reconstituted an implementation of the recursive algorithm, is to make a leap of insight. It’s not at all obvious; which it would be, one supposes, if the recursive algorithm had meant such a process all the time.

I feel that an overemphasis on the ‘problem reduction’ approach to program design—encouraged in recent years by the structured programming crusade—may have a deleterious effect on our students’ abilities to think in process structure terms, rather than in problem reduction terms. The most pernicious effect of such an overemphasis is a belief that an elegant and economical program structure is to be equated with an elegant and economical process structure. The contrast between MOVESTEEPELLE and HANOI would be a good counterexample to such a belief. I am not arguing here against structured programming, but against certain aspects of Structured Programming.

MOVETOWER is doubly recursive and cannot be made iterative by a trivial syntactic transformation. The nonrecursive HANOI is possible because the data structure being manipulated contains enough information to decide what to do at each stage, making the control structure—the stack—superfluous. One might draw a (rather weak) analogy with the use of threaded trees to eliminate recursion in tree traversing programs, where the data structure is modified to include just the information needed by the recursive algorithm. We might say that HANOI is an optimisation of MOVETOWER, in the same sense that an iterative factorial program is an optimisation of the recursive algorithm, or the iterative ‘backstitching’ list reversing algorithm is an optimisation of the obvious recursive list reverser. But, I suggest, it would be a very nontrivial exercise to convert MOVETOWER to HANOI, let alone convert it mechanically. In fact, to close, I hereby offer it as a challenge to optimistic optimisers, and to those who make it their business to prove that equivalent programs are equivalent.

References
in large measure upon its 'track record', its availability, and its ease of use. The Powell-Macdonald method in FORTRAN code has been (and is) available to anyone who wishes to use it since 1972. In some respects, there are advantages to the Britt-Luecke method (1973), but we have been pleased with the wide variety of successful employment of our algorithm by a large number of users. Despite Southwell's claims of superiority in method to both Powell-Macdonald and Britt-Luecke, a potential user would be well advised to compare all available methods carefully before deciding which would be 'best' (in some sense) for his own problems.

Yours faithfully,

D. R. Powell* and J. Ross Macdonald‡

References


Dr. Southwell replies:

No attack was intended; however, I felt, for example, the number of iterations comparison (Table 4, Powell and Macdonald, 1972), which lists, 'Southwell . . . did not converge,' was rather harsh in view of the fact (Southwell, 1969) that the method converged as fast or faster than any other method listed.

My reply will be to each of the numbered comments above.

1. The basic methods of the 1969 paper and of the more recent paper (Southwell, 1976) are the same. The fundamental approach is to minimize the sum-of-the-squares function with respect to the a parameters, with xi being regarded as a function a=(x; through the relationship \( \delta y/\delta x_i = 0 \). The 1969 paper suggests a Newton-Raphson approach (a well-known and 'highly successful' technique when starting points are near the solution) for the multi-parameter vector a. For any fixed a, a subproblem is solved for x, which involves finding a solution to a single variable transcendental equation. The 1969 paper suggests solving for the x_i using the iterative scheme \( x_i^{\text{new}} = F(x) \) (Eq. 3, 1969). This is also a well-known technique and is easy to implement, since it requires no additional derivatives. The problem is that it doesn't always converge, depending on the nature of the function F(x). Thus, in solving the transcendental equation for x_i, the 1976 paper states that:

(a) Newton's method (also well-known) is preferred.

(b) Convergence should be achieved rather than relying on a single step adjustment.

The 1976 paper also points out that the basic method may be implemented using other nonlinear least squares techniques instead of the Newton-Raphson.

2. The error in the 'exact' parameter variance derivation in the 1969 paper was first revealed to the author by J. R. Macdonald, however, it was in an unsigned referee's report.

3. The 1976 paper was submitted in March 1974.

4. The 1976 paper states that tests are given for the x_i.

5. In view of the fact that there are several ways to implement a method, what I said (p. 71) was, 'It should be pointed out that with one difference Powell and Macdonald's method is equivalent to the Newton-Raphson, numerical derivative application of the method.' I don't think that is misleading. The point raised seems to be, which was the earlier method?

6. The 1976 paper gives examples of convergence of nonlinear models using the improved implementation (see item 1 in the reply) of the 1969 method. The belabored point appears to be whether or not the modifications (a) and (b) above constitute a change in method. Perhaps the answer depends on the reader. One looking for a package with which to fit his data may not see that it is a change. A numerical-analyst, however, would certainly not solve a transcendental equation using \( x_i^{\text{new}} = F(x) \) without checking its convergence. If it failed, he would use Newton's another method and still be able to achieve convergence using the 1969 paper.

The numerical derivative approach (described on p. 70 in the 1976 paper) does not require evaluation of third order mixed partial derivatives.

7. Table 4 (p. 152, Powell and Macdonald, 1972) states the convergence was poor using exact analytical derivatives. Apparently, they only used partial derivatives as exact derivatives would have given different results.

8. The analytical derivatives on that example are trivial. But, generally, the choice depends on the application. For a dedicated program to fit a specific function, the extra effort may be worthwhile in terms of accuracy and computer running time. For an easy-to-use general-purpose program, the numerical derivatives would probably be preferred. Fortunately, the theory is well understood so that either approach may be used.

9. They indeed did say that (p. 148 and p. 149). This would be minor exception that the method \( F a = a \) is not a trivial case and has a long history in the literature. Because of this, it was decided to illustrate the basic approach in the 1969 paper.

10. I took the above IBM 370 double precision results and used them as input parameters in CURFIT on a Hewlett-Packard 9830A calculator. CURFIT is a commercially available (Cyanetics Service Co., 3508 Fifth Avenue, Pittsburgh, Pennsylvania 15213, USA) interactive curvefitting program that also handles data uncertainties in both x and y. The special value on input was 0.4503256739. After one iteration, the results were:

\[
\begin{align*}
   a_1 &= -5.91482 \\
   a_2 &= -0.603146 \\
   a_3 &= -0.0803391 \\
   a_4 &= 0.0263827 \\
   a_5 &= -8.2874 	imes 10^{-4} \\
   a_6 &= -1.67448 	imes 10^{-4}
\end{align*}
\]

with \( a \simeq 0.4503256674 \). These results are in close agreement with my example 3 (1976) and indicate a better least squares solution.