A Comparison of some Methods of Calculating Covariance Functions on an Electronic Computer

By I. J. Good

Let $a_1, a_2, \ldots, a_N; b_1, b_2, \ldots, b_N$ be $2N$ numbers, each of at most ν binary digits. We wish to calculate the $2M + 1$ "lagged products"

$$c_s = \sum_{r=1}^{N-s} a_r b_{r+s} \quad (s = -M, -M+1, \ldots, M),$$

for some $M \leqslant N - 1$. We suppose that N is large, and M not too small, and that ν is considerably smaller than the "word-length" of a computer. A comparison is made of three methods of organizing the calculation, in each of which several numbers are packed into a single word.

Introduction

Let $a_1, a_2, \ldots, a_N; b_1, b_2, \ldots, b_N$ be $2N$ numbers, each of at most ν binary digits, i.e. each is selected from $0, 1, \ldots, 2^\nu - 1$. We wish to calculate the $2M + 1$ "lagged products"

$$c_s = \sum_{r=1}^{N-s} a_r b_{r+s} \quad (s = -M, -M+1, \ldots, M),$$

for some $M \leqslant N - 1$. We suppose that N is large, and M not too small (see later), and that ν is a small integer. Such calculations arise, even with $\nu = 1$, in approximating the covariances of stochastic processes. (See, for example, Lomnicki and Zaremba, 1955). In fact, ν will be assumed to be considerably less than the word-length, ω, of the computer, so that it is natural to ask what advantage can be taken of the packing of several ν-bit numbers into a single word. Three methods of doing this will be considered.

Method I tries to take full advantage of the existence of the ordinary multiplier of the computer. Many electronic computers have expensive multiplying components in which much work is done in parallel, and it seems a pity to waste this large capital investment unless one is forced to do so. In this method both a's and b's are packed into words and these words are multiplied together. The method is an extension of a known method used on desk calculators and tabulators. (See, for example, Willers, 1947.) In Method II, several a's are packed into a single word, and at any time are multiplied by a single one of the b's. This method is logically simpler than Method I, but turns out to be slower. In both these methods it is necessary to interleave the packed numbers with zeros in order to avoid embarrassing carries. Method III is appropriate mainly for the case $\nu = 1$, and makes use of "logical" instructions.

The conclusions of the paper are stated at the end.

I am indebted to Mr. F. Steel for some useful discussions while preparing this paper.

Method I

Take some integer μ, where $\mu > \nu$; let $\omega = \rho\mu + \sigma$ $(0 \leqslant \sigma < \mu)$; and let

$$A_t = a_{t\rho+1} + a_{t\rho+2}2^\mu + \ldots + a_{t\rho+\rho}2^{\mu(\rho-1)} \quad (t = 0, 1, 2, \ldots),$$

$$B_u = b_{u\rho+\rho}2^\sigma + b_{u\rho+\rho-1}2^{\mu+\sigma} + \ldots + b_{u\rho+1}2^{\mu(\rho-1)+\sigma} \quad (u = 0, 1, 2, \ldots),$$

so that A_t and B_u can be regarded as polynomials in 2^μ of degree $(\rho - 1)$, except that B_u contains an extra factor of 2^σ, i.e. a shift of σ places to the left. A_t can be described as the result of packing ρ of the a's together into one word, each segment being of length μ, so that each of the a's is preceded by $\mu - \nu$ zeros. There are in addition σ "wasted" zeros in the σ most significant places. Similarly B_u packs ρ of the b's, with the wasted zeros in the *least* significant places, and also the order of the suffixes is reversed in B_u.

When the product of A_t and B_u is formed, the result, in the double-length accumulator, can be regarded as a polynomial in 2^μ of degree $2(\rho - 1)$, provided that there have been no embarrassing carries (i.e. the coefficients are all less than 2^μ after the multiplication).* A sufficient condition is that

$$2^\mu > (2^\nu - 1)^2\rho. \qquad (1)$$

It will be observed that the product A_tB_u has "wasted" digits in the σ least significant places and also in the $\mu + \sigma$ (not σ) most significant places of the double-length word, and it has one of the segments of length μ bang up against the left of the least significant half of the double-length accumulator. Thus none of the segments is divided between the halves of the double-length accumulator. The coefficients of powers of 2^μ in A_tB_u are seen to be equal to segments of the required sums $\Sigma a_r b_{r+s}$. It can also be seen, most easily by a "geometrical" argument, that in the multiplication of all A_t by all B_u, each product $a_r b_q$ occurs precisely once. In Fig. 1, the required sum, c_s, is equal to the total of a single "diagonal" of the whole array, and this is the sum of "diagonals" of the various squares. Moreover, half of these "sub-diagonals" are placed in identical places within their square, and the other half in the complementary places. Thus, we can give a simple rule

* Strictly, this polynomial is multiplied by 2^σ.

for defining the c's in terms of the C's defined by $C_S = \sum_R A_R B_{R-S}$.

Fig. 1 illustrates the case $N = 32$, $\rho = 8$. Each of the sixteen sub-squares contains $\rho^2 = 64$ *cells*. Within each cell is a product $a_r b_q$ ($r, q = 1, 2, \ldots, 32$). The shaded cells form a "diagonal" of the whole array, and the sum of the entries in this diagonal is $c_5 = a_1 b_6 + a_2 b_7 + \ldots + a_{27} b_{32}$. This is the sum of seven contributions, one from each of the sub-squares (A_1, B_1), (A_1, B_2), (A_2, B_2), (A_2, B_3), (A_3, B_3), (A_3, B_4) and (A_4, B_4). These contributions are, respectively, the coefficients of $2^{2\mu}$, $2^{10\mu}$, $2^{2\mu}$, $2^{10\mu}$, $2^{2\mu}$, $2^{10\mu}$, $2^{2\mu}$ in the polynomials in 2^μ which represent the seven products A_1B_1, A_1B_2, . . ., A_4B_4. More generally, the contributions would be coefficients of

$$2^{\tau\mu},\ 2^{(\rho+\tau)\mu},\ 2^{\tau\mu},\ 2^{(\rho+\tau)\mu},\ \ldots \quad (0 \leqslant \tau \leqslant \rho - 1),$$

or

$$2^{\tau\mu},\ 2^{(\tau-\rho)\mu},\ 2^{\tau\mu},\ 2^{(\tau-\rho)\mu},\ \ldots \quad (\rho \leqslant \tau \leqslant 2(\rho - 1)),$$

in which case the products would be A_1B_1, A_2B_1, A_2B_2, A_3B_2, . . .

It will be seen from Fig. 1 that, for example, the products A_1B_2, A_2B_3, A_3B_4, . . . (or any other products lying in a diagonal of sub-squares) can be added flat, provided that there are enough zeros to avoid undesirable carries. When k products $A_t B_u$ are added together, there will be no undesirable carries provided that

$$k\rho(2^\nu - 1)^2 \leqslant 2^\mu - 1, \tag{2}$$

a condition that swallows up condition (1). Therefore we can define k as

$$k = \left[\frac{2^\mu - 1}{(2^\nu - 1)^2 \rho}\right], \tag{3}$$

when $[x]$ is the largest integer not exceeding x. The condition (2) can be expressed by saying that $k \geqslant 1$.

After k such products have been added into the double-length accumulator, the $2\rho - 1$ coefficients of powers of 2^μ must be "distributed," in order to avoid undesirable carries. By "distribution" is meant that the coefficients are extracted, and added into suitable locations, corresponding to the various values of s. The coefficients can each be obtained by the logical operation of "masking." There are $2\rho - 1$ such masking operations required in each of the distributing operations. Also some shifting would be necessary, but it can be done rather rarely, except for the segment on the extreme left of the right-hand half of the accumulator. The $2\rho - 1$ additions are liable to be minimum-time for computers of variable addition time, since the addends each have only μ digits at the most.

The number of sub-squares is N^2/ρ^2, but only about $M(2N - M)/\rho^2$ are relevant *if M is reasonably large compared with ρ*. In estimating the running time we may ignore the input and packing, including the reversal of order of the b's, since this work is merely proportional to N. (But see the discussion below under "Inadequate store.") I shall also ignore "housekeeping" instructions

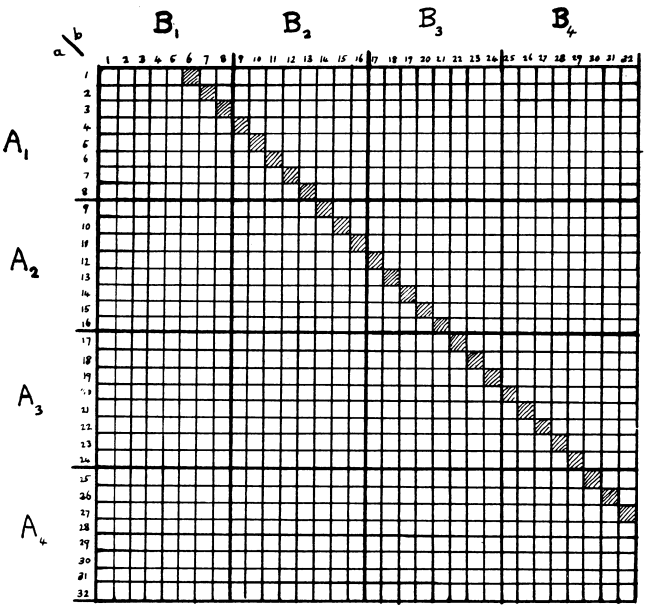

Fig. 1. The case $N = 32$, $\rho = 8$

on the assumption that "alarm-clock" facilities are available for counting (see the Appendix). If these facilities are not available, the following estimates will require substantial modification, but, for the sake of simplicity, I shall not discuss these modifications in detail.

The total running time will be approximately

$$\frac{M(2N - M)}{\rho^2}\left\{t_1 + \frac{2\rho(t_2 + t_3)}{k}\right\}, \tag{4}$$

if full alarm-clock facilities are available, where t_1 = time for a multiplication, with the product added into the double-length accumulator; t_2 = minimum time for an addition; t_3 = time for a masking operation.

Strictly, there is a little more time required for shifting, but it will be ignored here, since it is necessary all the time only for the numbers in the most significant segments of the words. I have in any case been a little unfavourable to this method since I have used the factor 2ρ instead of $2\rho - 1$.

For any given values of ν, ω, t_1, t_2, t_3, we can choose μ so as to minimize the time, (4). Note that for some consecutive values of μ, ρ (and therefore k also) has a fixed value; naturally we should select the smallest of such values of μ.

Method II

N; $a_1, \ldots, a_N$; $b_1, \ldots, b_N$; ν; and ω; are the same as in Method I, but μ, ρ, σ, k are to be replaced by μ', ρ', σ', k', where

$$\omega = \rho'\mu' + \sigma' \quad (0 \leqslant \sigma' < \mu'); \tag{5}$$

$$2^{\mu'} > (2^{\nu'} - 1)^2; \tag{6}$$

$$k' = \left[\frac{2^{\mu'} - 1}{(2^{\nu'} - 1)^2}\right] \geqslant 1. \tag{7}$$

There are about $M(2N - M)/\rho'$ multiplications, and $1/k'$ of these will be "distributing" operations. Hence, again assuming full alarm-clock facilities, the running time will be approximately

$$\frac{M(2N - M)}{\rho'}\left[t_1' + \frac{\rho'(t_2 + t_3)}{k'}\right], \tag{8}$$

where t_1' is the time for multiplication by a μ-bit word, and adding into a one-word accumulator.

Method III

This method is intended primarily for the case $\nu = 1$.

We pack the a's and b's with $\mu = 1$ and with the b's in direct order. We then use "logical" (i.e. bit-by-bit) multiplication. The effect is to produce ω relevant products, corresponding to a piece, ω cells long, of a diagonal of Fig. 1. The 1's in the product can then be added together by means of a "sideways-add" instruction, which counts the number of 1's in a word. In the whole program there would be about $M(2N - M)/\omega$ logical multiplications, the same number of "sideways-adds," and the same number of minimum-time additions. Therefore the running time would be about

$$\frac{M(2N - M)}{\omega}(t_4 + t_5 + t_2), \tag{9}$$

where t_4 = time for a logical multiplication; t_5 = time for a "sideways add."

Let us consider a hypothetical machine for which Method III could be applied with $\nu = 2$.

The logical multiplication would have to be replaced by what may be called segmented multiplication, in which two-bit numbers ("dibits") in each of $\omega/4$ segments ("tetrabits") of one word would be multiplied by the corresponding "dibits" of another word, the $\omega/4$ resulting products (tetrabits) being packed in the single-length accumulator. (Cf. Good, 1957.) The sideways-add instruction would need to be applicable to tetrabits instead of to just monobits, as it is in most existing computers. If these facilities were available, Method III with $\nu = 2$ would take just about four times as long to run as Method III with $\nu = 1$.

Numerical Estimates of Times

The estimates of times have so far been given algebraically, and they would vary considerably from one machine to another. But, in order to make the results more concrete, some specific figures will now be assumed for $t_1, t_2, \ldots, t_5$. Let us assume $\omega = 48$ and (in microseconds)

$$t_1 = 192,\ t_2 = 36,\ t_3 = 60,\ t_4 = 60,\ t_5 = 60,$$

$$t_1' = 192 \text{ if } \mu' > 6,\ t_1' = 108 \text{ if } \mu' \leqslant 6.$$

(These figures are probably about right for Orion.)

We then have the following table, in which each entry is to be multiplied by $M(2N - M)$ to give the approximate running time in microseconds. ("Alarm clocks" are assumed, and if not available some recalculations of these estimates would be necessary.)

Column II′ gives the times for Method II if $t_1' = t_1$, even when $\mu \leqslant 6$.

ν \ Method	I	II	III	II′
1	6	14	3·25	25
2	11	27	13	41
3	14	49	—	49
4	22	53	—	53

Inadequate Store

Finally we may consider the effect of the store's being not large enough to handle all the material in one go.

Suppose that the largest value of N that could be taken at once is N_0. Thus the machine can cope with N_0^2 multiplications without transfers to and from the store. Hence the total number of transfers must be of the order of $M(2N - M)/N_0^2$. If N_0 is large, so that the number of instructions in the program is unimportant, then this estimate is independent of whether we are using Method I, II or III. Therefore, if there is an inadequate but fairly large store, our main conclusions will not be much affected, although the ratios of the running times for the three methods will be made closer to 1.

Conclusion

Method III is best when $\nu = 1$, and Method I when $\nu \geqslant 2$, at any rate if the "alarm clock" instruction is in the computer. If the fast store is small a more careful analysis would be required, or the programs would even need writing in full, in order to estimate the running times.

Appendix: Alarm-clock Instructions

When we wish to repeat a loop a certain number of times, one method is to add or subtract 1 into a counter, and to test after each loop whether this counter has reached some value such as zero. We, so to speak, must keep one eye on the clock. If the loop is a very short one this wastes an appreciable proportion of time. It is, therefore, perhaps desirable that an electronic computer should contain several "alarm clocks." These would be registers somewhat analogous to the index registers or "B tubes." Each instruction would have, say, three spare digits, enough for seven alarm clocks and a null. An alarm clock would contain a counter from which 1 was subtracted when an instruction was obeyed in which that alarm clock was specified. When the counter in the alarm clock was reduced to 0, the

control would jump to the instruction at location x, where x is a number stored in another part of the alarm clock. At location x, a subroutine would start which would reset parameters, including those in the alarm clock.

The general effect of having alarm clocks would be that work done in counting out loops would be done in parallel with the rest of the calculation by means of additional circuitry, so that we would be trading circuitry for time. Part of this extra circuitry is the extra digitry. In a serial machine, the three extra digits per word would involve a proportional loss of time of $3/\omega$ in all machine programs, so that the case for alarm-clock instructions is better for parallel machines.

References

LOMNICKI, Z. A., and ZAREMBA, S. K. (1955). "Some Applications of Zero-one Processes," *J. Roy. Stat. Soc. Ser. B*, Vol. 17, p. 243.

WILLERS, F. A. (trans. by R. T. Beyer) (1947). *Practical Analysis* (Dover, New York), p. 55.

GOOD, I. J. (1957). "Variable-Length Multiplication," *Computers and Automation*, Vol. 6, p. 54.

Book Review

Automatic Language Translation, by A. G. OETTINGER (Harvard), 380 pages, price not given.

Automatic Translation, by D. YU. PANOV (Pergamon), 73 pages, 21s.

The best chapters of Dr. Oettinger's book are those describing the practical work done under his direction over the past few years. The Harvard group decided that a pre-requisite for automatic translation research was a thoroughly reliable automatic dictionary, to which reference would be made by the program for syntactic and other information about the words of an input text; they felt that, in the experimental stages of a translation program, so many unforeseen troubles would arise that progress would be almost impossible unless the researcher could have complete confidence in the reference stage of his program. In particular, since, in common with most groups in the U.S.A., they were proposing to work on Russian, it was necessary to develop very reliable methods for removing the endings of inflected Russian words, which could then be used with confidence that no unexpected anomalies would arise. In order to achieve this, the Harvard group were obliged to set up a new classification of Russian words, of such precision that the paradigm could be worked out exactly and not merely roughly from a knowledge of the classification of a word. The full paradigm was then produced mechanically, and each member of it subjected to the ending-removal algorithm. When each word was processed it was then possible to make an exhaustive test for failures and anomalies, and then to remove or note any that arose. All this work is thoroughly described, and anyone proposing to work in problems of mechanized linguistics would do well to study the Harvard experience at an early stage.

I am not so clear as to the usefulness of the remainder of the book. For instance, the chapter on the structure of signs concludes with an interesting discussion of the Cyrillic Unityper used at Harvard. Is it necessary to discuss set-theory, isomorphisms, use-mention, the identity of indiscernibles, the nature of models, the type-token distinction, and other such topics of college philosophy courses in order to understand this device, or indeed the program as a whole? While I would not deny that topics like these may come up in the ultimate analysis of language, Dr. Oettinger does not reach a point where they emerge and, in a work subtitled "Lexical and Technical Aspects," there would be no reason to expect him to refer to them. Why, therefore, was the material put in?

Professor Panov's book is the reverse of Dr. Oettinger's in almost every way. It describes the work on automatic translation, mostly English–Russian, done at the Institute of Precise Mechanics and Computing Technique and the Institute of Scientific Information of the USSR Academy of Sciences since 1955. It also discusses experiments now under way on the automatic translation of Chinese to Russian. Professor Panov starts with a discussion of the mechanism of ordinary (human) translation, and goes on to show how his workers have tried to represent in a program the various stages he describes.

In working from English, he had much fewer inflectional problems to deal with than Dr. Oettinger, and the dictionary compilation and reference methods are accordingly simpler. He has, however, to extract information about case in order to be able to construct the Russian output, and case information is not easily obtainable from English, which expresses it mostly by prepositions and word-order rules. His analytic procedures are all expressed very neatly as choice structures. For example, in the choice structure for English nouns, we find "5(6, 13) Test preceding word for 'let'."

This means that this is test 5; with a positive answer we proceed to test 6, with a negative answer to test 13. Words which have more than one Russian equivalent are given, as part of their dictionary entry, a special choice structure for selecting the right rendering. In effect, the dictionary contains a special subroutine for each recalcitrant word. This suggests that Panov believes that such words are something of a rarity; I cannot believe that this is so, and I fear that the program may become overburdened with large, rarely used choice structures. However, when the program has been extensively tested the answer will be known.

Professor Panov's book has been well translated, and is, probably on account of its short length and absence of frills, easier to understand than most accounts of experimental automatic translation programs.

Neither of these books should be omitted from the library of any computing laboratory where linguistic applications of computers are carried on or contemplated.

R. M. NEEDHAM.