```
        end;
        $i := i - m \div 2 - 1$;
        setmin := if $i < 0$ then 0 else if
        $i + m > n$ then $n - m$ else $i$
    end setmin;
    if $m > n$ then $m := n$; $j :=$ setmin (out);
    for $i := 0$ step 1 until $m$ do
    begin
        $z[i] := arg - x[j]$; $f[i] := y[j]$; $j := j + 1$
    end;
    mless 1 := $m - 1$;
    for $i := 0$ step 1 until mless 1 do
    begin
        $fi := f[i]$; $zi := z[i]$;
        for $j := i + 1$ step 1 until $m$ do
        $f[j] := fi + zi \times (f[j] - fi)/(zi - z[j])$
    end;
out: aitken := $f[m]$
end aitken
```

**Algorithm 11.**

**real procedure** *equipol* (*xbase, y, arg, n, m, h*); **value** *xbase, arg, m, n, h*; **real** *xbase, arg, h*; **array** *y*; **integer** *m, n*;
**comment** *array y*[0: *n*] *contains sample values of a function at corresponding equal interval values of the argument xbase, xbase* + *h, xbase* + 2 × *h*, . . ., *xbase* + *n* × *h. The procedure yields an approximation to the function at the specified value arg by evaluating an mth order polynomial* ($m \leqslant n$), *using Aitken's iterative method. The subset of* $m + 1$ *points used in the evaluation are suitably chosen to be evenly distributed about the value of arg. If the requested value of m exceeds n the assignment* $m := n$ *occurs.*

*For* $arg < x[0]$ *the procedure extrapolates using* $x[0], x[1], \ldots, x[m]$
*For* $arg > x[n]$ *the procedure extrapolates using* $x[n - m], x[n - m + 1], \ldots, x[n]$;

```
begin integer i, j, mless 1; real jh, fi; array f[0: m];
    if $m > n$ then $m := n$;
    $i :=$ entier $((arg - xbase)/h) - m \div 2$;
    $j :=$ if $i < 0$ then 0 else if $i + m > n$ then $n - m$
    else $i$;
    for $i := 0$ step 1 until $m$ do $f[i] := y[i + j]$;
    $arg := arg - j \times h - xbase$;
    mless 1 := $m - 1$;
    for $i := 0$ step 1 until mless 1 do
    begin $fi := f[i]$; $jh := h$;
        for $j := i + 1$ step 1 until $m$ do
        begin
            $f[j] := fi + arg \times (f[j] - fi)/jh$;
            $jh := jh + h$
        end;
        $arg := arg - h$
    end;
    equipol := $f[m]$
end equipol
```

## The Computer Journal

Published Quarterly by

The British Computer Society, 23 Dorset Square, LONDON, N.W.1, England.

*The Computer Journal* is registered at Stationers' Hall, London (certificate No. 20825, May 1958). The contents may not be reproduced, either wholly or in part, without permission.

Subscription price per volume £4 10s. (U.S. $12.60). Single Copies 25s. (U.S. $3.50)

All inquiries should be sent to the Secretary at the above address.

*Communications:* Papers submitted for publication should be sent to E. N. Mutch, The University Mathematical Laboratory, Corn Exchange Street, Cambridge. Intending authors should first apply for *Notes on the Submission of Papers*, as the onus of preparing papers in a form suitable for sending to press lies in the first place with the authors.

**Correspondence** (*continued from p.* 344)

2. *LL$^t$ or secant method?*

Although they do not describe it in these terms the authors' algorithm is essentially as follows. Begin with Sturm Sequence binary chop. At some stage (another *ad hoc* decision) switch to the secant method. Finally when the approximation (origin shift) is close enough to the lowest eigenvalue then the LL$^t$ transformation will deflate the matrix. The computation then proceeds to the next eigenvalue.

What is novel (and unfortunate) is that for much of the calculation the authors are doing the whole LL$^t$ transformation when they only use the value of the characteristic polynomial. This is somewhat wasteful. During the binary chop and secant phases little use is being made of the transformed matrices except when secondary factorization occurs.

The first point then is that their algorithm is essentially the secant method with deflation. Deflation by LL$^t$ turned out to be faster than deflation by QR. Not too surprising. To my mind a true LL$^t$ or QR technique makes use of the transformed matrix at each stage to determine where the next origin shift will be.

The second point is that the secant method has been studied for a long time (Ostrowski, *Solutions of Equations and Systems of Equations*, Academic Press, N.Y., or Ralston, *A First Course in Num. Anal.*, Chap. 8, McGraw-Hill). It can be made into a viable algorithm but only with much care and analysis; without this it has grave defects near root clusters. The convergence rate drops sharply and premature underflow can halt it altogether. At this juncture the authors' algorithm becomes LL$^t$ without shifts; linear convergence and a ratio close to unity.

On a nearly diagonal 30 × 30 matrix of the form

$$\begin{bmatrix} 1 & 10^{-4} & & & \\ 10^{-4} & -1 & 10^{-4} & & \\ & 10^{-4} & 1 & 10^{-4} & \\ & & 10^{-4} & -1 & 10^{-4} \\ & & & \cdot & \cdot & \cdot \end{bmatrix}$$

the following results occur on a B5500 computer (11 decimals).

| | Time |
|---|---|
| QR (P.A. Businger, Algorithm 253, *Comm. ACM*, Vol. 8 (1965), pp. 217–218) | $T_{Bu} = 1 \cdot 2$ sec. |
| Sturm Sequence (J. H. Wilkinson, *Numer. Math.*, Vol. 4, No. 4 (1962), p. 362) | $T_{Wi} = 13$ sec. |
| Fox–Johnson ($\epsilon = 10^{-22}$) *was stopped after* | 4 *min.* |

All these programs were set to give answers of maximal accuracy. However, if we take the value of $\epsilon$ given by Fox and Johnson then their program takes only 0·13 seconds to get answers with errors up to $10^3$ in the last place and no warning that this has occurred. In fact two eigenvalues are given multiplicity 14. Of course the other programs obtain answers of this accuracy in comparable times.

The first two algorithms were chosen because they have been published. It is not claimed that they are preferred realizations of their methods, and in fact better implementations do exist. Note that if the above matrix is multiplied by $10^4$ then Wilkinson's Sturm Sequence algorithm (being straight ALGOL) results in overflow on many computers.

My main point is that much better (and more compact) algorithms than the authors' are already in the literature. However, I would like to draw attention to the careful QR algorithm recently submitted to the *Communications of the ACM* by Kahan and Varah. It took 0·8 seconds on the above matrix to give maximal accuracy.

On a variety of other matrices the times taken by the above programs were related by $T_{Wi} > 2T_{FJ} > 4T_{Bu}$.

Further comments could be made on

2. Avoiding overflow/underflow in Sturm Sequence algorithms,
4. Implementing the QR method,

and

5. Brevity of program.

Yours faithfully,
BERESFORD PARLETT

P.S. I am grateful to Jim Varah for expert programming.

Mathematics Department,
University of California,
Berkeley.
13 September 1966.

## Correction

## Editorship of The Computer Journal

We regret that owing to a printer's error on p. xx of the May 1966 issue of this *Journal*, the name of the honorary editor was given as Mr. M. Bridger of Leicester. Mr. Bridger was at that time honorary editor of our associated publication *The Computer Bulletin*. We regret the inconvenience that has been caused by this mistake, and we wish to confirm that contributions to this *Journal*, other than to the Algorithms Supplement, should continue to be addressed to

**Mr. E. N. Mutch,**
**University Mathematical Laboratory,**
**Corn Exchange Street,**
**CAMBRIDGE.**