The Reduction of a Matrix to Codiagonal Form by Eliminations

By C. Strachey and J. G. F. Francis

This paper describes a method of reducing a general matrix to codiagonal form by using a sequence of elementary transformations, and identifies the second stage of the reduction with the Lanczos transformation. Some practical problems of computation are discussed.

Introduction

It is possible to derive several interesting similarity transformations of a matrix by means of series of elementary transformations. The whole operation may be very complicated, but each step is relatively simple, and when considered in this way we gain a new insight into the computing process and can often see more easily where numerical errors creep in.

We describe in this paper the use of elementary transformations to reduce a matrix first to almost triangular form and then from this to codiagonal form. We prove that the second stage of the reduction is exactly equivalent to the method of Lanczos (1950) and briefly discuss the relationship that the codiagonal form bears to the almost triangular matrix and to the original matrix. Various points concerning errors and accuracy, and some details of a practical program are then mentioned.

1. Notation

Capital Roman letters represent $n \times n$ matrices, bold-face lower-case Roman letters represent column vectors, and small Greek letters represent scalars. A prime indicates a transpose, so that $\boldsymbol{u}'$ is a row vector. We shall generally consider matrices to be made up of sets of column vectors, e.g. $A = (\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n)$; when individual elements of a matrix are required they will be in light-face lower-case Roman letters with a double suffix (occasionally omitted).

We shall use A for a general square matrix, B for a lower almost triangular matrix (i.e. with $b_{ij} = 0$ for $j > i + 1$) and C for a codiagonal matrix (i.e. with $c_{ij} = 0$ for $|i - j| > 1$). We shall also use $\boldsymbol{e}_r$ for the rth column of the unit matrix, so that $I = (\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n)$.

2. Elimination Transformations

We define an elimination transformation of the matrix A to be an elementary similarity transformation TAT^{-1}, where the matrix T (which consists of the unit matrix with one additional off-diagonal element) is chosen so that one particular element of TAT^{-1} is zero. For example

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & \mu & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -\mu & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & a_{15} \\ \underline{a_{21}} & \underline{a_{22}} & \underline{a_{23}} & \underline{a_{24}} & \underline{a_{25}} \\ a_{31} & a_{32} & a_{33} & \underline{a_{34}} & a_{35} \\ a_{41} & a_{42} & a_{43} & \underline{a_{44}} & a_{45} \\ a_{51} & a_{52} & a_{53} & \underline{a_{54}} & a_{55} \end{bmatrix}$$

where $\mu = \dfrac{a_{14}}{a_{12}}$ (a_{12} must be non-zero). Only the underlined quantities have been altered by the transformation.

This transformation will be referred to as "eliminating a_{14} with a_{12}" or "using a_{12} to eliminate a_{14}." It is to be thought of as first subtracting μ times the second column from the fourth and then adding μ times the fourth row to the second. These two operations (the column and row operations respectively) can, of course, be performed in the reverse order if required.

So long as the element which is used for eliminating (a_{12} in the example) is not on the principal diagonal, the multiple involved in the row and column operations is merely the ratio of two elements of the matrix, and is therefore quite simply found.

3. Reduction to Almost Triangular Form

The element immediately to the right of the principal diagonal is used to eliminate all the remaining elements in the same row to its right. The process starts with the top row. A typical stage is:

$$\begin{bmatrix} a & a & 0 & 0 & 0 & 0 & 0 & 0 \\ a & a & a & 0 & 0 & 0 & 0 & 0 \\ a & a & a & x & y & y & y & y \\ a & a & a & a & a & a & a & a \\ a & a & a & a & a & a & a & a \\ a & a & a & a & a & a & a & a \\ a & a & a & a & a & a & a & a \\ a & a & a & a & a & a & a & a \end{bmatrix} \succ \begin{bmatrix} a & a & 0 & 0 & 0 & 0 & 0 & 0 \\ a & a & a & 0 & 0 & 0 & 0 & 0 \\ a & a & a & x & 0 & 0 & 0 & 0 \\ \underline{a} & \underline{a} & \underline{a} & \underline{a} & \underline{a} & \underline{a} & \underline{a} & \underline{a} \\ a & a & a & a & \underline{a} & \underline{a} & \underline{a} & \underline{a} \\ a & a & a & a & \underline{a} & \underline{a} & \underline{a} & \underline{a} \\ a & a & a & a & \underline{a} & \underline{a} & \underline{a} & \underline{a} \\ a & a & a & a & \underline{a} & \underline{a} & \underline{a} & \underline{a} \end{bmatrix}.$$

The element x is used to eliminate the elements y. The corresponding row operations all alter the row below.

In this case we can also allow a permutation (i.e. the interchange of two rows and the corresponding columns) before the eliminations so that x is not smaller (in modulus) than any of the y's. This means that we can guarantee that none of the μ's used in the elimination will be greater than one. The whole operation can therefore be carried out in fixed-point arithmetic.

The final result is a matrix in the form of a lower triangle with one extra element in each row to the right of the principal diagonal. This "almost triangular" form is the starting point for several other transformations and is of interest on its own account. See Wilkinson (1959) for a full description of the reduction to this form.

4. The Codiagonal Form derived from the Almost Triangular Matrix

The elements immediately below the principal diagonal of the almost triangular form are used to eliminate all the other subdiagonal elements thus leaving the matrix in a codiagonal form.

We eliminate the elements column by column starting with $\boldsymbol{b}_1$, the first column. Thus we use b_{21} to eliminate the elements $b_{31}, b_{41}, \ldots, b_{n1}$. The elimination transformations multiply together to give the equation (shown in Table I on p. 170) where the numbers μ_r are the ratios b_{r1}/b_{21}.

In eliminating these elements the only other elements in the matrix that are changed are those (underlined above) in the second and third columns. We now eliminate the second column with the element $\underline{b}_{32}$ (which involves altering the elements in the third and fourth columns), and this process is repeated until the codiagonal form is reached.

5. Connection with the Lanczos Method

In this section we shall prove that the elimination method described above for reducing the almost triangular matrix to codiagonal form is in fact exactly equivalent to a method suggested by Lanczos (1950). We shall show that the columns of ratios which are used in the elimination process are identical with the right-hand set of vectors defined by Lanczos.

We shall use induction to prove a recurrence relation between these vectors, and later identify them with the Lanczos vectors.

Let us consider the elimination process when the first $(r-2)$ columns have been dealt with. The matrix will take the form

$$
\boldsymbol{M}_r = \left[\begin{array}{ccccccccccccc}
\alpha_1 & \gamma_2 & & & & & & & & & & & \\
\beta_2 & \alpha_2 & \gamma_3 & & & & & & & & & & \\
 & \beta_3 & \cdot & \cdot & & & & & & & & & \\
 & & \cdot & \cdot & \cdot & & & & & & & & \\
 & & & \cdot & \cdot & \gamma_{r-2} & & & & & & & \\
 & & & & \beta_{r-2} & \alpha_{r-2} & \gamma_{r-1} & & & & & & \\
 & & & & & \beta_{r-1} & \alpha_{r-1} & \gamma_r & & & & & \\
 & & & & & & \beta_r & y & b & & & & \\
 & & & & & & x & y & b & b & & & \\
 & & & & & & \cdot & \cdot & \cdot & \cdot & \cdot & & \\
 & & & & & & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\
 & & & & & & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
 & & & & & & x & y & b & b & \cdot & \cdot & \cdot & b \\
 & & & & & & x & y & b & b & \cdot & \cdot & \cdot & b
\end{array}\right] .
$$

The elements b in columns $(r+1)$ to n are unaltered from the almost triangular matrix $\boldsymbol{B}$; the first $(r-2)$ columns are in their final codiagonal form.

The next stage in the elimination process is to use the element β_r to eliminate all the elements x in the column below it. To do this, we form the ratios x/β_r. Let us therefore define a vector $\bar{\boldsymbol{v}}_r$ whose elements are these ratios, with zeros in the remaining positions (i.e. rows 1 to r). Thus for the $(r-1)$th column of $\boldsymbol{M}_r$ we have

$$
\boldsymbol{x}_{r-1} = \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ \gamma_{r-1} \\ \alpha_{r-1} \\ \beta_r \\ x \\ x \\ \cdot \\ \cdot \\ \cdot \\ x \end{bmatrix} = \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ \gamma_{r-1} \\ \alpha_{r-1} \\ \beta_r \\ 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ 0 \\ 0 \\ x \\ x \\ \cdot \\ \cdot \\ \cdot \\ x \end{bmatrix} = \boldsymbol{c}_{r-1} + \beta_r \bar{\boldsymbol{v}}_r
$$

Note that the vectors $\bar{\boldsymbol{v}}_r$ are defined successively by this equation at each stage of the process.

For the purposes of the induction we assume that the rth column of $\boldsymbol{M}_r$ takes the form $\boldsymbol{y}_r = \boldsymbol{b}_r - \gamma_r \bar{\boldsymbol{v}}_{r-1}$. Thus we have

$$\boldsymbol{M}_r = (\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_{r-2}, \boldsymbol{x}_{r-1}, \boldsymbol{y}_r, \boldsymbol{b}_{r+1}, \boldsymbol{b}_{r+2}, \ldots, \boldsymbol{b}_n)$$

where
$$\boldsymbol{x}_{r-1} = \boldsymbol{c}_{r-1} + \beta_r \bar{\boldsymbol{v}}_r$$
$$\boldsymbol{y}_r = \boldsymbol{b}_r - \gamma_r \bar{\boldsymbol{v}}_{r-1} .$$

The next stage of elimination is obtained using the transformation $\boldsymbol{M}_{r+1} = \boldsymbol{T}_r \boldsymbol{M}_r \boldsymbol{T}_r^{-1}$
where

$$\boldsymbol{T}_r = (\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_{r-1}, [\boldsymbol{e}_r - \bar{\boldsymbol{v}}_r], \boldsymbol{e}_{r+1}, \ldots, \boldsymbol{e}_n)$$

so that

$$\boldsymbol{T}_r^{-1} = (\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_{r-1}, [\boldsymbol{e}_r + \bar{\boldsymbol{v}}_r], \boldsymbol{e}_{r+1}, \ldots, \boldsymbol{e}_n)$$
$$= (\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_{r-1}, \boldsymbol{v}_r, \boldsymbol{e}_{r+1}, \ldots, \boldsymbol{e}_n) .$$

Table I

$$\begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & -\mu_3 & 1 & & & & \\ & -\mu_4 & & 1 & & & \\ & \cdot & & & \cdot & & \\ & \cdot & & & & \cdot & \\ & -\mu_n & & & & & 1 \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & & & & \\ b_{21} & b_{22} & b_{23} & & & \\ b_{31} & b_{32} & b_{33} & b_{34} & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \\ \cdot & \cdot & \cdot & \cdot & \cdot & b_{n-1\,n} \\ b_{n1} & \cdot & \cdot & \cdot & \cdot & b_{nn} \end{bmatrix} \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & \mu_3 & 1 & & & & \\ & \mu_4 & & 1 & & & \\ & \cdot & & & \cdot & & \\ & \cdot & & & & \cdot & \\ & \mu_n & & & & & 1 \end{bmatrix}$$

$$= \begin{bmatrix} b_{11} & b_{12} & & & & \\ b_{21} & \underline{b}_{22} & \underline{b}_{23} & & & \\ 0 & \underline{b}_{32} & \underline{b}_{33} & b_{34} & & \\ 0 & \underline{b}_{42} & \underline{b}_{43} & \cdot & \cdot & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \\ \cdot & \cdot & \cdot & \cdot & \cdot & b_{n-1\,n} \\ 0 & \underline{b}_{n2} & \underline{b}_{n3} & b_{n4} & \cdot & b_{nn} \end{bmatrix}$$

Note that $\boldsymbol{v}_r = \boldsymbol{e}_r + \boldsymbol{v}_r$ is a vector with unity in the rth row and zeros above it.

Then $\boldsymbol{M}_r\boldsymbol{T}_r^{-1}$ only differs from $\boldsymbol{M}_r$ in the rth column which becomes

$$\boldsymbol{M}_r\boldsymbol{v}_r = \boldsymbol{y}_r + \boldsymbol{M}_r\bar{\boldsymbol{v}}_r \ .$$

As $\bar{\boldsymbol{v}}_r$ is zero above the $(r + 1)$th row, and columns $(r + 1)$ to n of $\boldsymbol{M}_r$ are the same as those of $\boldsymbol{B}$, this becomes

$$\begin{aligned} \boldsymbol{M}_r\boldsymbol{v}_r &= \boldsymbol{y}_r + \boldsymbol{M}_r\bar{\boldsymbol{v}}_r \\ &= \boldsymbol{b}_r - \gamma_r\bar{\boldsymbol{v}}_{r-1} + \boldsymbol{B}\bar{\boldsymbol{v}}_r \\ &= \boldsymbol{B}\boldsymbol{v}_r - \gamma_r\bar{\boldsymbol{v}}_{r-1} \ . \end{aligned}$$

It can readily be seen that the first two non-zero elements in this vector are the $(r - 1)$th and the rth. The first of these is γ_r; we will call the second α_r. (This is the definition of α_r.)

We define γ_{r+1} as the top non-zero element of $\boldsymbol{b}_{r+1}$ (i.e. $\gamma_{r+1} = b_{r\,r+1}$) so that we get as the rth row of $\boldsymbol{M}_r\boldsymbol{T}_r^{-1}$

$$[0, 0, \ldots, \beta_r, \alpha_r, \gamma_{r+1}, 0, \ldots, 0].$$

The effect of premultiplying by $\boldsymbol{T}_r$ is to subtract multiples of this row from the rows below it so as to reduce the rest of column $(r - 1)$ to zero. This will alter columns r and $(r + 1)$ by subtracting from them $\alpha_r\bar{\boldsymbol{v}}_r$ and $\gamma_{r+1}\bar{\boldsymbol{v}}_r$ respectively.

Thus column $(r - 1)$ becomes $\boldsymbol{c}_{r-1}$, column r becomes $\boldsymbol{x}_r = \boldsymbol{B}\boldsymbol{v}_r - \gamma_r\bar{\boldsymbol{v}}_{r-1} - \alpha_r\bar{\boldsymbol{v}}_r$ and column $(r + 1)$ becomes $\boldsymbol{y}_{r+1} = \boldsymbol{b}_{r+1} - \gamma_{r+1}\bar{\boldsymbol{v}}_r$. Note that the form of $\boldsymbol{y}_{r+1}$ is similar to that of $\boldsymbol{y}_r$.

We define β_{r+1} as being the element in the $(r + 1)$th row of $\boldsymbol{x}_r$. The two elements above this have not been altered by premultiplying by $\boldsymbol{T}_r$, so that the first three non-zero elements of $\boldsymbol{x}_r$ are γ_r, α_r and β_{r+1} which are the non-zero elements of $\boldsymbol{c}_r$.

Thus we can put $\boldsymbol{x}_r = \boldsymbol{c}_r + \beta_{r+1}\bar{\boldsymbol{v}}_{r+1}$, where $\bar{\boldsymbol{v}}_{r+1}$ will be zero down to (and including) row $(r + 1)$. This is the form we assumed for $\boldsymbol{x}_{r-1}$ initially.

To complete the induction we must verify that the first step for $r = 2$ is correct. If we assume $\bar{\boldsymbol{v}}_1 = 0$ (so that $\boldsymbol{v}_1 = \boldsymbol{e}_1$), $\alpha_1 = b_{11}$ and $\beta_2 = b_{21}$ we can put

$$\beta_2\bar{\boldsymbol{v}}_2 = \boldsymbol{b}_1 - \boldsymbol{c}_1 = \begin{bmatrix} 0 \\ 0 \\ b_{31} \\ b_{41} \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ b_{n1} \end{bmatrix} .$$

Then we have $\boldsymbol{M}_2 = \boldsymbol{B}$ so that the correct conditions for starting exist with $\boldsymbol{x}_1 = \boldsymbol{c}_1 + \beta_2\bar{\boldsymbol{v}}_2 = \boldsymbol{b}_1$ and $\boldsymbol{y}_2 = \boldsymbol{b}_2 - \gamma_2\bar{\boldsymbol{v}}_1 = \mathbf{b}_2$ which are the first two columns of $\boldsymbol{B}$.

We can now form a recurrence relation connecting the $\boldsymbol{v}_r$. We have

$$\boldsymbol{x}_r = \boldsymbol{c}_r + \beta_{r+1}\bar{\boldsymbol{v}}_{r+1} = \boldsymbol{B}\boldsymbol{v}_r - \gamma_r\bar{\boldsymbol{v}}_{r-1} - \alpha_r\bar{\boldsymbol{v}}_r.$$

The vector $\boldsymbol{c}_r$ has elements only in rows $(r - 1)$, r and $(r + 1)$.

If we put these elements at the top of the vectors $\gamma_r\bar{\boldsymbol{v}}_{r-1}$, $\alpha_r\bar{\boldsymbol{v}}_r$ and $\beta_{r+1}\bar{\boldsymbol{v}}_{r+1}$ respectively we get

$$\beta_{r+1}\boldsymbol{v}_{r+1} = \boldsymbol{B}\boldsymbol{v}_r - \alpha_r\boldsymbol{v}_r - \gamma_r\boldsymbol{v}_{r-1}. \qquad (1)$$

This is exactly the recurrence relation used by Lanczos in his method of "minimized iterations." However, at first sight the method used by Lanczos for determining the constants (α, β and γ) involved appears to be quite different. Lanczos uses a second set of row vectors $\boldsymbol{u}_1'$, $\boldsymbol{u}_2'$, . . ., $\boldsymbol{u}_n'$ which satisfy a similar recurrence relation (with the same α, β, γ) and which form a biorthogonal set with $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$. The constants α_r and γ_r are determined by the requirement that $\boldsymbol{v}_{r+1}$ shall be orthogonal to $\boldsymbol{u}'_r$ and $\boldsymbol{u}'_{r-1}$ (which ensures that it is also orthogonal to the earlier $\boldsymbol{u}$'s), β_{r+1} is a scale factor chosen to make the scalar product $\boldsymbol{u}'_{r+1}\boldsymbol{v}_{r+1}$ equal to unity. If the matrix $\boldsymbol{V}$ is made up from the column vectors $\boldsymbol{v}$ (so that $\boldsymbol{V} = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n)$) and $\boldsymbol{U}'$ is similarly made up from the row vectors $\boldsymbol{u}'$, the recurrence relation gives the matrix equation $\boldsymbol{BV} = \boldsymbol{VC}$ while the biorthogonality condition (together with the scaling) gives $\boldsymbol{U}'\boldsymbol{V} = \boldsymbol{I}$ so that $\boldsymbol{U}'\boldsymbol{BV} = \boldsymbol{C}$ is in fact a similarity transformation which produces the codiagonal form $\boldsymbol{C}$.

The algorithm given by Lanczos shows that once the initial vectors $\boldsymbol{u}'_1$ and $\boldsymbol{v}_1$ have been chosen, all the remaining vectors are in general uniquely determined. (The exceptions are relatively unimportant.) Thus the transformation matrices $\boldsymbol{U}'$ and $\boldsymbol{V}$, and hence also the codiagonal form $\boldsymbol{C}$, are fixed by the choice of the initial vectors. We shall now show that the elimination method described above is equivalent to choosing the initial vectors $\boldsymbol{u}_1 = \boldsymbol{v}_1 = \boldsymbol{e}_1$.

If we choose $\boldsymbol{u}'_1 = \boldsymbol{e}'_1 = (1, 0, \ldots, 0)$ and take account of the nearly triangular form of $\boldsymbol{B}$ and the recurrence relation satisfied by the $\boldsymbol{u}_r$ (viz. $\gamma_{r+1}\boldsymbol{u}'_{r+1} = \boldsymbol{u}'_r\boldsymbol{B} - \alpha_r\boldsymbol{u}'_r - \beta_r\boldsymbol{u}'_{r-1}$) it is easy to see that each successive $\boldsymbol{u}_r$ has one more non-zero element than the last, so that the matrix $\boldsymbol{U}'$ whose rows are $\boldsymbol{u}'_1, \boldsymbol{u}'_2, \ldots, \boldsymbol{u}'_n$ is a strictly lower triangular matrix. The fact that $\gamma_{r+1} = b_{r\,r+1}$ ensures that the elements on the principal diagonal of $\boldsymbol{U}'$ are all unity. Thus $\boldsymbol{U}'$ is non-singular and $\boldsymbol{V} = \boldsymbol{U}'^{-1}$ is also a lower triangular matrix with a unit diagonal. Thus the orthogonality condition on the vectors $\boldsymbol{v}_r$ (i.e. $\boldsymbol{u}'_{r-1}\boldsymbol{v}_r = \boldsymbol{u}_{r-2}\boldsymbol{v}_r = 0$) is simply that they must form the triangular matrix $\boldsymbol{V}$ of the correct shape. It is an inherent property of the method of minimized iterations that the orthogonality conditions $\boldsymbol{u}'_{r-1}\boldsymbol{v}'_r = \boldsymbol{u}'_{r-2}\boldsymbol{v}_r = 0$ imply that $\boldsymbol{u}'_s\boldsymbol{v}_r = 0$ for all $s < r$, and also that $\boldsymbol{u}'_r\boldsymbol{v}_s = 0$ for all $s < r$. Thus the fact that the matrices $\boldsymbol{U}'$ and $\boldsymbol{V}$ are strictly lower triangular, combined with the recurrence relations, is sufficient to show that $\boldsymbol{U}'\boldsymbol{V} = \boldsymbol{I}$.

Returning now to the elimination method, the complete reduction will be

$$\boldsymbol{TBT}^{-1} = \boldsymbol{T}_n\boldsymbol{T}_{n-1} \ldots \boldsymbol{T}_2\boldsymbol{B}\boldsymbol{T}_2^{-1}\boldsymbol{T}_3^{-1} \ldots \boldsymbol{T}_n^{-1}$$

and it is easy to see that

$$\boldsymbol{T}^{-1} = \boldsymbol{T}_2^{-1}\boldsymbol{T}_3^{-1} \ldots \boldsymbol{T}_n^{-1} = (\boldsymbol{e}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \ldots, \boldsymbol{v}_n) = \boldsymbol{V}, \quad \text{as } \boldsymbol{v}_1 = \boldsymbol{e}_1$$

and furthermore, that $\boldsymbol{V}$ will be a strictly lower triangular matrix. Thus in equation (1) which is the recurrence relation both for the elimination method and for Lanczos' algorithm, the constants are determined to ensure that the vectors are of the right form, and this automatically ensures that they are orthogonal to the vectors $\boldsymbol{u}'_r$, although these are never found explicitly.

This shows that the two methods are equivalent and, moreover, that all the numbers involved are the same. The right-hand Lanczos vector is exactly the column of ratios used in eliminating the elements below the subdiagonal of the rth column of the matrix.

6. Some Remarks on the Codiagonal Form

We can choose the starting vectors $\boldsymbol{u}_1$ and $\boldsymbol{v}_1$ for the Lanczos method at will, apart from the normalizing condition $\boldsymbol{u}'_1\boldsymbol{v}_1 = 1$, and different choices will in general give different codiagonal forms. A codiagonal form with a fixed characteristic equation has generally $n - 1$ degrees of freedom if we disregard transformation by a diagonal matrix (diagonal scaling does not change the elements on the main diagonal nor the products of corresponding off-diagonal elements, and these are the essential quantities). If we write $\Delta_1(\lambda) = |\boldsymbol{C} - \lambda\boldsymbol{I}|$ and $\Delta_2(\lambda)$ for the minor of the first element of this determinant, then the quantities α_r and $\beta_r\gamma_r$ are defined by the continued fraction expansion:

$$\frac{\Delta_1(\lambda)}{\Delta_2(\lambda)} = \alpha_1 - \lambda - \cfrac{\beta_2\gamma_2}{\alpha_2 - \lambda - \cfrac{\beta_3\gamma_3}{\alpha_3 - \lambda - \ldots}}.$$

Thus the $n - 1$ degrees of freedom in $\boldsymbol{C}$ correspond to the $n - 1$ choices we have (in theory) for the coefficients of the polynomial $\Delta_2(\lambda)$.

The Lanczos starting vectors have $2n - 1$ degrees of freedom. In general, of these, $n - 1$ determine the resulting form $\boldsymbol{C}$, and n arise from the fact that $\boldsymbol{u}'_1$ and $\boldsymbol{v}_1$ can be replaced without effecting $\boldsymbol{C}$ by $\boldsymbol{u}'_1\boldsymbol{P}^{-1}$ and $\boldsymbol{P}\boldsymbol{v}_1$, where $\boldsymbol{P} = \boldsymbol{P}(\boldsymbol{B})$ is any polynomial in $\boldsymbol{B}$. Thus $\boldsymbol{U}'\boldsymbol{P}^{-1}\boldsymbol{BPV} = \boldsymbol{U}'\boldsymbol{BV} = \boldsymbol{C}$ because $\boldsymbol{B}$ commutes with $\boldsymbol{P}$. In general an arbitrary polynomial in a matrix of order n has n degrees of freedom.

In the elimination method $\boldsymbol{u}_1$, which is implicit, is necessarily equal to $\boldsymbol{e}_1$, but we can choose $\boldsymbol{v}_1$ by a slight modification of the process. If $\boldsymbol{v}_1 = \boldsymbol{e}_1 + \bar{\boldsymbol{v}}_1$ the first element of $\bar{\boldsymbol{v}}_1$ is zero (as $\boldsymbol{u}'_1\boldsymbol{v}_1 = 1$), while the remaining elements are at choice giving the required $n - 1$ degrees of freedom corresponding to those of the codiagonal form. If we wish $\bar{\boldsymbol{v}}_1$ to be non-zero we initially pre- and post-multiply $\boldsymbol{B}$ by $\boldsymbol{T}_1$ and $\boldsymbol{T}_1^{-1}$ respectively,

where $\qquad \boldsymbol{T}_1 = (\boldsymbol{e}_1 - \bar{\boldsymbol{v}}_1, \boldsymbol{e}_2, \boldsymbol{e}_3, \ldots, \boldsymbol{e}_n)$

and $\qquad \boldsymbol{T}_1^{-1} = (\boldsymbol{e}_1 + \bar{\boldsymbol{v}}_1, \boldsymbol{e}_2, \boldsymbol{e}_3, \ldots, \boldsymbol{e}_n)$.

Thus we have

$$\boldsymbol{TBT}^{-1} = \boldsymbol{C} \text{ where } \boldsymbol{T} = \boldsymbol{T}_n\boldsymbol{T}_{n-1} \ldots \boldsymbol{T}_2\boldsymbol{T}_1$$

and so we see that

$$\boldsymbol{T}^{-1} = (\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \ldots, \boldsymbol{v}_n) = \boldsymbol{V}.$$

The transformation of $\boldsymbol{B}$ by $\boldsymbol{T}_1$ is exactly similar to the subsequent transformations or eliminations by the other $\boldsymbol{T}_r$. Multiples of the first row of $\boldsymbol{B}$ (i.e. of the elements b_{11} and b_{12}) are subtracted from the rows below, and the same multiples of corresponding columns are added into the first column. The multiples are the elements of $\bar{\boldsymbol{v}}_1$. One can show that in general the row operations can change the minor of the first element of $|\boldsymbol{B} - \lambda \boldsymbol{I}|$ arbitrarily and can thus change arbitrarily $\Delta_2(\lambda)$ defined above. (This minor is unchanged by the subsequent eliminations because the matrices $\boldsymbol{T}_2, \boldsymbol{T}_3, \ldots, \boldsymbol{T}_n$ have their first rows and columns equal to $\boldsymbol{e}_1$.) Thus one can show independently of any consideration of degrees of freedom that, in general, a suitable choice of multiples is sufficient to determine any codiagonal form $\boldsymbol{C}$ similar to $\boldsymbol{B}$.

When we consider the complete reduction to codiagonal form starting from the square matrix $\boldsymbol{A}$, as described in Sections 3 and 4, and not including a choice of $\boldsymbol{v}_1$ different from $\boldsymbol{e}_1$, it is interesting to note that all the transformation matrices concerned in the eliminations have first rows and columns equal to those of the unit matrix. Therefore, if $\boldsymbol{X}'\boldsymbol{A}\boldsymbol{Y} = \boldsymbol{C}$ is the complete transformation, we have $\boldsymbol{x}_1 = \boldsymbol{y}_1 = \boldsymbol{e}_1$. Thus we can easily see that the elimination method produces the same codiagonal form as the Lanczos method using matrix $\boldsymbol{A}$ and starting vectors equal to $\boldsymbol{e}_1$, and that the minor of the first element of $|\boldsymbol{A} - \lambda \boldsymbol{I}|$ is equal to the same minor of $|\boldsymbol{C} - \lambda \boldsymbol{I}|$ and determines the particular codiagonal form obtained. We must point out that the intermediate arithmetic involved in the Lanczos method in this case is quite different from that in the elimination method.

7. Accuracy

When considering the accuracy of the method it is worth while investigating where the errors can come in and how they can grow. In common with most direct methods the method described above for reducing a square matrix to a codiagonal form will give a precise answer provided the operations are carried out with infinite precision. It follows therefore that the original source of all errors lies in the finite precision of the arithmetic, in other words in the round-off. When they are first formed these round-off errors are quite small and, unless the overall error of the process is considerably larger than this, there is not usually much cause for concern. There are two ways in which the originally small round-off errors may grow and become objectionable. The first is by the accumulation of a large number of small errors by addition or subtraction. The second is by multiplication of a small error by a quantity appreciably larger than one.

In the first part of the process described, i.e. the reduction to nearly triangular form, we have seen that by suitable permutations it is possible to ensure that none of the multiples used in the process is greater than one. This means in turn that there is no possibility that any round-off error is multiplied by a large quantity, so that the only way in which errors can accumulate is by addition. Procedures of this type are numerically very well behaved; not only are they eminently suitable for fixed-point operation, but they also turn out to be very accurate. It is not always possible to give an exact error analysis and it might seem at first sight possible for the error to double at each operation. However, in practice the rate of build-up of error is very much lower than this. Wilkinson has carried out an exact error analysis for a process which is somewhat similar to the one described here. In the case of triangulation by Gaussian eliminations with interchanges he has shown that the result, although not the exact solution to the problem given, is the exact solution to another problem which differs from the original one in only a few units in the last place.

We feel confident therefore that the first part of this process is stable and extremely well-behaved, and that unless the elements of the original matrix are known exactly it is sufficient to do the arithmetic to single precision and fixed point. The nearly triangular matrix formed will be an exact similarity transformation of a matrix which differs from the original matrix by only a few units in the last place. There are various practical advantages to this. In the first place about 5/6ths of the total number of multiplications in the whole reduction are consumed in the first part of the process. If this can be done single precision and fixed point the saving in time is very considerable. Furthermore this part of the reduction is the only one in which a large number of elements of the matrix have to be stored. It is quite simple to arrange a program so that the original matrix is overwritten by the nearly triangular form, so that one matrix space will be sufficient for this part of the operation.

When we come to the second part of the process, the reduction of the nearly triangular form to a codiagonal form, the situation is not so satisfactory. We are not able to perform any interchanges before eliminating the elements of a column so that we are unable to guarantee the size of the multiples involved. Nor does there seem to be any reasonable method of choosing an initial set of multiples $\bar{\boldsymbol{v}}_1$ so that the multiples subsequently involved can be controlled in size even if such a set exists. This has two consequences: in the first place we are no longer able to be sure that the numbers will remain fractional, and in the second, we are faced with the possibility of errors growing catastrophically by multiplication by a large number. The cure for both these difficulties is to work to greater precision.

We have found it convenient to continue to do the arithmetic fixed point allowing an integral part and two or three lengths for the fractional part of each number. It will be seen that as there are only two columns of the matrix which are altered during a stage of elimination (the $\boldsymbol{x}$ and $\boldsymbol{y}$ of Section 5), there are at most two vectors of numbers which need to be kept to multiple precision. This means that the storage requirements are not prohibitive. Furthermore, as the total number of multiplications in this part of the process is only about $\frac{1}{6}n^3$, the time taken for the multiple precision operation is not very serious.

It is worth while investigating a little more closely where the numerical difficulties may arise. These can only occur if an element β_r immediately below the diagonal which is used for eliminating the column below it turns out to be small. Two cases have to be distinguished; in the first not only is the element β_r small but all the x's below are also small; in the second, although the β_r is small not all the x's below are small.

In the first case the multiples involved are not large, but are badly determined as they are the ratio of two small quantities. From our previous argument we would not expect that this situation would introduce a very large error, and this is in fact the case. This situation corresponds exactly with the case in the Lanczos transformation where the new vector turns out to be small and has to be multiplied by a considerable factor in order to normalize it. In this case it is necessary to reorthogonalize the vector against all the previous vectors, and provided this is done it is immaterial that the vector was itself to a certain extent ill-defined. In our case the reorthogonalization is automatic, owing to the form of the matrices involved, so that no error is introduced by the fact that the ratios are badly determined. In effect, the final result is a codiagonal form which is a perfectly good similarity transformation of the original matrix, but not one which is very well determined by the starting conditions.

The second case, where β_r is small and one of the x's in the column below it is not small, is more troublesome. In this case one of the multiplies will be large and there is a danger that some of the round-off errors will be multiplied by this factor. It remains true, however, that an increase in the precision of the arithmetic will avoid these difficulties, and it is for this reason that the second part of the program is arranged to work multiple length. It is perhaps worth repeating that since this process has been identified with the Lanczos method the same difficulty must make its appearance there. It has in fact been described by Wilkinson (1958, page 152) and occurs when the corresponding left and right vectors turn out to be nearly orthogonal. Fortunately, this situation does not seem to occur very frequently and in practice has caused very little difficulty.

8. Practical Results

A program based on the method described above has been in operation on Pegasus since Spring 1959 and will accept matrices of order up to 48. The first part of the program is single length, fixed point. The second part of the program is basically five-length fixed point, but there is a provision for placing the binary point between any two sections of the number and arrangements have been made that zero integral sections shall not cause any unnecessary work to be done. If the number of fractional sections is made too large, so that the integral part of the number will not fit into the remaining sections available for it, an alarm indication is given. Thus the second part of the program is effectively variable-precision arithmetic and an interesting check on the accuracy of the results can be obtained by repeating the calculation with one extra length in the fractional part of the numbers. The codiagonal form which is the final result of this part of the program is stored in the form of two vectors of multiple-precision floating-point numbers. As an indication of the time involved, a matrix of order 24 is reduced to almost triangular form in 1 min and to codiagonal form in a further 2 min 20 sec.

No provision has been made for using an initial set of multiples before starting the elimination proper in the second part of the program (i.e. we take $\boldsymbol{v}_1 = \boldsymbol{e}_1$) though this might be useful and could be quite easily incorporated in the program. As mentioned above, little difficulty has been caused by large multiples when eliminating the subdiagonal elements, but it is true that a codiagonal form may arise which is unstable in the sense that its eigenvalues are sensitive to small changes in its elements. This would be a rare occurrence but a provision for changing $\boldsymbol{v}_1$ would permit one easily (without recalculating the almost triangular matrix) to derive a different codiagonal form which would probably be more stable—the unstable case being the exception. However, we are able to reverse the original matrix about the principal anti-diagonal and this provides a simple way of forming a different codiagonal form via a different almost triangular stage. We can also transpose the original matrix, which leads to a different almost triangular form, but to the same codiagonal form as that produced from the untransposed matrix (the minors of the first elements of $|\boldsymbol{A} - \lambda \boldsymbol{I}|$ and $|\boldsymbol{A}' - \lambda \boldsymbol{I}|$ are the same). This gives us a check on the errors introduced by the first part of the program.

The method compares very favourably with that of Lanczos (starting from the original matrix) in the amount of work and space required by the computation. If single-precision arithmetic were used throughout, the elimination method would involve about n^3 multiplications and use one matrix space, while the Lanczos method would involve $4n^3$ multiplications and need three matrix spaces. Greater precision increases the advantage of the elimination method.

One point of detail may be of interest. When the program was first written it was thought that the occurrence of an exact zero as an eliminating element either above or below the diagonal would be such a rare event that it would be sufficient merely to stop if it occurred. A zero as the eliminating element above the diagonal would imply that all the remaining elements in that row were exactly zero. In practice this has happened sufficiently often to be a nuisance, and it became desirable to modify the program so that it no longer stopped. Fortunately, since we know that the rest of that row must also be zero, no further eliminations are needed and it is sufficient merely to skip to the next row. Zeros or very small numbers have also occurred as elements to be eliminated in a column below the diagonal. In this case it is sufficient to treat a zero eliminating element as a half in the last fractional place.

The reason for this state of affairs seems to be that the original matrices are far from general, but quite often appear to be built up out of several sub-matrices in such a way that the large matrix can be factorized by inspection, or at most with very little numerical work. A further complication is introduced by the fact that these factors are often repeated, which makes the problem of finding the eigenvalues of the large matrix very considerably harder. In an extreme case of this sort a matrix of order 18 could be reduced, by permutations only, to three identical matrices of order 6, and these in turn could be factorized into three matrices of order 2. It was thus possible to find all the 18 eigenvalues of the original matrix merely by solving three quadratic equations.

This kind of thing is an example of the sort of hazard which the designer of general-purpose eigenvalue routines must be prepared to face. It is easy to imagine circumstances in which it would be perfectly legitimate to pose the problem of finding the eigenvalues of a rather special matrix of this sort, and it seems quite reasonable to ask any general-purpose eigenvalue routine to cope with it. Fortunately, this sort of matrix provides no problems for the codiagonalizing routine; but neither are the difficulties removed—they are merely passed on to the root-finding routine.

In order to find the eigenvalues it is still necessary to find the roots of the codiagonal form, and as the whole process is designed to deal with unsymmetric matrices it is essential to consider complex roots. This is by no means a trivial problem. However, as we are not immediately concerned with it in this paper, we merely state that the Pegasus program includes a root finder based on the process due to Muller (Frank, 1958) which finds complex roots one at a time.

References

Frank, W. L. (1958). "Finding Zeros of Arbitrary Functions," *J. Assoc. Comput. Mach.*, Vol. 5, p. 154.

Lanczos, C. (1950). "An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators," *J. Res. Nat. Bur. Stand.*, Vol. 45, p. 255.

Wilkinson, J. H. (1958). "The Calculation of Eigenvectors by the Method of Lanczos," *The Computer Journal*, Vol. 1, p. 148.

Wilkinson, J. H. (1959). "Stability of the Reduction of a Matrix to Almost Triangular and Triangular Forms by Elementary Similarity Transformations," *J. Assoc. Comput. Mach.*, Vol. 6, p. 336.

Appendix

The program given opposite is an informal publication-language version of an ALGOL program to reduce a matrix to a codiagonal form using the method described above. The principal departures from standard ALGOL are:

(1) A determined effort to make use of layout and other typographical devices to make the program easier to read. Statements are placed on separate lines and are not terminated by semicolons.
(2) The use of "=" for ":=" and "otherwise" for "else."
(3) The inclusion in comments of certain vital information concerning precision which cannot be adequately expressed in ALGOL.

Experience of attempting to communicate programs in any generalized notation has shown the value of having sets of numerical examples with which to test the program, if possible with intermediate results as well as the final one. Two such sets are given below; the nearly triangular form is an intermediate result which appears at the point indicated in the program. The test matrices are given as integers and the results are exact. For a program using fixed-point fractions they should be scaled—say by multiplying by 0·01. The eigenvalues are also given for interest.

	Test 1			*Test* 2			
Matrix	4	3	1	0	1	2	2
	6	13	3	4	11	0	11
	−6	−13	1	3	7	−7	2
				−4	−9	7	−4
Near Triangle	4	3	0	0	2	0	0
	4	26/3	4/9	1	0	7/2	0
	−6	−13	16/3	0	7	−3/2	3/2
				−4	7	−25/2	3/2
Codiagonal	4	1	0	0	1	0	0
	12	8	1	2	0	1	0
	0	−4	6	0	7/2	0	1
				0	0	9/2	0
Eigenvalues	2.24123...			−3			
	6.69459...			−1			
	9.06418...			1			
				3			

PROGRAM

```
Procedure  Form Codiagonal ($a$) order ($n$) result ($\alpha$, $\beta$)
Value  $n$
Integer  $n$
Array  $a[1 : n, 1 : n]$, $\alpha[1 : n]$, $\beta[2 : n]$                    Comment  See note 1
Begin  Real $x$, $w$                                                          Comment  See note 2
    Integer $r$, $m$, $k$, $j$
    For $r = 1$ step 1 until $n - 2$ do
      Begin  $m = r + 1$                                                      Comment  Find pivot
        For $k = r + 2$ step 1 until $n$ do
          If $|a_{r,k}| > |a_{r,m}|$ then $m = k$
        If $m \neq r + 1$ then                                                Comment  Interchange if necessary
          Begin  For $k = 1$ step 1 until $n$ do
                   Begin  $x = a_{m,k}$
                     $a_{m,k} = a_{r+1,k}$
                     $a_{r+1,k} = x$
                   End
                 For $k = r$ step 1 until $n$ do
                   Begin  $x = a_{k,m}$
                     $a_{k,m} = a_{k,r+1}$
                     $a_{k,r+1} = x$
                   End
          End
        If $a_{r,r+1} \neq 0$ then                                            Comment  Eliminate row $r$
          For $k = r + 2$ step 1 until $n$ do
            Begin  $x = a_{r,k}/a_{r,r+1}$
              For $j = r + 1$ step 1 until $n$ do
                $a_{j,k} = a_{j,k} - x . a_{j,r+1}$
              For $j = 1$ step 1 until $n$ do
                $a_{r+1,j} = a_{r+1,j} + x . a_{k,j}$
            End
      End                                                                     Comment  Near Triangle
    $\alpha_1 = a_{11}$                                                       Comment  Set up first column
    For $k = 2$ step 1 until $n$ do
      Begin  $\beta_k = a_{k,1}$
        $\alpha_k = a_{k,2}$
      End
    For $r = 1$ step 1 until $n - 2$ do                                       Comment  Eliminate column $r$.
      Begin  $\beta_{r+1}$ = If $\beta_{r+1} \neq 0$ then $\beta_{r+1}$ otherwise $\varepsilon$     See note 3
        For $k = r + 2$ step 1 until $n$ do
          $\beta_k = \beta_k/\beta_{r+1}$
        For $k = r + 1$ step 1 until $n$ do
          For $j = r + 2$ step 1 until (If $k + 1 \leqslant n$ then $k + 1$ otherwise $n$) do
            $\alpha_k = \alpha_k + \beta_j . a_{k,j}$
        For $k = r + 2$ step 1 until $n$ do
          Begin  $w = \beta_k$
            $\beta_k = \alpha_k - w . \alpha_{r+1}$
            $\alpha_k = a_{k,r+2} - w . a_{r+1,r+2}$
          End
      End
    For $r = 2$ step 1 until $n$ do                                           Comment  Scale across diagonal
      $\beta_r = \beta_r . a_{r-1,r}$
End  Form Codiagonal
```

Comment *Notes.*—(1) The matrix a is usually considered to be fixed point with its elements in the range $(-1, 1)$. Floating point will obviously do equally well but is not necessary. No provision has been made for fixed-point overflow, but this is unusual if the elements lie in a somewhat smaller range, e.g. $(-0\cdot 1, 0\cdot 1)$.

The vectors α, β which hold the result must be floating point and should preferably be multiple precision (double length seems enough almost always). For smallish non-pathological matrices single length should be sufficient.

(2) x is of the same type as a—i.e. single length fixed or floating point. w is of the same type as α and β, i.e. floating point, multiple precision if possible.

(3) If the eliminating element β_{r+1} is exactly zero it is replaced by ε which should be a very small number. If the near triangle has been formed fixed point, ε should be rather less than one unit in the least significant place kept.

Correspondence

Continued from p. 167

Sir,

The authors' reply:

Mr. Watt's letter raises some interesting points, to which we would like to reply.

Firstly, if it were not possible to use statements as parameters of more complex statements, this would indeed be restrictive. However, by suitably defining the class of "auxiliary statements" we can always avoid this difficulty. It is true that for languages like ALGOL and Nebula, the class of "source statements" proves less useful than in the case of more "primitive" languages such as Fortran and Mercury Autocode, where this class is comparatively large: in the former case emphasis is placed almost entirely on the class of auxiliary statements.

The first example given by Mr. Watt could be treated by means of the existing proposals were it not for a quite different kind of difficulty, which arises in connection with the definition of [GE], namely

$$[GE] = [GE]\,[\pm]\,[T],\ [\pm\ ?]\,[T]$$

This type of recursive definition cannot be used in conjunction with a forward scanning recognition routine, because it would be continually searching for a [GE]! Instead, it has to be recast thus:

$$[GE] = [\pm\ ?]\,[GE']$$
$$[GE'] = [T]\,[\pm]\,[GE'],\ [T]$$

With this definition of [GE] Mr. Watt's example becomes, in the notation of our January paper, as follows:

```
    statement definition: A = [GE′]
    → 1 if [GE′] ≡ [T]
    let [GE′] ≡ [T] [±] [GE′]
    A = [GE′]
    A = [±]A + [T]
    end
1]  A = [T]
    end
```

One reason why the class of secondary statements and the class of auxiliary statements are treated exceptionally is because they are large classes and are defined in a cumulative manner. At any stage they can be regarded as complete and are used for recognizing such statements in statement definitions. There is, nevertheless, something in what Mr. Watt says, in so far as it may be desirable to treat other classes in this same cumulative fashion, and associate specific compiling routines (i.e. "statement definitions") with each member of these classes. At present the meanings of a phrase are embodied in the statement definitions of the formats which employ them (and indeed may be different in different contexts). If a phrase has several alternative forms this is reflected in the relevant statement definitions by the appearance of a multi-way switch (e.g. β_1 = category of [Y], $\rightarrow \beta_1$) or other means of discrimination. In the case of those phrases which we would like to define in a cumulative manner and which have many members, the corresponding statement definitions would become unwieldy, and it is convenient to be able to call in a routine to deal with the appropriate category of the phrase on hand. The need for this was not very apparent, however, in the study of Mercury Autocode, Fortran, or even ALGOL, but first showed up in some preliminary studies of Nebula, where an example of such a class is the "logical description statement" (see Nebula Manual, Ferranti, November 1960). We have, therefore, generalized the conception of cumulative classes to take account of this.

Yours faithfully,

R. A. Brooker, D. Morris.

IFIP CONGRESS 62—Call for Papers

The International Federation of Information Processing Societies (IFIPS) will hold a Congress in Munich, Germany, from 27 *August to* 1 *September* 1962.

The Congress will cover all aspects of Information Processing and Digital Computers. An outline of the proposed programme of the Congress was given in **The Computer Journal,** Vol. 4, p. 19 (April 1961). Those wishing to offer papers are invited to send abstracts of 500–1,000 words to:

M. V. Wilkes,
The British Computer Society,
c/o University Mathematical Laboratory,
Corn Exchange Street,
Cambridge,

by 15 *September* 1961. These abstracts will be considered by the international program committee of IFIPS, and authors of selected abstracts will be invited to submit their complete papers (in French or English) for consideration by the program committee in March 1962.