Some explanation of these results is necessary. Barraclough's definition of inadequate spacing is the percentage of periods on the 'wrong day'. A period is defined as being on the wrong day if it occurs on the same day as another period in the same subject and if there is also a day which contains no period in this subject. The % teacher periods required is given by

((number of teacher periods required)/(number of teacher periods available)) × 100%

The high percentage of sets or double periods have different causes in the two problems.

It is rather difficult to apply the concept of setting to the RMCS case. For the purposes of compiling Table 3 we have assumed that setting occurs whenever a 'course' of students (e.g. 2nd year Engineers) is split in different ways for different subjects. The high figure given for the RMCS for this parameter is almost entirely due to 'setting' as we have few double period lessons. The school by contrast has very little setting but a high percentage of quadruple, triple and double period lessons.

15. Acknowledgement

We would like to acknowledge the early work done on the timetabling problem by Captain J. Dennis, REME (Dennis, 1968).

References

ALMOND, M. (1966). An algorithm for constructing University timetables, *The Computer Journal*, Vol. 8, pp. 331-340.

BARRACLOUGH, E. (1965). The application of a digital computer to the construction of timetables, *The Computer Journal*, Vol. 8, pp. 135-146.

COX, N. S. M. (1969). Operational requirements for computer timetabling systems for schools, Proc. of Glasgow programmed learning conference 1968, pp. 76-78.

CSIMA, J., and GOTLIEB, G. C. (1964). Tests on a computer method for construction of school timetables, *Comm. Assoc. Comp. Mach.*, Vol. 7, No. 3, pp. 160-163.

DENNIS, J. (1968). The application of a digital computer in formulating the College timetable, RMCS Internal report.

GOTLIEB, G. C. (1962). The Construction of Class Teacher timetables, Proc. of IFIP Congress 1962, pp. 73-77.

JOHNSTON, H. C., and WOLFENDEN, K. (1968). Computer aided construction of school timetables, Proc. of IFIP Congress 1968, pp. 83-86.

LIONS, J. (1967). The Ontario School Scheduling Program, *The Computer Journal*, Vol. 10, pp. 14-20.

YULE, A. P. (1968). Extensions to the heuristic algorithm for university timetables, *The Computer Journal*, Vol. 8, pp. 360-364.

Correspondence

To the Editor
The Computer Journal

Sir,

It was with great interest that I read Messrs. Cranley and Patterson's article on the automatic numerical evaluation of definite integrals (this *Journal*, Vol. 14, No. 2, May 1971). However, it seemed to me that their aim of guaranteed accuracy within stated limits is impossible given their presentation of the problem. They themselves admit that no finite set of values of a function in an interval can give us any certain knowledge of the value of the integral (assuming that the function is in fact integrable in any sense). This is true even if the function is known to be continuous.

The fact that the usual integration methods can be so widely used is due to the fact that most functions to be integrated are sufficiently well behaved, and the answers are usually subject to some human checking which weeds out the pathological cases.

However, I quite agree with Cranley and Patterson that fool-proof methods ought to be available, and it seems to me that Riemann integration provides the answer. That is to where one establishes in each sub-interval of the given interval an upper and a lower bound of the function. From these one can form an upper and a lower sum, and the value of the integral is guaranteed to lie between these. By repeated subdivision of the intervals in some suitable way one may obtain whatever degree of accuracy is desired.

The problem is then to establish bounds of the function in an interval. Let us assume that the function definition is in the form of some composition of simple functions with known properties. Then in general it should be possible to find the bounds simply be finding extreme values in the interval of each term, factor or argument. These must be put together by routines which take account of the properties of the elementary functions from which the function definition is composed. Addition, subtraction, multiplication, pose few problems. Division will fail if the upper and lower bounds of the devisor have opposite signs. Circular and other functions are all fairly straightforward. If the bounding process fails the interval must simply be subdivided until it succeeds, or the integration is abandoned.

For this method to be practically usable, the function to be integrated would not be provided in the usual way. Normally one makes available to the integration routine a procedure or subprogram which calculates the value of the function at any required point. The best way to implement this method, however, would be to make available the function in symbolic form, and to work in ALGOL 68 or in LISP, which are built for symbolic manipulation.

As described, the method would I believe be workable, but enormously inefficient. However, some improvement is possible. Usually the function will be differentiable, and symbolic differentiation is easy in the appropriate languages. If in some perhaps large interval the differential coefficient can be shown by the above method to be always positive or always negative then the function itself is monotonic, and its bounds in any interval are given by its end-values. Thus over any such interval we can subdivide as required and quickly form the upper and lower sums.

By considering second and third derivatives one may even be able to justify the use of trapezium rule or Simpson's rule approximations, but now as guaranteed upper or lower bounds, as the case may be, rather than just as approximations.

The method has no real hope to rival the conventional methods in efficiency, and might also sometimes fail to converge to the accuracy demanded, but it would never give spurious convergence, which cannot be said for any of the usual ways.

I hope to implement a program on these lines in due course.

Yours faithfully,

S. H. VALENTINE

Department of Computing
North Staffordshire Polytechnic
Beaconside
Stafford
18 May 1971

To the Editor
The Computer Journal

Sir,

While agreeing whole-heartedly with the ideas of Messrs. D. G. Evershed and G. E. Rippon (High level languages for low level users, this *Journal*, Vol. 14, No. 1), I would like to ask how they would parse the statement

LOOP K, TOTOTOBYBYBY

in the form

LOOP K, I TO N BY M

without introducing further arbitrary rules about the use of blanks to confound the 'low-level user'.

Yours faithfully,

R. C. BACKHOUSE

Department of Computing and Control
Imperial College of Science and Technology
48 Prince's Gardens
London SW7
29 March 1971