The optimum word length for a computer with a given speed and memory size depends upon the type of problems the computer is required to solve. Thus, for a general purpose computer word length determination becomes difficult because it is hard to predict what type of problem the machine will be expected to solve. This is further complicated by the fact that the knowledge of computational errors is sketchy for large classes of problems. The problem of matrix inversion has been studied (1), and will be used as a guide in what follows.

Given a problem there exists a precision p less than which the result is useless (or at least not very valuable). Each problem requires a given set of operations O_1, O_2, . . . O_n and each operation is carried out with a frequency f_1, f_2, . . . f_n. The maximum or average loss of precision (L_1, L_2, . . L_n) for each operation can be determined, however this may be a function of the data. There is also, roughly speaking, a factor R which is the measure of the recursiveness of the problem, that is, the number of times parts of the data are subjected to the various operations. Now given a set of instructions to be carried out and a precision p that is required, and the initial data it is presumably possible to determine the word length required to give the precision p. However, this result would not be very useful since it would apply to merely the particular initial data, and would not even apply to problems which used the same sequence of instructions, but differed in the initial data. Thus, one might attempt to determine a bound for the error assuming that the initial data is [redacted] subject to certain conditions. [redacted] This is essentially what was done in (1) and lacking results in other types of problems, I will make an out-of-the-hat comparison of other problems with matrix inversion and then discuss the relation of speed to word length.

It would seem that matrix inversion is at least as recursive as the average problem and probably more so. In general, it is possible also for the given operations to be as detrimental to precision as any. As to the frequency of such operations, it might be remarked that if a matrix is ill-conditioned precision is lost rapidly. Thus, it would seem fair to say that matrix ~~c~~inversion demands at least as much word length as a large class of problems.

If the size of a matrix is n x n then according to (1) an approximate inverse can be found if $n < .152^{s/4}$ where s is the word length, and for the time required we have $tp \sim n^3$. Thus, given two machines of speeds a_1 and a_2 for a fixed time T, the ratio of the orders of matrices that can be inverted in time T is $\frac{n_1}{n_2} = \left(\frac{s_1}{s_2}\right)^{1/3}$.

So word length is related to speed by $\left(\frac{a_1}{a_2}\right)^{1/3} = 2^{s_1/4} / 2^{s_2/4}$ or letting $a_1/a_2 = r$

we have $\log_2 r = 3/4\,(s_1 - s_2)$

or $s_2 = s_1 + 4/3 \log_2 r$.

From the point of view of memory, the amount of memory required is proportionate to the square of the ~~other~~ order of the matrix. (This essentially states the dimension of the problem. For some problems this gives a much higher power than 2.) Thus if two machines have memory sizes m_1 and m_2 respectively, we have

$$\left(\frac{n_1}{n_2}\right)^{1/3} = r, \quad \frac{n_1}{n_2} = \left(\frac{m_1}{m_2}\right)^{1/2} \therefore \frac{m_1}{m_2} = (r)^{2/3}.$$

On this basis one merely finds a machine which is well balanced considering the above and given the ratio of speeds find a machine which will be well balanced at the new speed. If this were done assuming that a 701 is well balanced and that the new machine 1024 times faster than the 701, we obtain a word length of 48 bits and a memory of 200,000 words.

Beckman

Since the word length is in most cases a great deal longer than the precision desired, one might consider the possibility of processes which successively approximate the solution. That is a process which is short at each stage and thus gives little error and, so if enough stages were used the precision might be almost as large as the word length. Thus, the relation of speed to word length should be discussed. ~~XXXXXXXXXXX~~

If an addition scheme were used which delayed long enough for all carries to be performed in the longest possible case, then addition time would be more or less linear with word length. However, current thinking is that the machine will proceed after the carries have stopped and thus the add time will depend only upon the average number of carries. ~~xxxx~~ Thus, the time goes up only as a function of $\ln_2 s$. Multiplication, if looked upon as a sequence of s adds, would then go up as s log s, however, again more sophisticated methods reduce this to something like $(\log s)^2$. On the basis of this, it seems unlikely that approximation methods look very good, for in general, they require much more time.

Martin A. A. [illegible]

In a floating point machine, the length of the exponent needs to be considered. Since it requires only a small number of bits for rather large scale factors, I believe that a rather large figure should be used as the additional cost would not be excessive. With respect to the base, it would seem that a base 2 is the best, for the gain in bits in no way compensates for the waste resulting from shifting in larger steps when other bases are used. The only advantage seen in using a base other than 2 would be the reduced number of shifts required before doing a floating pt. add, due to the higher probability that the exponent would be the same.

One factor which is well worth considering is using very short word length with the possibility of automatic or relatively easy multiple precision.

(1) Numerical Inverting of Matrices of High Order - J. VonNeumann and H. Goldstine. Bal. A.M.S. Nov. 1947, Vol. 53, No. 11