```
      SUBROUTINE SUMI (N,XN,XI,IKS,KE,LR,PL,SSU)
      DIMENSION XN(1),XI(1),IKS(1),LR(1),PL(1)
C
      SSU=1.
      DO 60 IE=1,KE
      J= (LR(IE)-1)*N
      S=0.
      ISK=IKS(IE)
      GOTO (10,30),ISK
   10 DO 20 I=1,N
      J=J+1
      S=S + PL(J)*XN(I)
   20 CONTINUE
      GOTO 50
   30 DO 40 I=1,N
      J=J+1
      S=S + PL(J)*XI(I)
   40 CONTINUE
C
   50 SSU = SSU*S
   60 CONTINUE
      RETURN
      END
```

References

Himmelblau, D. (1972). *Applied nonlinear programming*, McGraw-Hill.

Fiala, F. (1973). Algorithm 449—Solution of Linear Programming Problems in Q—1. Variables (M), *ACM*, Vol. 16 no 7.

Kelley, J. (1960). The Cutting-plane Method for Solving Convex Programs, *J. Soc. Ind. Appl. Math.* Vol. 8, p. 703.

Mottl, J. (1971). Optimalizace parametrů rypadla, *Strojnický časopis* č, Vol. 3, SAV Bratislava.

Book reviews

TEX and METAFONT, New Directions in Typesetting by D. E. Knuth, 1979; 305 pages. (*American Mathematical Society and Digital Press*, £8·60)

As one who has served ten years' hard labour editing papers which included the setting of almost anything and everything from Old Icelandic via ALGOL programs to mathematics, it was with a feeling of blessed relief that I read Donald Knuth's book on *TEX and METAFONT*. The author of *The Art of Computer Programming* has diverted his attention to the art of mathematical type design (METAFONT) and a system for the composition of technical texts (TEX—tau epsilon chi; signifying both art and technology) with the original objects of ensuring that not only could the second edition of *Seminumerical Algorithms* be set in the same type style as the first but also other volumes and editions, transcending a possible succession of printers through the years and changes of their inhouse equipment. In so doing he has made available to all those involved in publishing a system that combines elegance with simplicity 'intended for the creation of beautiful books'.

TEX and METAFONT comprises three discrete sections: 'Mathematical Typography', the 1978 Gibbs lecture delivered to members of the American Mathematical Society and the current versions of the TEX and METAFONT user manuals. 'Mathematical Typography' illustrates how Donald Knuth has applied his mind to the problem with characteristic thoroughness, the solution to which may embody more mathematics than some readers can comprehend, yet the system is designed in such a manner that the end user need be unaware of the underlying principles. He describes the typographical research on which he has been engaged, thus setting the scene for his own design and composition methods. A perfectionist, he is able to accept the quality from high resolution raster scan devices, even when armed with a magnifying glass, simply because the viscosity of ink is such that it smooths otherwise jagged edges. In METAFONT each character is specified in a matrix of 0s and 1s (no ink: ink), character design being based on cubic splines. A complete font is built up from a series of shapes common to several characters; other founts of the same alphabet type (for example bold and italic) and other styles may be created by subtle alteration of the parameters. Variation in line width is achieved by different sizes and types of 'pens' (circular and elliptical) which are used in conjunction with 'erasers'. To design a new character a user would write a program in this language for describing pen and eraser strokes; even the pen type may be specified—for ideograms an elliptical pen with rotational properties is suggested.

The composition is done by TEX, a unified system able to deal with 'ordinary' text as well as more sophisticated requirements. It works on the principle of rectangular boxes and glue where the smallest box may consist of part of a character or even a line. The glue is elastic in nature, the amount varying dependent on the space available, taking into account three parameters specified with each box—the 'normal' space, the maximum (stretch), and the minimum (shrink). A series of boxes may be built up with glue to form horizontal and vertical lists, and thus a page. In composing a line of text, possible word division is taken into account; similarly, in page makeup a check is made to ensure that no partial line is left stranded on the following page (a widow). The system is generalised in that there are no separate routines for various operations; it is extensible, however, and a set of macros may be written to deal with a particular house style as is being done by the American Mathematical Society for their publications. One beauty of TEX is that text may be keyed in by a typist after only a few hours' instruction to result in high quality composed output; but flexibility means that it can be used too in a more complex yet uncomplicated way, dependent on what is needed.

Donald Knuth's infectious enthusiasm pervades the manuals which are written in a delightfully easy style (complete with JOKES), illustrated with a wealth of examples, conspiring to make the reader want to try out the system. (Elsewhere I read that it is coded in SAIL for Digital systems (DECsystem-20 and PDP-10) and a Pascal version should now be available for implementation on IBM 370 and CDC CYBER machines. Memory requirements are relatively large, and use is made of the host file system. The device independent output file must be interfaced to appropriate raster scan hardware. As an added incentive to adopt the language the software is available for a handling charge only.) I regard TEX and METAFONT as a break-through in publishing methodology. But do not take it from me: read the book, use the software, and join TUG (the TEX Users' Group)—contribute to the development of the language and founts available. If, in a few years' time, there is a proliferation of mediocre languages for this purpose, the fault, gentle reader, will lie not in Donald Knuth but in ourselves.

Joan M. Smith (Manchester)

Writing Interactive Compilers and Interpreters by P. J. Brown, 1979; 260 pages. (*John Wiley*, £9·75)

This book is a good read not only for compiler writers, but for others too: those using or choosing compilers could learn much from this book, as could designers of any interactive program. Rival authors should be interested in the style.

The book deals with the problems to be overcome when implementing an interactive language like BASIC. No previous knowledge about compiler design is assumed. There are eight chapters entitled: Planning, The structure of a compiler, The design of an internal language, The run time system, Other modules, Testing and issuing, Some advanced and specialised topics. The coverage is good, particularly on the hazards of hasty planning, bad design and the neglect of error diagnosis. On the technical aspects of design, the author is practical, introducing theory only as necessary; he does not concentrate on the areas of grammar and parsing, but explores the less well understood aspects of design, for example in storage management and dictionaries. In such places, several alternatives are critically discussed. Apart from style, this book's virtue is its intelligent discussion of the many problems of design that arise.

The style is fluent and easy to read: there is no padding. The author never wastes space by stressing the obvious, and by giving each topic its necessary space has packed a lot of useful information into his book. This reviewer recommends it.

A. E. Glennie (Reading)