashion, that it is extremely difficult to know exactly <u>where</u> they ought

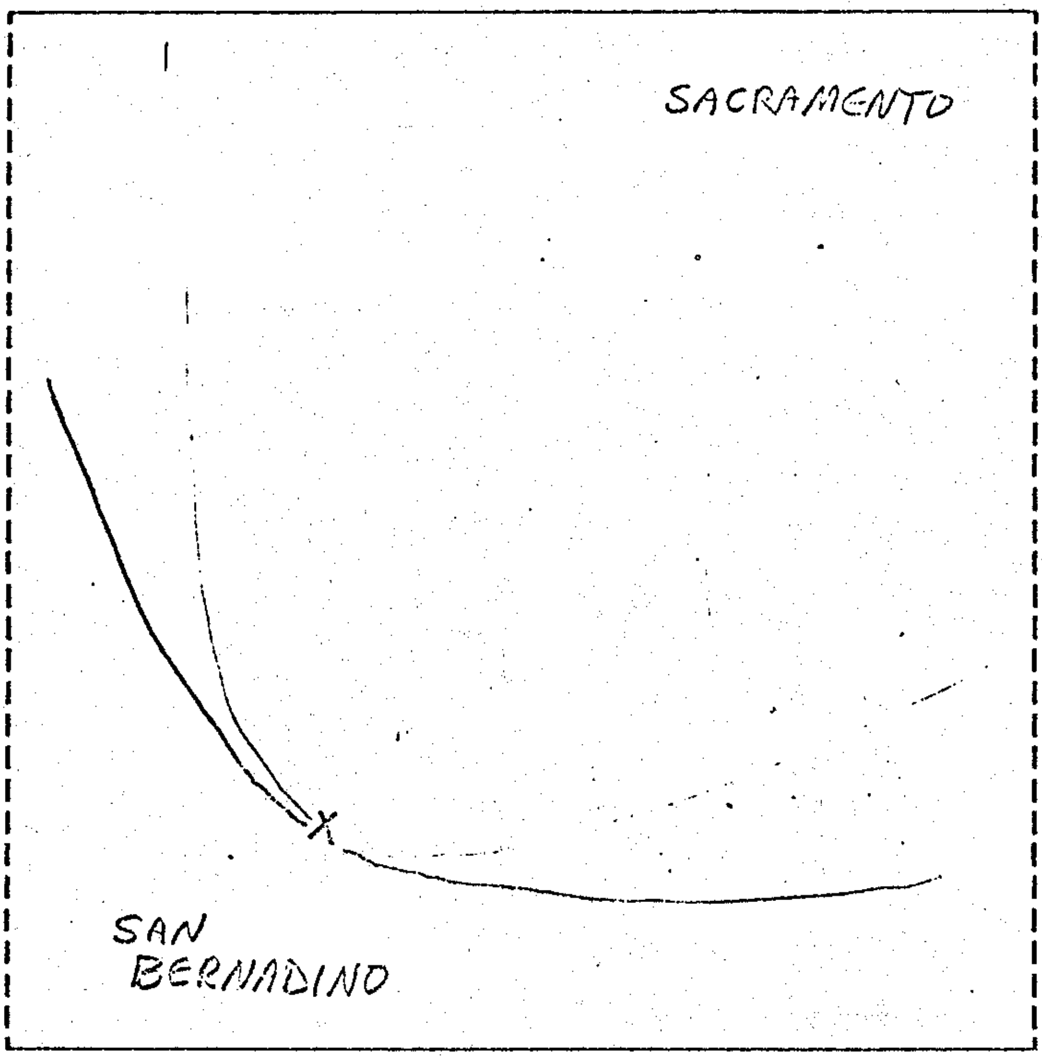

Figure 11

to intersect, and had his data been slightly different he could easily have been sent elsewhere.

Then, of course, there are others who testify that indeed the program used is of a kind standard in the industry, more or less, and no worse than most. [From audience: was it portable?]

In his summing up, the lawyer for the prosecution says that, "The defense has adduced certain hypothetical instances that the data might have been other than it was. We too," says the lawyer for the plaintiff, "we too could consider a hypothetical instance; the man might have bought the other company's calculator and, if so, he would be alive today."

In his summing, the lawyer for the defense says, "There are very important technical issues concerned in discerning what is and what is not possible in the realm of numerical calculation. To calculate this result more accurately is provably not possible."

The jury, while considering this summing up, is reminded by one of their member, who is an avid reader, that in the cargo of the first steamship to cross the Atlantic, was a book written by a Frenchman, who proved categorically that no steamship could ever cross the Atlantic.

The decision, of course, is rendered for the plaintiff. The stock of the computer company falls drastically. A few days later, stocks in other computer companies start to fail. I wonder what we should do. It is very likely that such cases will be determined by conditions that may have a rather dubious scientific validity at best.

Now, in this particular instance, there is a kind of defense. That is to say, one could realize that there are certain calculations whose results are so important that if they are uncertain some warning must be produced.

Such a warning could be produced by a properly implemented interval arithmetic, though it would take a certain amount of skill and care to do the job right. It's not so much that you do the whole calculation in interval arithmetic; that would probably be both pointless and unduly expensive. But you would at least need some kind of reasonable implementation of interval arithmetic (I would recommend my own) in order to display when a result is perhaps sufficiently uncertain that you want not to rely upon it for navagational purposes.

Now unfortunately, interval arithmetic has been slow in coming. Aside from the difficulties involved in its implementation, which difficulties arise principally because of hardware and software designs that suffer unbelievable inefficiencies when you try to make interval arithmetic work. For example, an interval arithmetic program merely to add or subtract on a CDC 6400 will take at least sixteen times as long as an ordinary add or subtract. At least that much longer. I fear that it may be a good deal longer than that when we finally get the bugs out. But I believe that problem could be addressed by correct design of hardware; in fact it certainly can be addressed much more easily in the hardware than anywhere else.

However, when someone tries to persuade the manufacturer of machines that he should have interval arithmetic, what normally happens is that among the various conflicting demands for the attention of his programmers are

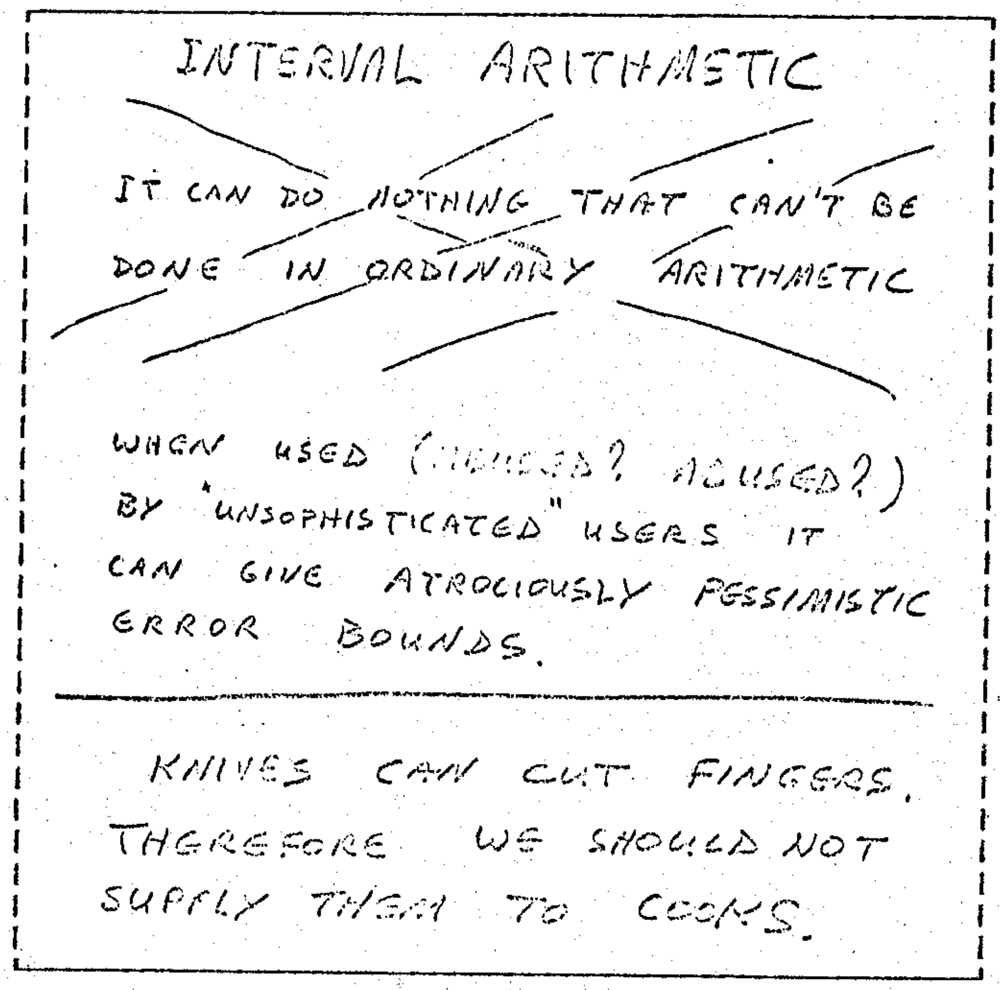

Figure 12

demands for any number of other things, and so the manufacturer will say, "Velvel, your arguments are impecable, your logic cannot be faulted; however, you have been outvoted."

Outvoted by people who may have used arguments like this. [Fig. 12] This argument is <u>false</u> to begin with. Directed rounding is something which is intrinsic in interval arithmetic and cannot be done if you do not build it into the hardware except at an impossible price. But in any event, if you use that argument, you could also take complex variables out of Fortran.

Or here's another argument: you can get atrociously pessimistic error bounds from interval arithmetic. Yes indeed you can, if you abuse it. Well, I suppose we could tell the cook that we have decided (in his best interests, of course) to take away his knives so that he won't cut his fingers. What do you think his reply will be? [Fig. 13]

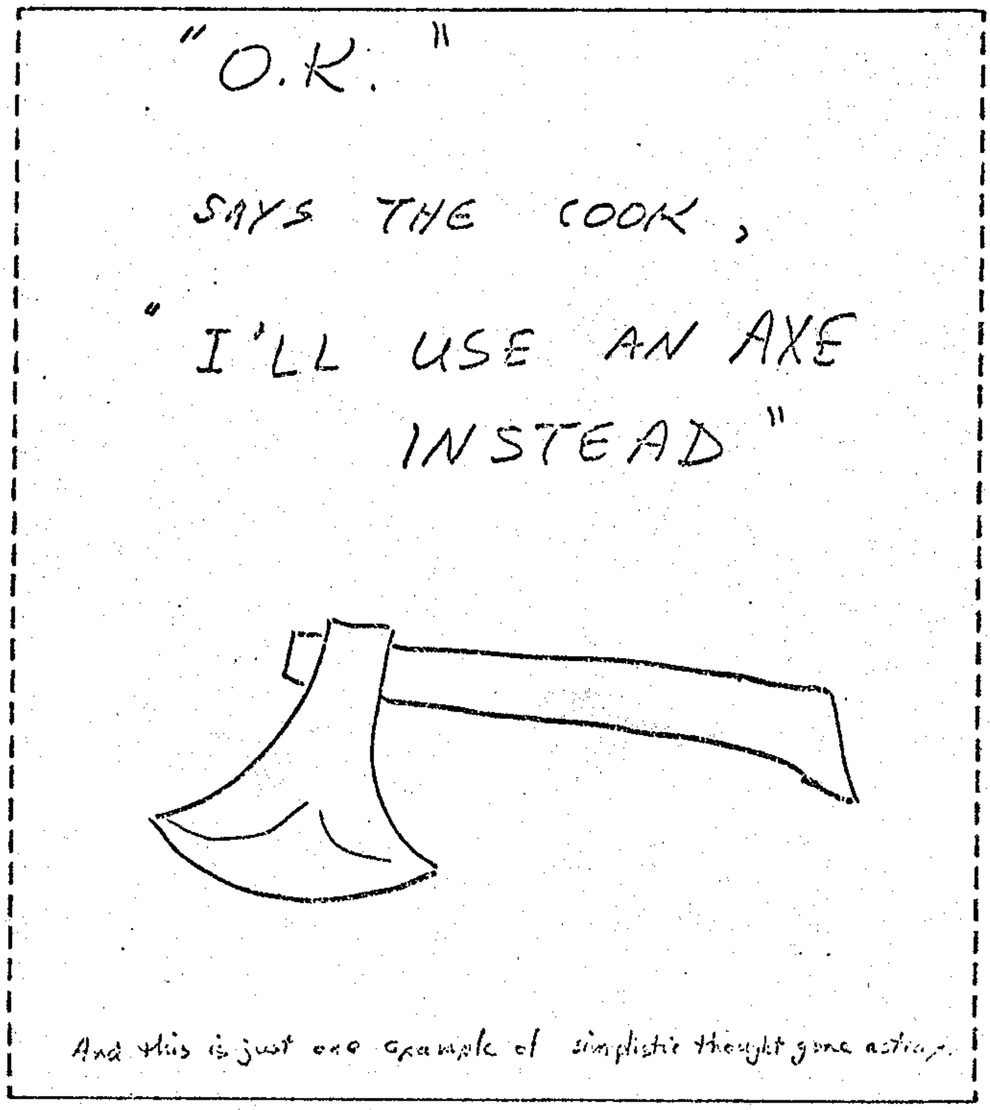

Figure 13

Well, I think I've really surveyed only a tiny bit of the nature of this problem; and that is that it's really rather easy for simplistic thought to go astray in this area. It takes a great deal of careful consideration of ramifications that can sometimes surprise you; ramifications which succumb to mathematical analysis up to a point. However, in order to do more, I should try to tell you what you ought to do, but as I say, I regret that in the time allowed I can't explain just what to do instead, any more than a surgeon could explain to you what to do in order to take out your own appendix. But at least I'll make myself available for questions now and hope that somebody can get some sort of information.

Q: Have you given up on significant digit arithmetic then? Is interval arithmetic the only way to find out error?

A: Significant digit arithmetic contains the seeds of its own demise in the following dilemma. [GAP IN TAPE] ... bounds, if they are bounds, grow exponentiallly faster than the error can as a function of time. Or, there exists a simple calculation whose alleged bounds decay exponentially faster than the error can grow. Or both kinds of calculations exist. And this occurs because significance arithmetic attempts to provide intervals which are such crude subsets of those provided in ordinary interval arithmetic as to make the difficulties in interval arithmetic look rather small, _except_ in certain special applications where the ranges of numbers happen to be modest. But significance arithmetic is, in my view, a dead loss, and I'm prepared, if you wish, to provide you with the proof of my allegations about the dilemma.

Q: Could you comment about the availability of software for interval arithmetic?

A: Well, I've got students working on my package, the documentation for it can be found in Michigan notes--I think you know about those--[From audience: Is it portable?] There is _no way_ to implement interval arithmetic in a portable fashion, nor in an efficient fashion, on any hardware currently being used on a large scale on this continent. The computers produced by Telefunken for the Karlsruhe group were produced with the ability to do directed roundings and made interval arithmetic extremely cheap to implement there, and consequently you find that the people who are most enthusiastic about interval arithmetic generally live around there. They have done a good deal, I think, of useful work. If we were to face, as a serious threat, the possibility that we might be sued because we had neglected to warn somebody that his answer, though correct for his data, was disturbingly sensitive to small perturbations, we would recognize how necessary it is to have something like interval arithmetic to do the sensitivity analysis, if only to display the results. And then perhaps we could put the squeeze on the manufacturers, not by lobbying, but by simply saying, "Well, if your computer won't support interval arithmetic in a reasonably economical way, then I'm afraid that I'll have to find something else like, say, a microprogrammable computer where I'll put the darned thing in for whatever it costs, but at least I know I'll have it, and I'll be safe from lawsuits." But, as you know, lawsuits are sort of capricious things, and they could easily go quite the other way.

Q: Yesterday you asked the question about how to solve the problem of writing numerical algorithm libraries for mini-computers. Nobody was able to give you an answer. I wondered if you in fact have some answers of your own to this problem.

A: Well, I think the crucial question is: Who will pay? If the users and the manufacturers of the mini-computer will pay what amounts to the salary of a reasonably well-educated man for about a year, he could almost certainly adapt the bulk of those programs in present libraries, but he will have to be extremely familiar not only with numerical software [EXCESSIVE NOISE ON TAPE]. I do not believe in portable subroutines at the deck level. I think that portability has to be at a level which is intelligible to humans and which offers something that looks almost like Fortran code, but is not quite that. And at <u>that</u> level you can make it portable without an undue intellectual investment. But if you try to make it portable at the deck level, then I think you are going to be wiped out. The costs are too great. Not only the costs, but when you write programs which are portable in this sense, the compromises that you make in order to achieve portability show up as <u>performance</u> penalties which leave your code vulnerable to a local man who will say "Look at all those library programs--I can beat those!" Now Gresham's law in computing is very simple. The <u>fast</u> program drives out the <u>slow</u>, even if the first one is <u>wrong</u>. So you cannot afford to pay both an intellectual penalty and a performance penalty, both of which penalties will turn up as costs or as customers lost in order to achieve portability which is, after all, just an abstraction, like justice. [TAPE RAN OUT.]