Fox's method would produce the same results apart from possibly different rounding errors.

For

$$I = -\int_0^1 x \ln^3 x \, dx = \frac{3}{8} = 0{\cdot}375,$$

we find

$$T(h) - I = Ah^2 + Bh^2 \ln h + Ch^2 \ln^2 h + Dh^2 \ln^3 h + Eh^4 + Fh^6 + \ldots.$$

The Romberg columns are obtained from (2) with $\eta = 2, 2, 2, 2, 4, 6, \ldots$, and the results are shown in **Table 2.**

Reference

Fox, L. (1967). Romberg integration for a class of singular integrands, *The Computer Journal*, Vol. 10, No. 1, pp. 87-93.

Correspondence

To the Editor
The Computer Journal

Sir

'Packing' in FORTRAN

A commonly occurring problem in FORTRAN mathematical programming is the indexing (or subscripting) of 'sparse' multiple subscripted variables where the use of arrays of three, four or more dimensions is inefficient and prohibitively space consuming. One of the 'traditional' solutions used by quantum chemists and physicists[1] —who are often dealing with sparse four subscript quantities—has been to take a leaf from the commercial programmer's book and 'pack' the least significant bits of four or more integers into one real variable, and use this variable as a label. This has always been done via an Assembler routine CALLed from a FORTRAN program. However, modern FORTRAN compilers allow the use of single characters as logical variables and so this packing (and of course 'unpacking') can now be performed entirely in FORTRAN using appropriate EQUIVALENCES. I give below for your amusement annotated program fragments to pack and unpack four integers (0-255) into one real variable. Other character manipulations can be performed in the same way.

```
      LOGICAL*1 LOG1(4), LOG2(8)
      INTEGER*2 ID(4), I, J, K, L
      EQUIVALENCE (WORD, LOG1(1)), (ID(1), LOG2(1))
      EQUIVALENCE (ID(1), I), (ID(2), J) (ID(3), K), (ID(4), L)
C     THESE EQUIVALENCES SET UP THE CHARACTERS
      .
      .
C     THE FOLLOWING FOUR STATEMENTS PACK I, J, K, L
        INTO WORD
      LOG1(1) = LOG2(2)
      LOG1(2) = LOG2(4)
      LOG1(3) = LOG2(6)
      LOG1(4) = LOG2(8)
      .
      .
C     THE FOLLOWING FOUR STATEMENTS UNPACK
        WORD INTO I, J, K, L
      LOG2(2) = LOG1(1)
      LOG2(4) = LOG1(2)
      LOG2(6) = LOG1(3)
      LOG2(8) = LOG1(4)
      .
      .
```

Yours faithfully,

D. B. Cook

Department of Chemistry
The University
Sheffield S3 7HF

7 June 1972

[1]See, for example, 'The POLYATOM System' by I. G. Csizmadia, M. C. Harrison, J. W. Moskowitz, S. Seung, B. T. Sutcliffe, and M. P. Barnett, M.I.T.

To the Editor
The Computer Journal

Sir

The postage stamp problem

With reference to W. F. Lunnon's article (this *Journal*, Vol. 12, p. 377) I should like to report the solution V(10, 3) = 155. The original run using techniques developed from Lunnon's work took 250 Plessey XL4 processor hours, but with further sophistication a re-run was accomplished in only 99 hours. The calculations show a unique set of stamps, 1 2 6 8 19 28 40 43 91 103.

It has been reported to me by M. L. V. Pitteway that V(8, 4) = 213. Apparently this ran in approximately 100 low priority hours on a KDF 9; the stamp denominations themselves have unfortunately been mislaid.

Yours faithfully,

J. L. Seldon

Squadron Leader, Royal Air Force

7 June 1972