Clearly, line generation by exploiting repeated patterns is applied at the line generation stage rather than the output stage (although this inhibits the straightforward implementation of drawing in broken format). For curves, no simple parallel formulation at the generation stage would appear to be possible, other than that similar to the implementation at the output stage (i.e. corresponding to encoding repeated codes for a graph containing curves).

The decision whether or not to cater for curves by compression depends strongly on the expected benefits, the latter being related to the proportion of code information corresponding to curves and also the extent to which such information can be compressed in general. The analysis of the code information for a graph containing curves (**Fig. 12**) showed that (excluding the drawing of the border) the compression of repeated codes ($>$ 5 identical codes in sequence) gave a saving of 19% in the total space occupied by the code information (17894 codes). This corresponds principally to the drawing of the curves. A more detailed analysis of 6058552 codes corresponding to curve drawing information only, gave a saving of 17% by the compression of more than five repeated codes in sequence. The results of the analysis of the coding for Fig. 12 are therefore not unrepresentative of the general overall savings that can be made for curves.

It is possible to implement compression of curve drawing codes by duplication of the output routine; one copy then being used specifically for curves. The modifications for the compression of repeated codes can then be inserted into this output routine only, and will not apply to all other output which is routed through the usual (unchanged) output routine. In this way, the additional central processor overheads necessary for the compression will only apply to the time spent in generating curves rather than the time to generate the complete graph. As the former is 10% (on average) of the total central processor time spent in code generation, the overheads introduced thus apply only to this 10%. The overall increase in the central processor time for this modification is therefore of the order of 2%.

4. Conclusions

Line tracking by compression of repeated patterns gives a significant compression of the information required for the graph ($\sim$49%) for those cases using straight lines only, and more importantly, gives a significant decrease ($\sim$20–40%) in the overall central processor time needed to generate the lines.

These algorithms have been incorporated into a general purpose graphics system (Earnshaw, 1976).

The algorithms presented in this paper can readily be extended to cater for incremental plotters with more than eight vector modes.

5. Acknowledgements

The author would like to thank John Boothroyd and David Holdsworth for helpful discussion and the referee for constructive suggestions.

References

Boothroyd, J. (1974). Private communication.

Boothroyd, J. and Hamilton, P. A. (1970). Exactly reversible plotter paths, *Australian Computer Journal*, Vol. 2, No. 1, pp. 20–21.

Bresenham, J. E. (1965). Algorithm for computer control of a digital plotter, *IBM Systems Journal*, Vol. 4, No. 1, pp. 25–30.

Dijkstra, E. W. (1976). *A Discipline of Programming*, Prentice-Hall, Englewood Cliffs.

Earnshaw, R. A. (1976). Graph plotting in ALGOL 68–R, *Software Practice and Experience*, Vol. 6, No. 1, pp. 51–60.

Knuth, D. E. (1969). *The Art of Computer Programming*, Vol. 2, Seminumerical Algorithms, Addison Wesley Publishing Company, pp. 293–338.

Pitteway, M. L. V. (1967). Algorithm for drawing ellipses or hyperbolae with a digital plotter, *The Computer Journal*, Vol. 10, pp. 282–289.

Pitteway, M. L. V. (1972). The impact of computer graphics, *Nature*, Vol. 235, pp. 83–85.

Stein, J. (1967). *J. Comp. Phys.*, Vol. 1, pp. 397–405.

Stockton, F. G. (1963). Algorithm 162, *CACM*, Vol. 6, No. 4.

Thompson, J. R. (1964). Straight lines and graph plotters, *The Computer Journal*, Vol. 4, No. 3, p. 227.

Book review

SIMULA BEGIN (Revised Edition) by G. M. Birtwistle, O-J. Dahl, B. Myhrhaug and K. Nygaard, 1979; 391 pages (paperback). (*Input Two-Nine*, £6·50)

This book is 'an introduction to system description in the programming language SIMULA' according to its preface. It is not very clear to what audience the book is directed—the preface states that it is based on material developed from SIMULA courses given at the Norwegian Computing Centre. Clearly, it is of not much use on its own, but requires a context of a fairly solid course in software engineering.

SIMULA is rather a strange language: its origins are ALGOL 60 (indeed when the reviewer was participating in the revision of ALGOL 60, SIMULA was proposed as a potential standard). It has incorporated ideas from 'ALGOL-W', but placed side-by-side with languages such as ALGOL-68 and Pascal, it seems to be very incomplete. As its name implies, SIMULA is mainly used as a simulation language, but this book is not a book on SIMULA as a simulation language, but, again, its intention is unclear.

The would-be reviewer always faces the problem of trying to understand the way in which the book is presented to its putative audience; as a rule, the reviewer is not in this audience, and a certain amount of imagination has to be employed. The reviewer found it very difficult in this case: presuming himself to be in the situation of giving a course based on this book, he would have given up in despair! The contents of the book, taken as a whole, are very useful but the sequence is confusing, and at times, misleading. The reader is introduced to SIMULA concepts in an erratic way: odds and ends of SIMULA are thrown into the text in a completely unstructured way and the SIMULA class concept and co-routines only emerge late in the book.

However, the examples provided in Chapter 7 onwards are excellent and, indeed it might be a good idea to start reading the book at this point (page 209, more than halfway through). There are many exercises, with over 30 pages of solutions, though the commentary is highly variable.

The book is attractively presented with ample inner margins. The examples are both in rather broadly spaced printing with strop-words in italics (difficult to read) and computer printed output (with rather erratic tabulating conventions). R. M. De Morgan (Reading)