Euler's conjecture about the non-existence of two orthogonal Latin squares of order $4t + 2$. *Proc. Nat. Acad. Sciences of USA*, Vol. 45, p. 734.

DENES, J., and KEEDWELL, A. D. (1974). *Latin Squares and their Applications*, London: E.U.P.

FISHER, R. A., and YATES, F. (1934). The six by six Latin squares, *Proc. Camb. Phil. Soc.*, Vol. 30, p. 492.

FISHER, R. A., and YATES, F. (1963). *Statistical Tables for Biological, Agricultural and Medical Research*, 6th edition, Edinburgh: Oliver and Boyd.

GOLAY, M. J. E. (1958). Notes on the penny-weighing problem, lossless symbol coding with non-primes, etc. *IRE Trans.*, Vol. IT-4, p. 103.

MANN, H. B. (1943). On the construction of sets of orthogonal Latin squares. *Ann. Math. Statist*, Vol. 14, p. 401.

PARKER, E. T. (1959). Orthogonal Latin squares. *Proc. Nat. Acad. Sciences of USA*, Vol. 45, p. 859.

To the Editor
The Computer Journal

Sir

Saving CPU time in Scientific computation

I was interested to read the letter written by P. J. Hathaway and D. Van Vliet in the May issue of *The Computer Journal* in which they showed that by using a very simple subroutine a considerable time could be saved when handling large arrays.

What struck me with equal force was the fact that for non-professional mathematicians and programmers like myself who learn their elementary programming from an introductory text-book is that time-saving dodges and methods are rarely mentioned. This state of affairs is fine until the CPU time required for any program becomes large.

I would like to suggest that a page or so of every issue of *The Computer Journal* be devoted to time-saving methods in computing and for ALGOL and FORTRAN in particular.

I am sure that any such articles would prove useful to many readers.

Yours faithfully,
J. P. N. EDWARDS

Velindre Hospital
Whitchurch
Cardiff CF4 7XL
8 January 1975

Editor's comment:
The *Journal* will welcome any contributions along the lines suggested by Mr. Edwards. Whether or not such contributions will justify one page of every issue will depend on the number received.

To the Editor
The Computer Journal

Sir

Comparison of hybrid and digital computation performances for distillation simulation

The value of a comparison only has significance if the results used are for the same system under the same conditions. Comparing digital and hybrid simulations is a difficult task since the equipment available, the user charge rate, and the type of problem chosen each have a significant affect on the result.

The conclusion that the time advantage of 2:1 in operating speed of the hybrid is only 'marginally faster' than the digital would have a great significance when a statistical property (requiring a large number of runs) was being investigated.

The comparison has been made on 'solution time', and one hybrid solution time has been compared to seven digital solution times which have a range of 150:1.

A typical hybrid simulation has three main factors in determining its solution time, these can be expressed as:

$$T = t_1 + t_2 + t_3$$

where T total solution time
t_1 time to establish initial conditions
t_2 time of hybrid computation
t_3 time for data analysis and output,
t_2 can be subdivided into

$$t_2 = t_4 + (m + n)/100 * (t_2 - t_4)$$

where t_4 time of data transfer between analogue and digital
m per cent of remaining solution time, analogue waiting for digital
n per cent of remaining solution time, digital waiting for analogue.

The total solution time is therefore

$$T = T_1 + (m + n)/100 * (t_2 - t_4) + t_3 + t_4 \ .$$

The parameters in this expression require consideration before embarking upon preparing a problem (second order effects are ignored).

t_1 is determined by the component allocation
t_2 is determined by the digital/analogue division of the problem
t_3 is determined by the output peripheral speed
t_4 is determined by the machine configuration
m is determined by the machine configuration
n is determined by the analogue time scale chosen (normally $n \ll m$)

From this list it is evident that the machine used for the hybrid simulation will have a great effect on the solution time.

Experience in use of hybrid computers has shown that the limiting factors in the operating speed are the digital operating speed and the interface transfer time. Hence the comparison of one hybrid solution time (utilising an early analogue modified into a hybrid) without giving any indication of the various component times, against seven digital solutions is not a fair one.

Yours faithfully,
D. B. CROMBIE

20 Silecroft Road
Luton
Bedfordshire LU2 0RN
20 December 1974

To the Editor
The Computer Journal

Sir

Ballard *et al* (1974) have presented an interesting algorithm for the numerical solution of constrained generalised polynomial programming problems. May I make some points concerning this article and about the algorithm and test problems described.

It was unfortunate that numerous misprints marred the presentation. For example, equation (9) defining the ω_{0t} weighting factors is not correctly normalised to unity, equation (20) has a σ_{mt} instead of the correct C_{mt} and incorrect subscripts on a_{mtn}. Also, cross references to the numbered formulae throughout are confused.

A fundamental restriction on the algorithm in its present form is that only those problems which have all the primal constraints active at the optimum may be solved. I note that all five sample results fall into this restricted category, whereas, in most practical applications of polynomial programming, slack constraints abound (Bradley *et al*, 1974) and cannot be eliminated a-priori.

Should the algorithm be applied to a problem involving slack constraints, an examination of the matrix of **Fig. 1** in the article shows that terms of the form $1/\overline{\Omega}_{mt}$ and $1/\overline{\lambda}_m$ tend to infinity, while those on the right hand side of the form $\ln(\overline{\Omega}_{mt}\sigma_{mt}/C_{mt}\lambda_m)$ will become indeterminate as the optimum is approached. Ad-hoc corrections of the type indicated in Section 6 are not, in our experience, satisfactory if both reasonable CPU times and fairly accurate answers are desired.

Various methods are being used to overcome this problem. The most successful involve either the use of slack variables as in linear programming or the progressive identification and elimination of the slack primal constraints by using properties of the dual programme.

Unfortunately, only one of the test problems given involves the minimisation of negative term (and hence non-convex) polynomials, i.e. problem number four. The results quoted in the paper (and there could be a simple misprint) cannot be correct since substitution of the optimal primal variables into the primal constraints yield the following;

Value of constraint number 1 = 0·9663
Value of constraint number 3 = 1·3102