Automatica, Vol. 18, No. 4, pp. 415–430, 1982
Printed in Great Britain.

0005-1098/82/040415-16$03.00/0
Pergamon Press Ltd.

Computation of Zeros of Linear Multivariable Systems*

A. EMAMI-NAEINI† and P. VAN DOOREN‡

Unitary transformations, performed on the system matrix of a linear system, provide numerically stable computations of its zeros.

Key Words—Transmission zeros; decoupling zeros; multivariable systems; state-space methods; stability of numerical methods.

Abstract—Several algorithms have been proposed in the literature for the computation of the zeros of a linear system described by a state-space model $\{A, B, C, D\}$. In this paper we discuss the numerical properties of a new algorithm and compare it with some earlier techniques of computing zeros. The method is a modified version of Silverman's structure algorithm and is shown to be backward stable in a rigorous sense. The approach is shown to handle both nonsquare and/or degenerate systems. Several numerical examples are also provided.

1. INTRODUCTION

DURING the past decade, considerable attention has been paid to the computation of the zeros of a linear multivariable system and especially to the development of reliable numerical software for this problem. Zeros of a multivariable system play an important role in several problems of control theory, such as the study of regulation, robust servomechanism design, and decoupling (Davison, 1976; Davison and Wang, 1974; Desoer and Schulman, 1974; Francis and Wonham, 1975; Franklin, 1978).

Consider the linear time invariant system

$$\begin{aligned} \lambda x &= A_{nn}x + B_{nm}u \\ y &= C_{pn}x + D_{pm}u \end{aligned} \tag{1}$$

where x, u, and y are the state vector, control vector, and output vector, respectively, and where λ can be the differential operator or the delay operator. The *transfer function* of the system $\{A, B, C, D\}$ given in (1), is the $p \times m$ rational matrix $R(\lambda) = D + C(\lambda I_n - A)^{-1}B$. Its *system matrix* is the $(n+p)\times(n+m)$ pencil

$$S(\lambda) = \begin{bmatrix} \lambda I - A & B \\ \underbrace{-C}_{n} & \underbrace{D}_{m} \end{bmatrix} \begin{matrix} \}n \\ \}p \end{matrix}. \tag{2}$$

The *Smith zeros* (Gantmacher, 1959; Rosenbrock, 1970) of (2) are commonly called the *invariant zeros* of the system (1). When $p = 0$, these are the *input decoupling zeros* of the system, and when $m = 0$, these are the *output decoupling zeros* of the system. When the system is minimal, these are the *transmission zeros* of the system (see MacFarlane and Karcanias, 1976, for an elaborate discussion).

Note that in Davison and Wang (1974, 1976, 1978) and Laub and Moore (1978), the Smith zeros of (2) are called the transmission zeros of this system. Moreover these authors restrict themselves to what they call the *nondegenerate case*, i.e. where the *normal rank* r of $R(\lambda)$ equals $\min(m, p)$, or equivalently, where the normal rank $n + r$ of $S(\lambda)$ equals $n + \min(m, p)$. Zeros have also been defined for the *degenerate case* (McMillan, 1952; Gantmacher, 1959; Rosenbrock, 1970; Moore and Silverman, 1974) and have been interpreted from a physical viewpoint as well (Desoer and Schulman, 1974; MacFarlane and Karcanias, 1976; Kouvaritakis and MacFarlane, 1976). The Smith zeros of $S(\lambda)$ are indeed the points where the rank of $S(\lambda)$ drops below its normal rank $n + r$, and this holds as well for the degenerate case as for the nondegenerate case. When $S(\lambda)$ is minimal these are also the *McMillan zeros* of the transfer function $R(\lambda)$. In the sequel we make no special distinction anymore between the different types of zeros discussed above (decoupling, invariant,

*Received 26 August 1980; revised 28 July 1981; revised 4 February 1982. The original version of this paper was not presented at any IFAC meeting. This paper was recommended for publication in revised form by associate editor E. J. Davison.

†Information Systems Laboratory, Stanford University, Stanford, California 94305, U.S.A. Present address: Systems Control Inc., 1801 Page Mill Road, Palo Alto, CA 94304, U.S.A.

‡Information Systems Laboratory and Department of Computer Science, Stanford University, Stanford, CA 94305, U.S.A. Present address: Philips Research Laboratory, Av. Van Becelaere 2, B-1170 Brussels, Belgium.

transmission) since they are all the *zeros* of a specific type of system matrix (2).

In this paper we give a 'fast' implementation of a method that was designed to tackle the computation of the zeros of an arbitrary state-space system (1). The algorithm borrows ideas from Silverman's structure algorithm as modified by Moylan (Silverman, 1976; Moylan, 1977) and is based on the numerical principles discussed in Van Dooren (1979) (see also, Emami-Naeini, 1978; Van Dooren, Emami-Naeini and Silverman, 1979; Emami-Naeini, Van Dooren and Silverman, 1980; Van Dooren, 1981). We also compare this algorithm with those of Davison and Wang (1974, 1976, 1978) and Moore and Laub (1978), two methods with 'controlled numerical behavior'. We first briefly review the methods.

The first technique (as described in Davison and Wang, 1978) uses the invariance property of zeros under high gain output feedback to determine their locations. Assume without loss of generality that $m \geq p$ (a dual method is used in the other case) and let K be a 'random' $m \times p$ matrix. If the system $\{A, B, C, D\}$ is nondegenerate, then for 'almost all' K, the system $\{A, BK, C, DK\}$ is also nondegenerate and the zeros of the system $\{A, B, C, D\}$ are a subset of the zeros of the system $\{A, BK, C, DK\}$. The matrix $DK - I/\rho$ is also invertible for 'almost all' K and the eigenvalues of the matrix $A - BK(DK - I/\rho)^{-1}C$ are then the zeros of the system matrix

$$S\rho(\lambda) = \begin{bmatrix} \lambda I - A & BK \\ -C & DK - I/\rho \end{bmatrix}. \tag{3}$$

When ρ goes to infinity, $S_\rho(\lambda)$ converges to $S(\lambda)$ and the above eigenvalues thus converge to the zeros of the system $\{A, BK, C, DK\}$ or to infinity. One then proceeds as follows: compute the limiting eigenvalues of $A - BK(DK - I/\rho)^{-1}C$ by running an eigenvalue routine (Garbow and co-workers, 1977) on this matrix for several values of ρ; repeat this for another 'random' matrix K, and select the eigenvalues that are common to both runs. The other eigenvalues (so called 'extraneous zeros') do not correspond to zeros of the system $\{A, B, C, D\}$. The advantage of this method is its basic simplicity and the availability of reliable software to implement the method. Its possible disadvantages are (i) the sorting of true zeros and extraneous zeros which may be unclear in some situations and (ii) the possible ill conditioning introduced by the inversion of $DK - I/\rho$ (see Laub and Moore, 1978, for a discussion of this phenomenon). Moreover the method only works for nondegenerate systems; a test is built in to check this property (see Davison and Wang, 1978, for more details). For the computation of decoupling zeros a different but related method is given by Davison, Gesing and Wang (1978) which does not suffer from the second disadvantage. The second method is discussed in Laub and Moore (1978), and also works only for nondegenerate systems. In the square case, the QZ algorithm (Moler and Stewart, 1973) is used to determine the generalized eigenvalues of the square invertible system matrix

$$S(\lambda) = \begin{bmatrix} \lambda I - A & B \\ -C & D \end{bmatrix}. \tag{4}$$

Since this system matrix is a regular matrix pencil, its generalized eigenvalues are also the zeros of the system (Laub and Moore, 1978). For the nonsquare case, random rows (columns) are added to the matrices C and D (B and D) if $m > p$ ($m < p$), in order to obtain a modified but square pencil of the type (4). This random 'bordering' is performed twice and the common generalized eigenvalues of both runs can be shown to be the zeros of the system $\{A, B, C, D\}$. The others are again extraneous zeros. The method only works for nondegenerate systems and a test for this is provided by the QZ algorithm (Moler and Stewart, 1973). The advantage of this method over the first one is the numerical stability of the QZ algorithm in contrast with the possible instability introduced by the inversion of $DK - I/\rho$. However, the sorting has still to be performed and the 'squaring up' via bordering should rather be replaced by the multiplication trick of the previous method since this results in a generalized eigenvalue problem of dimension $n + \min(m, p)$ instead of $n + \max(m, p)$. Another simple method of 'squaring up' the system matrix is reported by Porter (1979), but it increases the size of the pencil to be processed.

The third technique of computing zeros is based on the Kronecker canonical form (Gantmacher, 1959) of the system matrix $S(\lambda)$ and on recent methods for computing it (Emami-Naeini, 1978, Van Dooren, 1979; Van Dooren, Emami-Naeini and Silverman, 1979; Emami-Naeini, Van Dooren and Silverman, 1980; Van Dooren, 1981). The system matrix (4) can always be transformed by unitary transformations P and Q to the generalized (upper) Schur form

$$PS(\lambda)Q = \begin{bmatrix} A_r - \lambda B_r & * & * & * \\ 0 & A_f - \lambda B_f & * & * \\ 0 & 0 & A_i - \lambda B_i & * \\ 0 & 0 & 0 & A_l - \lambda B_l \end{bmatrix} \tag{5}$$

where (i) $(A_r - \lambda B_r)$ and $(A_l - \lambda B_l)$ are non-square pencils with no zeros and revealing the right and left minimal indices of $S(\lambda)$, respectively; (ii) $(A_i - \lambda B_i)$ is a regular pencil with no finite zeros and revealing the infinite zeros of $S(\lambda)$; and (iii) $(A_f - \lambda B_f)$ is a regular pencil whose generalized eigenvalues are the finite zeros of $S(\lambda)$.

After this preliminary reduction, the QZ algorithm is applied to the 'finite structure' pencil $A_f - \lambda B_f$, which contains the zeros of the system matrix $S(\lambda)$ (Van Dooren, 1979). The overall procedure is proved to be numerically backward stable in a strict sense, namely that the computed zeros correspond exactly to a slightly perturbed system $\{\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D}\}$, with $\begin{bmatrix} \tilde{A} & \tilde{B} \\ \tilde{C} & \tilde{D} \end{bmatrix}$ being $\pi \cdot \epsilon$-close to $\begin{bmatrix} A & B \\ C & D \end{bmatrix}$, and where ϵ is the machine precision of the computer and π is a polynomial expression in the dimensions of the system. This does not hold for the method of Davison and Wang. While the left and right minimal indices and the multiple infinite zeros are also determined by the decomposition (5), they are exactly the cause of some problems encountered by the two previous methods. Note that the pencils $A_r - \lambda B_r$, $A_i - \lambda B_i$ and $A_l - \lambda B_l$ also contain valuable information about the system $\{A, B, C, D\}$ (Verghese, Van Dooren and Kailath, 1979; Van Dooren, 1981).

This third method is also related to earlier (sometimes implicit) methods to compute the information contained in the Kronecker canonical form of $S(\lambda)$: Aplevich (1979); Jordan and Godbout (1977); Molinari (1978); Moore and Silverman (1974); Moylan (1977); Silverman (1976); Thompson and Weil (1972). These algorithms are more conceptual methods and may run into numerical difficulties since they use possibly unstable transformations. We therefore did not include them in our comparison.

The organization of this paper is as follows. In Section 2 the new algorithm for computing zeros is presented. In Section 3 we discuss the properties of the algorithm and compare it with other methods. In Section 4 several numerical examples are presented. Some concluding remarks appear in Section 5. The data used in some of the numerical examples as well as a listing of the implementation of the algorithm in FORTRAN, can be found in Emami-Naeini and Van Dooren (1980). A listing of the subroutine codes is also included in the appendix of this paper.

2. THE NEW ALGORITHM: SYSTEM MATRIX REDUCTION

In this section all matrices are assumed to be complex. For the real case the algorithms require only minor modifications. A^* denotes the conjugate transpose of A, and A^T denotes the transpose of A.

2.1. *Reduction method*

In this section we present a method to construct a reduced order system matrix

$$S_r(\lambda) = \left[\begin{array}{c|c} \lambda I - A_r & B_r \\ \hline -C_r & D_r \end{array}\right] \tag{6}$$

with D_r invertible and with $S_r(\lambda)$ having the same (finite) zeros as a given system matrix

$$S(\lambda) = \left[\begin{array}{c|c} \lambda I - A & B \\ \hline -C & D \end{array}\right] \tag{7}$$

with no restictions on $\{A, B, C, D\}$. This algorithm is then used as the heart of the algorithm for the computation of multivariable zeros. The latter is based on the generalized Schur form of the pencil $S(\lambda)$ as discussed above, but differs from it in that the special structure of the pencil is exploited to the fullest in all the necessary computations.

The algorithms only use matrix reductions of the type

$$U^*A = \begin{bmatrix} A_r \\ \hline 0 \end{bmatrix}\}\rho; \qquad AV = [\underbrace{A_c}_{\rho} \quad 0] \tag{8}$$

where A is an arbitrary matrix of rank ρ and U and V are unitary matrices *compressing* A to a *full row rank* matrix A_r and *full column rank* matrix A_c, respectively. Several techniques can be used for this purpose; more details are given in Section 2.2. The algorithm is stated in an ALGOL type language:

Algorithm REDUCE (A, B, C, D, m, n, p) result $(A_r, B_r, C_r, D_r, m_r, n_r, p_r)$

comment initialization;

$A_0 := A$; $B_0 := B$; $C_0 := C$; $D_0 := D$; $\nu_0 := n$;
$\delta_0 := 0$; $\mu_0 := p$; $j := 1$;

step_j: *comment* compress the rows of D_{j-1} with U_j^* and transform simultaneously the rows of C_{j-1};

$$\begin{matrix} \sigma_j \{ \\ \tau_j \{ \end{matrix} \left[\begin{array}{c|c} \bar{C}_{j-1} & \bar{D}_{j-1} \\ \hline C_{j-1} & 0 \end{array}\right] := U_j^* [C_{j-1} | D_{j-1}];$$

if $\tau_j = 0$ *then go to* exit_1;

comment compress the columns of $\bar{C}_{j-1}$ with V_j ($\bar{C}_{j-1} = C_{j-1}$ if $\sigma_j = 0$);

if $\rho_j = 0$
then go to exit_1;
$[\underbrace{0}_{\nu_j} \quad \underbrace{S_j}_{\rho_j}] := \tilde{C}_{j-1} V_j$;
if $\nu_j = 0$
then go to exit_2;

comment update;

$$\mu_j := \rho_j + \sigma_j;\ \delta_j := \delta_{j-1} + \rho_j;$$

$$\begin{matrix}\nu_j\{\\ \mu_j\{\end{matrix}\left[\begin{array}{ccc|c} A_j & * & & B_j \\ \underbrace{C_j}_{\nu_j} & \underbrace{*}_{\rho_j} & & \underbrace{D_j}_{m} \end{array}\right] := \left[\begin{array}{c|c} V_j^* & 0 \\ \hline 0 & I_{\sigma_j} \end{array}\right]\left[\begin{array}{c|c} A_{j-1} & B_{j-1} \\ \hline \tilde{C}_{j-1} & \tilde{D}_{j-1} \end{array}\right] \times \left[\begin{array}{c|c} V_j & 0 \\ \hline 0 & I_m \end{array}\right];$$

$j := j + 1$; *go to* step_j;

exit_1: *comment* $\{A, B, C, D\}$ and $\{A_r, B_r, C_r, D_r\}$ have the same zeros;

$k := j - 1$; $A_r := A_k$; $B_r := B_k$; $C_r := \tilde{C}_k$;
$D_r := \tilde{D}_k$; $n_r := \nu_k$; $p_r := \sigma_j$: $m_r := m$;

stop;

exit_2: *comment* $\{A, B, C, D\}$ has no zeros;

$k := j$; $n_r := 0$; *stop*.

Theorem 1. The systems $\{A, B, C, D\}$ and $\{A_r, B_r, C_r, D_r\}$ have the same (finite) zeros.

Proof: We prove the result by induction.

Step j of the above algorithm reduces the system matrix of

$\{A_{j-1}, B_{j-1}, C_{j-1}, D_{j-1}\}$ to the form

$$\left[\begin{array}{c|c} V_j^* & 0 \\ \hline 0 & U_j^* \end{array}\right]\left[\begin{array}{c|c} \lambda I_{\nu_{j-1}} - A_{j-1} & B_{j-1} \\ \hline -C_{j-1} & D_{j-1} \end{array}\right]\left[\begin{array}{c|c} V_j & 0 \\ \hline 0 & I_m \end{array}\right] = \left[\begin{array}{c|c|c} \lambda I_{\nu_j} - A_j & * & B_j \\ \hline -C_j & * & D_j \\ \hline 0 & -S_j & 0 \end{array}\right] \tag{9}$$

where S_j has full column rank ρ_j. Using S_j as the pivot, (9) can be transformed by unimodular row and column transformations to

$$P_j(\lambda)\left[\begin{array}{cc} \lambda I_{\nu_{j-1}} & B_{j-1} \\ -C_{j-1} & D_{j-1} \end{array}\right], -Q_j(\lambda) = \left[\begin{array}{c|c|c} \lambda I_{\nu_j} - A_j & B_j & 0 \\ \hline -C_j & D_j & \\ \hline 0 & & \begin{array}{c|c} I_{\rho_j} & \\ \hline 0 & \end{array} \end{array}\right] \tag{10}$$

and the systems $\{A_k, B_k, C_k, D_k\}$ for $k = j-1, j$ have thus the same zeros.

□

The following algorithm shows how REDUCE can be used to compute a pencil $(\lambda B_f - A_f)$ with only finite generalized eigenvalues which are the zeros of the system $\{A, B, C, D\}$. In order to compute the zeros, we then use the QZ algorithm on $(\lambda B_f - A_f)$.

Algorithm ZEROS (A, B, C, D, m, n, p) result $(A_f, B_f, n_f, \text{rank})$

step_1: *comment* reduce the system $\{A, B, C, D\}$ to a new system $\{A_r, B_r, C_r, D_r\}$ with the same zeros and with D_r of full row rank; *call* REDUCE (A, B, C, D, m, n, p) result $(A_r, B_r, C_r, D_r, m_r, n_r, p_r)$; rank: $= m_r$; *if* $n_r = 0$ *then begin* $n_f := 0$; *go to* exit *end*;

step_2: *comment* reduce the transposed system $\{A_r^T, C_r^T, B_r^T, D_r^T\}$ to a new system $\{A_{rc}, B_{rc}, C_{rc}, D_{rc}\}$ with the same zeros and with D_{rc} invertible; *call* REDUCE $(A_r^T, C_r^T, B_r^T, D_r^T, p_r, n_r, m_r)$ result $(A_{rc}, B_{rc}, C_{rc}, D_{rc}, m_{rc}, n_{rc}, p_{rc})$; *if* $n_{rc} = 0$ *then begin* $n_f := 0$; *go to* exit *end*;

step_3: *comment* compress the columns of $[C_{rc} \quad D_{rc}]$ to $[0 \quad D_f]$ and apply the transformation to the system matrix; $n_f := n_{rc}$; *if* rank $= 0$ *then go to* exit;

$$\left[\begin{array}{c|c} A_f & * \\ \hline 0 & D_f \end{array}\right] := \left[\begin{array}{c|c} A_{rc} & B_{rc} \\ \hline C_{rc} & D_{rc} \end{array}\right] W; \left[\begin{array}{c|c} B_f & * \\ \hline 0 & 0 \end{array}\right] := \left[\begin{array}{c|c} I & 0 \\ \hline 0 & 0 \end{array}\right] W;$$

exit: *stop*;

Theorem 2. The (finite) zeros of the system $\{A, B, C, D\}$ are the generalized eigenvalues of the 'finite structure pencil' $(\lambda B_f - A_f)$.

Proof: In the first step of ZEROS the routine REDUCE yields a system matrix

$$S_r(\lambda) = \left[\begin{array}{c|c} \lambda I_{n_r} - A_r & B_r \\ \hline -C_r & \underbrace{D_r}_{m_r} \end{array}\right]\begin{matrix}\}n_r \\ \}p_r\end{matrix} \tag{11}$$

with the same zeros as $S(\lambda)$ but with D_r of full row rank. In the second step, the transposed system matrix $S_r^T(\lambda)$, which still has the same zeros, is reduced again by REDUCE to a system matrix of the form

$$S_{rc}(\lambda) = \left[\begin{array}{c|c} \lambda I_{n_{rc}} - A_{rc} & B_{rc} \\ \hline -C_{rc} & \underbrace{D_{rc}}_{m_{rc}} \end{array}\right]\begin{matrix}\}n_{rc} \\ \}p_{rc}\end{matrix} \tag{12}$$

$\underbrace{\qquad}_{n_{rc}}$

where now D_{rc} has full row rank. Note that D_r^T had full column rank originally and REDUCE does not decrease the rank of this matrix. Therefore D_{rc} has full column rank as well and

thus is invertible. The third step transforms $S_c(\lambda)$ to

$$\begin{matrix} n_{rc}\{ \\ \text{rank}\{ \end{matrix}\left[\begin{array}{c|c} \lambda I_{n_{rc}} - A_{rc} & B_{rc} \\ \hline \underbrace{-C_{rc}}_{n_{rc}} & \underbrace{D_{rc}}_{\text{rank}} \end{array}\right]$$

$$W = \left[\begin{array}{c|c} \lambda B_f - A_f & * \\ \hline \underbrace{0}_{n_f} & \underbrace{D_r}_{\text{rank}} \end{array}\right] \begin{matrix} \}n_f \\ \}\text{rank} \end{matrix} \qquad (13)$$

where $n_f = n_{rc}$ and rank $= m_{rc} = p_{rc}$ and D_f is, of course, invertible. Since $S_{rc}(\lambda)$ has $n_{rc} = n_f$ finite zeros (namely the eigenvalues of $\hat{A} = A_{rc} - B_{rc}D_{rc}^{-1}C_{rc}$) then the $(n_f \times n_f)$ pencil $(\lambda B_f - A_f)$ has only finite generalized eigenvalues, and they are the zeros of $S(\lambda)$.

□

The zeros are now computed by the *QZ* method (Moler and Stewart, 1973; Garbow and co-workers, 1977), which decomposes $(\lambda B_f - A_f)$ into

$$Q(\lambda B_f - A_f)Z = \lambda \begin{bmatrix} \beta_1 & & * \\ & \ddots & \\ 0 & & \beta_{n_f} \end{bmatrix} - \begin{bmatrix} \alpha_1 & & * \\ & \ddots & \\ 0 & & \alpha_{n_f} \end{bmatrix} \qquad (14)$$

where Q and Z are unitary.

The ratios $\lambda_i = (\alpha_i/\beta_i)$, $i = 1, \ldots, n_f$ are then the (finite) zeros of $S(\lambda)$. It should be noted that this path is to be preferred over the use of the *QR* algorithm on the matrix $\hat{A} = A_{rc} - B_{rc}D_{rc}^{-1}C_{rc}$, because of the possible bad conditioning of D_{rc}.

Remarks.

(1) The reductions performed by ZEROS and REDUCE can be rewritten (up to some permutations) as a decomposition of the type (5). Therefore the infinite zero structure and the left and right null space structure can also be retrieved by these algorithms (Van Dooren, 1979). If $S(\lambda)$ is minimal these are also the infinite transmission zeros and left and right minimal indices of the transfer function $R(\lambda)$ (Verghese, Van Dooren and Kailath, 1979). Note that $m_{rc} = p_{rc} =$ rank is the normal rank of the transfer function $R(\lambda)$. When rank = 0 then step_3 of ZEROS can be skipped and we get the standard eigenvalue problem since $B_f = I$.

(2) When the system $\{A, B, C, D\}$ is real, the transformations in REDUCE and ZEROS are also real. However, the decomposition in (14) has to be slightly modified so that Q and Z are real. Under orthogonal transformations one can indeed only reduce $(\lambda B_f - A_f)$ to a *block triangular form* (Moler and Stewart, 1973)

$$Q(\lambda B_f - A_f)Z = \lambda \begin{bmatrix} B_{11} & & * \\ & \ddots & \\ 0 & & B_{kk} \end{bmatrix} - \begin{bmatrix} A_{11} & & * \\ & \ddots & \\ 0 & & A_{kk} \end{bmatrix} \qquad (15)$$

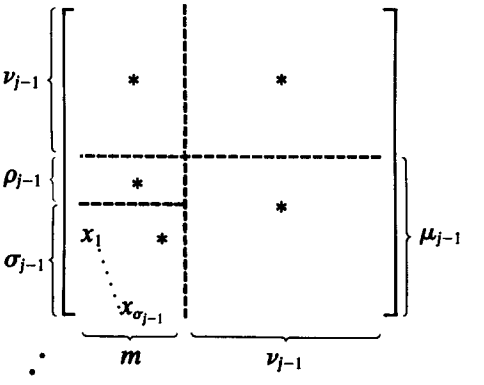
where the $(\lambda B_{ii} - A_{ii})$ blocks are 1×1 or 2×2. The generalized eigenvalues of the 2×2 blocks are complex conjugate pairs and those of the 1×1 blocks are real.

2.2. *Details of implementation*

In order to take full advantage of the special problem at hand and in order to increase numerical accuracy and speed, the structure of the pencil must be fully exploited. Therefore we use Householder transformations for the row and column operations in these algorithms. Special care is also taken to exploit the previously created zeros at each stage of the algorithm. It is more convenient for the organization of the data to deal with the matrix

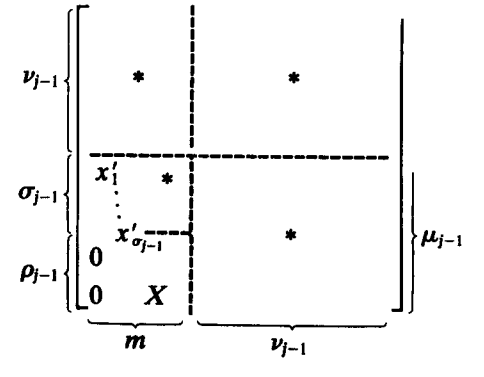
$$\left[\begin{array}{c|c} B_{j-1} & A_{j-1} \\ \hline D_{j-1} & C_{j-1} \end{array}\right]. \qquad (16)$$

At the beginning of step j, (16) has the special form



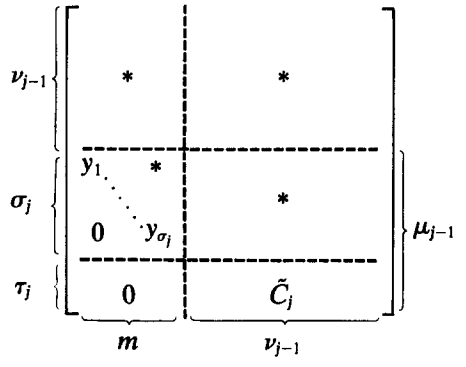
(17)

and the x_is are nonzero (when $j = 1$, we have $\nu_0 = n$; $\mu_0 = p$; $\rho_0 = p$; $\sigma_0 = 0$). Step_j first performs an output transformation U_j^* to compress D_{j-1} to full row rank. Therefore we first use σ_{j-1} Householder transformations without pivoting (since x_is are nonzero) to reduce (17) to



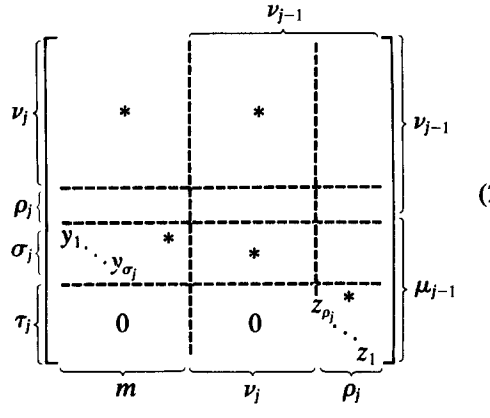
(18)

where again the x_i's are nonzero. We then continue with Householder reduction *with* column pivoting to reduce X to trapezoidal form, yielding finally a row compression of D_{j-1}



(19)

where the y_is are nonzero and $\sigma_j = \sigma_{j-1} + \text{rank}\,(X)$. Step j then continues with a state space transformation V_j to compress the columns of $\tilde{C}_j$. Therefore we use Householder transformations with row pivoting on $\tilde{C}_j$ and obtain



(20)

where the z_is are also nonzero. The last ρ_j columns and τ_j rows can now then be discarded giving (17) again for j updated by one. This process is continued until no reduction is possible anymore (see exit_1 and exit_2 in REDUCE).

In step_3 of ZEROS we also exploit the triangular shape of D_{rc} by reducing a matrix of the form

$$\left[\begin{array}{c|c} B_{rc} & A_{rc} \\ \hline D_{rc} & C_{rc} \end{array}\right] = \left[\begin{array}{c|c} * & * \\ \hline x_1 \cdots * & * \\ 0 \cdots x_{rc} & \end{array}\right] \tag{21}$$

to the form

$$\left[\begin{array}{c|c} A_f & * \\ \hline 0 & D_f \end{array}\right] = \left[\begin{array}{c|c} * & * \\ \hline 0 & x'_1 \cdots * \\ & 0 \cdots x'_{rc} \end{array}\right]. \tag{22}$$

For this we use Householder transformation without pivoting on the columns of (21). For the construction of B_f the same transformations are also performed on the matrix $[I_{n_{rc}} \;\; 0]$.

A comment ought to be made here about the practical implementation of the several rank evaluations performed by REDUCE. As recommended in for example, Golub, Klema and Stewart (1976), the *numerical rank* of a given $s \times t$ matrix M is defined as the number of *singular values* larger than a given threshold EPS. The other singular values are thus in fact put equal to zero, thereby inducing an error bounded by EPS in the matrix M. When M consists of measured data, EPS is the *noise level* of these data; otherwise it is chosen equal to $\pi \cdot \epsilon \cdot \|M\|_2$—where π is a polynomial expression in s and t, ϵ is the machine precision of the computer and $\|\cdot\|_2$ is the spectral norm—which is the *inherent noise level* of computations performed on that computer (Wilkinson, 1965). In order to save some computing time, this rank criterion is often replaced by a Householder reduction on M (with or without row/column pivoting), and checking the number of pivots that are larger than EPS. In REDUCE Householder reductions *with* (row or column) pivoting are used whenever a rank determination is involved, because in that respect it performs better than the Householder reduction without pivoting (see Golub, Klema and Stewart, 1976). This is the case for the reductions of X in (18) and $\tilde{C}_j$ in (19) to trapezoidal forms. Householder transformations *without* pivoting are used whenever rank properties are guaranteed because of the previous steps: the x_i in (17) and (21) are known to be nonzero (i.e. larger than EPS), ensuring the full rank of the corresponding submatrices. It is precisely the use of Householder transformations without pivoting in these cases, that allows us to exploit the previously created zeros in order to save considerable amount of computations (see the next section).

3. PROPERTIES OF THE ALGORITHM AND COMPARISON

Two important properties of our method are its numerical stability and its efficiency.

3.1. *Operation count*

In the sequel, one operation stands for a single addition and multiplication (for the real or complex case). A Householder transformation acting on a $s \times t$ matrix requires $2st$ operations (Wilkinson, 1965). Using this, and the assumption that $m \leq p$, we optain the following operation count for REDUCE. The row compression of D_{j-1} requires at most $\sigma_j \leq m$ Householder transformations, each working on matrices

smaller than $(\rho_{j-1}+1)\times(\nu_{j-1}+m)\leq(\rho_{j-1}+1)\times(n+m)$ [see (17) and (18)]. For this step we thus have less than

$$a_j = 2(\rho_{j-1}+1)(n+m)m \text{ operations.} \quad (23)$$

The state space transformation to compress the columns of $\bar{C}_j$ requires ρ_j Householder transformation, each working on matrices smaller than $(\mu_{j-1}+\nu_{j-1})\times\nu_{j-1}\leq(p+n)\times n$ for V_j, and $(m+\nu_{j-1})\times\nu_{j-1}\leq(m+n)\times n$ for V_j^* [see (19) and (20)]. For this step we thus have less than

$$b_j = 2\rho_j(p+m+2n)n \text{ operations.} \quad (24)$$

If REDUCE requires k steps, then $\Delta=\rho_1+\rho_2+\ldots\rho_{k-1}$ is the amount by which the state dimension is reduced. Using this and the fact that $\rho_0=p$, $\rho_k=0$ and $k\leq n$, we have the total operation count of

$$\sum_{j=1}^{k}(a_j+b_j)\leq 2(p+n+\Delta)(n+m)m + 2\Delta(p+m+2n)n \quad (25)$$

for REDUCE. The routine ZEROS then uses less than the following number of operations

—for step_1; less than:

$$2\{(p+n+\Delta_1)(n+m)m+\Delta_1(p+m+2n)n\} \text{ operations} \quad (26)$$

where $\Delta_1=n-n_r$ is the reduction of the state dimension in the first run of REDUCE

—for step_2; less than:

$$2\{(m+n+\Delta_2)(n+p_r)p_r+\Delta_2(m+p_r+2n)n\} \text{ operations} \quad (27)$$

where $\Delta_2=n_r-n_{rc}$ is the reduction of the state dimension in the second run of REDUCE and where $p_r\leq m$

—for step_3; less than:

$$2\{(n_{rc}+m_{rc})(n_{rc}+1)m_{rc}\} \text{ operations} \quad (28)$$

where $m_{rc}\leq m$.

This last step indeed requires m_{rc} Householder transformations working on matrices smaller than $(n_{rc}+m_{rc})\times(n_{rc}+1)$. Denoting $\Delta=\Delta_1+\Delta_2=n-n_{rc}$ as the number of state reductions, and $M=\max\{p+n,m+n\}=p+n$ we obtain the reduction to (λB_f-A_f) in less than

$$\begin{aligned}&4(p+n)(n+m)m+2\Delta(n+m)m+2\Delta(m+p+2n)n\\&+2(n+m)(n+1)M\\&\leq 4M^2m+2\Delta Mm+4\Delta M^2+2M^2m\\&\leq 6(\Delta+m)M^2\leq 6(\Delta+p)M^2 \text{ operations.}\end{aligned} \quad (29)$$

Notice that $\Delta+p=M-n_{rc}$ is the total reduction of the dimension of $S(\lambda)$ to the pencil λB_f-A_f. ZEROS thus requires less than $6M^2$ operations per deflation, while for example, the QZ method used on $S(\lambda)$ directly, would require approximately $16M^2$ operation per deflation [according to Moler and Stewart (1973) about 1.2–1.3 QZ iterations are needed per computed eigenvalue and one QZ iteration requires approximately $13M^2$ operations]. Moreover, the operation count (29) is a rather generous upper bound. Consider, for example, the simple case where D is scalar (i.e. $m=p=1$) and invertible. Then ZEROS only performs the Householder transformation described in step 3. This single deflation requires $2M^2$ operations (with $M=n+1$) versus $16M^2$ operations for a corresponding deflation of the QZ algorithm.

It should be noted here that such operation counts reflect only part of the computation time used by an algorithm: the organizational burden can sometimes also be considerable. This is confirmed by the comparison of the first two methods reported in Laub and Moore (1978), for example: the QZ method has a smaller computing time although the method of Davison and Wang requires fewer operations.

3.2. *Numerical stability*

An important property of the proposed method is its backward stability. For the unitary transformations performed in REDUCE [see (9)], the following result can be proved (Wilkinson, 1965). In the presence of rounding errors, the right-hand side of (9) is exactly equal to

$$\left[\begin{array}{c|c}\bar{V}_j^* & 0\\\hline 0 & \bar{U}_j^*\end{array}\right]\left[\begin{array}{c|c}\lambda I_{\nu_{j-1}}-\bar{A}_{j-1} & \bar{B}_{j-1}\\\hline -\bar{C}_{j-1} & \bar{D}_{j-1}\end{array}\right]\left[\begin{array}{c|c}\bar{V}_j & 0\\\hline 0 & I_m\end{array}\right]$$

$$=\left[\begin{array}{c|c|c}\lambda I_{\nu_j}-A_j & * & B_j\\\hline -C_j & * & D_j\\\hline 0 & -S_j & 0\end{array}\right] \quad (30)$$

where $\bar{U}_j$ and $\bar{V}_j$ are still unitary. If ϵ is the machine precision of the computer, and a threshold EPS of the order of $\epsilon\cdot\|M_j\|_2$ is used, where $M_j=\begin{bmatrix}A_j & B_j\\C_j & D_j\end{bmatrix}$, then

$$\left\|\begin{bmatrix}\bar{A}_{j-1} & \bar{B}_{j-1}\\\bar{C}_{j-1} & \bar{D}_{j-1}\end{bmatrix}-\begin{bmatrix}A_{j-1} & B_{j-1}\\C_{j-1} & D_{j-1}\end{bmatrix}\right\|_2 \leq \Pi_j\cdot\epsilon\left\|\begin{matrix}A_{j-1} & B_{j-1}\\C_{j-1} & D_{j-1}\end{matrix}\right\|_2 \quad (31)$$

with Π_j a constant depending on the dimension of the matrices. Note that in (30) the coefficient

of λ is *not* perturbed because no computations are actually performed on it [the asterisks in (30) are not computed]. The errors performed in each step of REDUCE can be worked back to the original matrix $S(\lambda)$ without affecting their norms because unitary transformations do not change the $\|\cdot\|_2$ norm of a matrix. When doing this for the two calls of REDUCE in ZEROS, we obtain the equality

$$\bar{U}^* \begin{bmatrix} \lambda I - \bar{A} & \bar{B} \\ -\bar{C} & \bar{D} \end{bmatrix} \bar{V} = \begin{bmatrix} * & * & * & * \\ 0 & \lambda I - A_{rc} & B_{rc} & * \\ 0 & -C_{rc} & D_{rc} & * \\ 0 & 0 & 0 & * \end{bmatrix} \tag{32}$$

with $\bar{U}$, $\bar{V}$ unitary and

$$\left\| \begin{bmatrix} \bar{A} & \bar{B} \\ \bar{C} & \bar{D} \end{bmatrix} - \begin{bmatrix} A & B \\ C & D \end{bmatrix} \right\|_2 \leq \Pi_{\text{RED}} \cdot \epsilon \left\| \begin{matrix} A & B \\ C & D \end{matrix} \right\|_2 \tag{33}$$

where Π_{RED} is the sum of the Π_js in (31). Note that the threshold EPS can as well be chosen of the order of $\epsilon \cdot \|M\|_2$ with $M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$, for all steps without seriously affecting the bound (33). This is usually done in practice. A similar error analysis of step_3 in ZEROS and the QZ decomposition of $(\lambda B_f - A_f)$ yields

$$\left[\begin{array}{c|c} \bar{Q} & 0 \\ \hline 0 & I \end{array}\right] \left[\begin{array}{c|c} \lambda(I + E_a) - \bar{A}_{rc} & \lambda E_b + \bar{B}_{rc} \\ \hline -\bar{C}_{rc} & \bar{D}_{rc} \end{array}\right] \bar{W} \left[\begin{array}{c|c} \bar{Z} & 0 \\ \hline 0 & I \end{array}\right]$$

$$= \lambda \left[\begin{array}{ccc|c} \beta_1 & & * & \\ & \ddots & & * \\ 0 & & \beta_{n_f} & \\ \hline & 0 & & 0 \end{array}\right] - \left[\begin{array}{ccc|c} \alpha_1 & & * & \\ & \ddots & & * \\ 0 & & \alpha_{n_f} & \\ \hline & 0 & & D_f \end{array}\right] \tag{34}$$

with $\bar{Q}$, $\bar{Z}$, $\bar{W}$ unitary and

$$\left\| \begin{bmatrix} \bar{A}_{rc} & \bar{B}_{rc} \\ \bar{C}_{rc} & \bar{D}_{rc} \end{bmatrix} - \begin{bmatrix} A_{rc} & B_{rc} \\ C_{rc} & D_{rc} \end{bmatrix} \right\|_2 \leq \Pi_{rc} \cdot \epsilon \left\| \begin{bmatrix} A_{rc} & B_{rc} \\ C_{rc} & D_{rc} \end{bmatrix} \right\|_2 \tag{35}$$

$$\|E_a\|_2, \|E_b\|_2 \leq \Pi_e \cdot \epsilon \tag{36}$$

for some expressions Π_{rc} and Π_e.

Note that the coefficient of λ is perturbed, but that its rank is unaltered because of the special structure of the transformations in (32). Because of (36), there exists then a column transformation $(I + F)$ with $\|F\|_2 < 3\Pi_e \cdot \epsilon$ such that

$$\left[\begin{array}{c|c} \lambda(I + E_a) - \bar{A}_{rc} & \lambda E_b + \bar{B}_{rc} \\ \hline -\bar{C}_{rc} & \bar{D}_{rc} \end{array}\right] (I + F) = \left[\begin{array}{c|c} \lambda I - \tilde{A}_{rc} & \tilde{B}_{rc} \\ \hline -\tilde{C}_{rc} & \tilde{D}_{rc} \end{array}\right] \tag{37}$$

with

$$\left\| \begin{bmatrix} A_{rc} & B_{rc} \\ C_{rc} & D_{rc} \end{bmatrix} - \begin{bmatrix} \tilde{A}_{rc} & \tilde{B}_{rc} \\ \tilde{C}_{rc} & \tilde{D}_{rc} \end{bmatrix} \right\|_2 \leq (\Pi_{rc} + 3\Pi_e) \cdot \epsilon \left\| \begin{bmatrix} A_{rc} & B_{rc} \\ C_{rc} & D_{rc} \end{bmatrix} \right\|_2. \tag{38}$$

This error can again be worked back to the original matrix $S(\lambda)$. We finally obtain that the ratios $\lambda_i = (\alpha_i/\beta_i)$ are the *exact* zeros of a system $\{\lambda I - \tilde{A}, \tilde{B}, \tilde{C}, \tilde{D}\}$ such that

$$\left\| \begin{bmatrix} \tilde{A} & \tilde{B} \\ \tilde{C} & \tilde{D} \end{bmatrix} - \begin{bmatrix} A & B \\ C & D \end{bmatrix} \right\|_2 \leq \tilde{\Pi} \cdot \epsilon \left\| \begin{bmatrix} A & B \\ C & D \end{bmatrix} \right\|_2 \tag{39}$$

where

$$\tilde{\Pi} = \Pi_{\text{RED}} + \Pi_{rc} + 3\Pi_e. \tag{40}$$

Note that the expression $\tilde{\Pi}$ is a rather generous upperbound. One can estimate $\tilde{\Pi}$ experimentally (see next section) and it is fair to say that $\tilde{\Pi}$ is close to 1 for matrices of reasonable size ($n \leq 20$). For larger matrices, it is recommended to use doubled accuracy for the inner products in the Householder transformations in order to keep $\tilde{\Pi}$ close to one [see Wilkinson (1965), p. 152 for more details].

Additional features. The method described in Section 2 requires no assumption on $\{A, B, C, D\}$ and requires no special treatment for different cases as opposed to methods 1 and 2. It handles the case where the normal rank of the transfer function is smaller than $\min\{m, p\}$ and has no difficulty with high multiplicity zeros at infinity under small perturbations.

The 'degenerate' case has been pretty much neglected in the past because of its ill-posedness. Recently more attention has been paid to this problem and justifications have been given, from a numerical and physical point of view, for computing zeros of such systems (Wilkinson, 1978, 1979; Van Dooren, 1979, 1981). Another nice property of our approach is that the sorting of so-called extraneous zeros is avoided via the rank decisions of the Kronecker approach. It is known (Wilkinson, 1978, 1979; Van Dooren 1979, 1981) that such rank decisions with respect to the Kronecker canonical form can be quite delicate also, but they are only affected by the sensitivity of the *order* of the regular part $\lambda B_f - A_f$, or in other words by the number of zeros of the considered system. The first two methods, on the other hand, make decisions based on computed eigenvalues, which thus

may be affected by the sensitivity of the *zeros* themselves (badly conditioned zeros might thus be ruled out as extraneous zeros, while their number could be robustly fixed).

Finally, the generation of the random matrix K is replaced in our approach by a single parameter EPS. The user should put EPS equal to the noise level of his data, provided that it is larger than, say, $10 \cdot \epsilon \cdot \left\| \begin{bmatrix} A & B \\ C & D \end{bmatrix} \right\|_2$. Otherwise, EPS should be put equal to this lower bound (this could easily be taken care of by the program).

4. EXAMPLES

All the examples considered are real. The computations were carried out on the IBM 370/3033 at Stanford University. We used the FORTRAN H Extended Compiler, OPTIMIZE(2) and all computations were performed in double precision (REAL*8). The driver program RGG of EISPACK (Garbow and co-workers, 1977) was used to call the *QZ* algorithm and singular values were obtained using the routine DSVDC of LINPACK (Dongarra, and co-workers, 1979). The value of EPS used in the rank tests was chosen equal to $100\,\epsilon \cdot \left\| \begin{matrix} A & B \\ C & D \end{matrix} \right\|_2$ for each example (the value of ϵ here was $15 \cdot 16^{-14} \approx 2.08 \times 10^{-16}$). For each example, we also compute the ordered singular values $\sigma_1 \geq \sigma_2 \geq \ldots$ of the system matrix

$$S(\lambda_0) = \begin{bmatrix} \lambda_0 I - A & B \\ -C & D \end{bmatrix} \tag{41}$$

at each computed zero λ_0. We refer to the ratio

$$RBA = \frac{\sigma_{(n+\text{rank})}}{\sigma_1} \tag{42}$$

as the 'relative backward error'. Note that according to the backward error analysis, RBA is of the order of $\tilde{\Pi} \cdot \epsilon$. Indeed, λ_0 is the exact zero of the system $\{\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D}\}$, or

$$\sigma_{(n+\text{rank})} \left\{ \begin{bmatrix} \lambda_0 I - \tilde{A} & \tilde{B} \\ -\tilde{C} & \tilde{D} \end{bmatrix} \right\} = 0. \tag{43}$$

Hence, because of (39), the *actual* backward error

$$\eta = \left\| \begin{bmatrix} \tilde{A} & \tilde{B} \\ \tilde{C} & \tilde{D} \end{bmatrix} - \begin{bmatrix} A & B \\ C & D \end{bmatrix} \right\|_2$$

satisfies

$$\eta = \sigma_{(n+\text{rank})}\{S(\lambda_0)\} \approx \tilde{\Pi} \cdot \epsilon \left\| \begin{matrix} A & B \\ C & D \end{matrix} \right\|_2 \approx \tilde{\Pi} \cdot \epsilon \sigma_1\{S(\lambda_0)\}. \tag{44}$$

This allows us to estimate the value of $\tilde{\Pi}$. Note that an ϵ-small ***backward error does not*** imply that the zeros are computed up to EPS-accuracy. This also depends on the conditioning of each separate zero.

Indeed, if $\kappa(\lambda_0)$ denotes the conditioning of the zero λ_0 then the error performed when computing this zero, is at most $\eta \cdot \kappa(\lambda_0)$, where η is the norm of the actual backward error of the algorithm (Wilkinson, 1965). The importance of having a stable method is that η—i.e. the contribution of the algorithm to the error in the computed zero—is minimized (we have that $\eta <$ EPS). The value of RBA in the examples is always less than ϵ. Because of (44), we may say that $\tilde{\Pi} \cdot \epsilon \approx \epsilon$ and thus $\tilde{\Pi} \approx 1$. Hence, the backward error η is of the order of $\epsilon \cdot \left\| \begin{matrix} A & B \\ C & D \end{matrix} \right\|_2$, which is unavoidable on a computer with machine precision ϵ. The precision of the zero λ_0 can be estimated by performing a second run, whereby a random ϵ-perturbation is added to the system. This, in fact, is implicitly done in the two previous methods, but on the other hand, these methods may have troubles discerning sensitive eigenvalues [where $\kappa(\lambda_0)$ is very large] from extraneous ones. In each of the examples the value of rank, the normal rank of the transfer function , is also given. The computed value RBA is only explicitly given when it is larger than ϵ (this occurs only once!) since it can only be computed up to ϵ accuracy (Dongarra and co-workers, 1979). The exact digits in the computed zeros are underlined. These were obtained by comparison with extended precision results (see also Aplevich, 1979).

Example 1

This example is the sixteenth-order linearized model of the F100-PW-100 jet engine used as the theme problem for the International Forum on Alternatives for linear multivariable control (Sain and co-workers, 1978).

D has rank 4, and there are fifteen zeros. The following shows the computed zeros to sixteen significant digits.

Zeros	RBA $= \sigma_{21}/\sigma_1 \approx \tilde{\Pi} \cdot \epsilon$ (rank = 5)
−829.2490955651110	$< \epsilon$
789.8985828158399	$< \epsilon$
141.2294550203129	$< \epsilon$
−50.46757476394580	
± j1.031914160423297	$< \epsilon$
−49.63760103236723	$< \epsilon$
−13.76530730452916	
± j9.110214747547156	$< \epsilon$
−0.6659561616385485	$< \epsilon$
−6.710651036803525	$< \epsilon$
−2.003403155575229	$< \epsilon$
−23.13366516893961	$< \epsilon$
−20.55602749379905	
± j1.417353350011828	$< \epsilon$
−18.958550189406822	$< \epsilon$

Example 2

This is a ninth-order model of a boiler system (Axelby, Laub and Davison, 1978).

The zeros along with the corresponding relative backward errors, are as follows:

Zeros	RBA = $\sigma_{11}/\sigma_1 \approx \tilde{\Pi} \cdot \epsilon$ (rank = 2)
−26.39513728882773	$< \epsilon$
−2.957771985292086	
± j0.3352672071870191	$< \epsilon$
0.7486064441907556	$< \epsilon$
0.09403261020202233	$< \epsilon$
−0.009546070612163396	$< \epsilon$

$$A = \begin{bmatrix} -0.129 & 0.000 & 0.396 \times 10^{-1} & 0.250 \times 10^{-1} & 0.191 \times 10^{-1} \\ 0.329 \times 10^{-2} & 0.000 & -0.779 \times 10^{-4} & 0.122 \times 10^{-3} & -0.621 \\ 0.718 \times 10^{-1} & 0.000 & -0.100 & 0.887 \times 10^{-3} & -0.385 \times 10^{1} \\ 0.411 \times 10^{-1} & 0.000 & 0.000 & -0.822 \times 10^{-1} & 0.000 \\ 0.361 \times 10^{-3} & 0.000 & 0.350 \times 10^{-4} & 0.426 \times 10^{-4} & -0.743 \times 10^{-1} \end{bmatrix}$$

Notice that, despite the fact that our method is numerically stable, it gives less significant digits of accuracy than the other two methods. The problem here is the wide numberical range of data, which can be overcome by balancing the $\{A, B, C, D\}$ system. This is implicitly done in methods 1 and 2 via the EISPACK routines (Laub and Moore, 1978). For example if we use the transformation

$$T = \{10000, 1, 1, 10000, 1, 1, 1, 1, 1\}$$

then the system $\{TAT^{-1}, TB, CT^{-1}, D\}$ has much less sensitive zeros. Our method now gave the numerical results:

Zeros	RBA = $\sigma_{11}/\sigma_1 \approx \tilde{\Pi} \cdot \epsilon$ (rank = 2)
−26.39513729219998	$< \epsilon$
−2.957771983411052	
± j0.3352672040387147	$< \epsilon$
0.7486064352915261	$< \epsilon$
0.09403261463283083	$< \epsilon$
−0.009546070736639054	$< \epsilon$

Example 3

This is a sixth-order example from Davison and Wang (1974) with

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$$

Zeros	RBA = $\sigma_8/\sigma_1 \approx \tilde{\Pi} \cdot \epsilon$ (rank = 2)
−0.6823278038280190	$< \epsilon$
0.3411639019140096	
± j1.161541399997251	$4.07 \times 10^{-16} < 2\epsilon$
0.9999999999999997	$< \epsilon$

Example 4

The is a fifth-order model of a drum boiler (Bengtsson, 1973) with,

$$B = \begin{bmatrix} 0.000 & 0.139 \times 10^{-2} \\ 0.000 & 0.359 \times 10^{-4} \\ 0.000 & -0.989 \times 10^{-2} \\ 0.249 \times 10^{-4} & 0.000 \\ 0.000 & -0.534 \times 10^{-5} \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \quad D = 0.$$

The zeros and the relative backward errors are:

Zeros	RBA = $\sigma_7/\sigma_1 \approx \tilde{\Pi} \cdot \epsilon$ (rank = 2)
−0.368051203603595	$< \epsilon$
−0.06467751189941505	$< \epsilon$

Example 5

Consider the system with

$$S(\lambda) = \left[\begin{array}{ccc|c} \lambda & & & -1 \\ & \ddots & 0 & \\ & -1 & \ddots & 0 \\ 0 & & \ddots \; \lambda & 0 \\ \hline 0 & 0 & 1 & 0 \end{array}\right] \Bigg\} 16$$

corresponding to the transfer function $H(s) =$

$(1/s^{15})$. If we perturb the (16, 16) element by the order of machine precision ($\epsilon \approx 10^{-16}$), then the *QZ* algorithm will yield one zero at infinity and the other zeros lie on a circle with radius $\epsilon^{-1/n}$. Our algorithm has no such difficulty and will indicate that there are no finite zeros. The high gain method also has no problems with this example.

Example 6

Consider the case where

$$S(\lambda) = \left[\begin{array}{ccc:c} \lambda - 2 & 1 & 0 & 0 \\ 0 & \lambda & 0 & 0 \\ 1 & 0 & \lambda & 1 \\ \hdashline 0 & 1 & 0 & 0 \end{array}\right].$$

This pencil has right and left Kronecker indices equal to one. The normal rank of the transfer function is zero. The *QZ* algorithm (Laub and Moore, 1978) will indicate degeneracy (i.e. a (0/0) eigenvalue).

Theoretically one should not trust any of the other computed ratios as some of them could be arbitrary. But practically speaking only special perturbations could alter the true zero at 2. More about this can be found in Wilkinson (1978, 1979). The method of Davison and Wang (1978) also returns that this system is degenerate. Our algorithm will extract the singular part of $S(\lambda)$ and will yield a regular pencil containing the single zero at 2.

Example 7

Consider the system (rank = 0)

$$S(\lambda) = \left[\begin{array}{cc:c} \lambda & 0 & 0 \\ 0 & \lambda & 1 \\ \hdashline 1 & 0 & 0 \end{array}\right]$$

with left and right Kronecker indices equal to one and with vanishing transfer function (rank = 0). Both examples 6 and 7 are degenerate, but in contrast with the previous example, there are no zeros here. Methods 1 and 2 correctly indicate degeneracy, but they do not proceed further with the investigation of possible zeros, while our algorithm determines that $S(\lambda)$ has indeed no zeros.

Example 8

This is the model of an electrical network (Kaufman, 1973) with

$$A = \begin{bmatrix} -2 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 1 & -1 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$C = [0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0], \quad D = [0].$$

Zeros	RBA = $\sigma_7/\sigma_1 \approx \tilde{\Pi} \cdot \epsilon$ (rank = 1)
-0.9999999999999994	
$\pm j0.1821927265261 \times 10^{-7}$	$< \epsilon$

The system actually has two zeros at -1.0, but the error of the order of $\epsilon^{1/2}$ is to be expected because of the presence of a 2×2 Jordan block. The computed roots are indeed the exact roots of an ϵ-perturbed characteristics polynomial $\lambda^2 + (2 + \epsilon_1)\lambda + (1 + \epsilon_2)$ with negative discriminant $D = \epsilon_1 - \epsilon_2 + \epsilon_1^{2/4} \approx -\epsilon$. If instead we had $D \approx +\epsilon$, one would obtain two real roots close to $-1 + \epsilon^{1/2}$. This example illustrates that sensitive zeros could sometimes be confused with extraneous ones, when one merely looks at their invariance for differen runs.

Example 9

Consider the system with

$$S(\lambda) = \left[\begin{array}{ccc:c} \lambda & & 0 & -1 \\ & \ddots & & \vdots \\ -1 & & & 0.05 \\ & \ddots & & \vdots \\ 0 & & & 0 \\ & \ddots & & \vdots \\ & -1 & \lambda & 0 \\ \hdashline 0 & 0 & 1 & 0 \end{array}\right] \Bigg\} 15.$$

This system has a zero at 20. However, if we perturb the (16, 16) element by ϵ, the *QZ* algorithm will yield a zero at ∞ and the rest are located on a circle with radius $\epsilon^{-1/n}$. The eigenvalue at 20 is also absorbed in this 'cluster' and can no longer be discerned from the rest. Our algorithm will compute the zero at 20 with no difficulty. The normal rank of the transfer function is one.

Example 10

This example is taken from Kalman (1963)

$$A = \left[\begin{array}{cccc:cccc:cccc} 0 & 0 & 0 & -24 & & & & & & & & \\ 1 & 0 & 0 & -50 & & & & & & & & \\ 0 & 1 & 0 & -35 & & & & & & & & \\ 0 & 0 & 1 & -10 & & & & & & & & \\ \hdashline & & & & 0 & 0 & 0 & -30 & & & & \\ & & & & 1 & 0 & 0 & -61 & & & & \\ & & & & 0 & 1 & 0 & -41 & & & & \\ & & & & 0 & 0 & 1 & -11 & & & & \\ \hdashline & & & & & & & & 0 & 0 & -15 & \\ & & & & & & & & 1 & 0 & -23 & \\ & & & & & & & & 0 & 1 & -9 & \end{array}\right],$$

$$B = \begin{bmatrix} 135 & 18 & 14 & 20 \\ 117 & 42 & 25 & 33 \\ 33 & 30 & 13 & 15 \\ 3 & 6 & 2 & 2 \\ \hdashline 4 & 10 & 50 & 32 \\ 6 & 17 & 20 & 32 \\ 2 & 8 & 2 & 8 \\ \hdashline 0 & 1 & 0 & 0 \\ \hdashline 36 & 0 & 5 & 68 \\ 14 & -10 & 6 & 54 \\ 2 & -2 & 1 & 10 \end{bmatrix}$$

$$C = \left[\begin{array}{cccc:cccc:ccc} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right].$$

The system has input decoupling zeros at

Zeros	RBA $= \sigma_{11}/\sigma_1 \approx \tilde{\Pi} \cdot \epsilon$ (rank = 0)
-4.999999999999955	$< \epsilon$
-2.999999999999988	$< \epsilon$

and no output decoupling zeros.

Example 11

Consider the rectangular system (Kouvaritakis and MacFarlane, 1976)

$$A = \begin{bmatrix} -2 & -6 & 3 & -7 & 6 \\ 0 & -5 & 4 & -4 & 8 \\ 0 & 2 & 0 & 2 & -2 \\ 0 & 6 & -3 & 5 & -6 \\ 0 & -2 & 2 & -2 & 5 \end{bmatrix}, \quad B = \begin{bmatrix} -2 & 7 \\ -8 & -5 \\ -3 & 0 \\ 1 & 5 \\ -8 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & -1 & 2 & -1 & -1 \\ 1 & 1 & 1 & 0 & -1 \\ 0 & 3 & -2 & 3 & -1 \end{bmatrix}, \quad D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

The system has a left Kronecker index equal to one, no right Kronecker index and two zeros at infinity.

Zeros	RBA $= \sigma_7/\sigma_1 \approx \tilde{\Pi} \cdot \epsilon$ (rank = 2)
-3.000000000000000	$< \epsilon$
3.999999999999972	$< \epsilon$

5. CONCLUSIONS

We have presented a new algorithm for computing the zeros of a linear multivariable system. The algorithm deals effectively with the degenerate case as well and is proved to be backward stable. The method also yields the normal rank of the transfer function matrix, and has the potential of yielding more information about the structure of the given system. It has been successfully implemented on the computer.

Acknowledgements—We gratefully acknowledge the active interest of L. Silverman and several useful discussions with G. Franklin. We also thank the reviewers for providing several additional references and for their relevant criticisms. This research was supported by the National Science Foundation under grant ENG 78-1003, and by the U.S. Air Force under grant AFOSR-79-0094.

REFERENCES

Aplevich, J. D. (1979). Tableau methods for analysis and design of linear systems. *Automatica*, **15**, 419.

Axelby, G. S., A. J. Laub and E. J. Davison (1978). Further discussion on the calculation of transmission zeros. *Automatica*, **14**, 403.

Bengtsson, G. (1973). A theory for control of linear multivariable systems, Report 7341, Division of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Davison, E. J. (1976). The robust control of a servomechanism for linear time-invariant multivariable systems. *IEEE Trans Aut. Control* **AC-21**, 25.

Davison, E. J. and S. H. Wang (1974). Properties and calculation of transmission zeros of linear multivariable systems. *Automatica*, **10**, 643.

Davison, E. J. and S. H. Wang (1976). Remark on multiple transmission zeros of a system. *Automatica*, **12**, 195.

Davison, E. J. and S. H. Wang (1978). An algorithm for the calculation of transmission zeros of the system (*C*, *A*, *B*, *D*) using high gain output feedback. *IEEE Trans Aut. Control*, **AC-23**, 738.

Davison, E. J., W. Gesing and S. H. Wang (1978). An algorithm for obtaining the minimal realization of a linear time-invariant system and determining if a system is stabilizable-detectable. *IEEE Trans Aut. Control*, **AC-23**, 1048.

Desoer, C. A. and J. D. Schulman (1974). Zeros and poles of matrix transfer functions and their dynamical interpretation. *IEEE Trans Circuits & Syst.*, **CAS-21**, 3.

Dongarra, J. J., C. B. Moler, J. R. Bunch and G. W. Stewart (1979). *LINPACK User's Guide*. SIAM, Philadelphia.

Emami-Naeini, A. (1978). Multivariable poles and zeros. Internal Report, Information Systems Laboratory, Stanford University, Stanford, California.

Emami-Naeini, A. and P. Van Dooren (1980). Computation of zeros of linear multivariable systems. Report NA-80-03, Department of Computer Science, Stanford University, Stanford, California.

Emami-Naeini, A., P. Van Dooren and L. M. Silverman (1980). An efficient generalized eigenvalue method for computing zeros. *Proc. IEEE Conference. on Decision and Control*, 176.

Franics, B. A. and W. M. Wonham (1975). The role of transmission zeros in linear multivariable regulators. *Int. J. Control*, **22**, 657.

Franklin, G. F. (1978). A new formulation of the multivariable robust servomechanism problem. Contribution to *Workshop on Adaptive Control*, Champaign, Illinois.

Gantmacher, F. R. (1959). *The Theory of Matrices, I, II.* Chelsea.

Garbow, B. S., F. M. Boyle, J. J. Dongarra and C. B. Moler (1977). *Matrix eigensystem routines—EISPACK guide extension.* Lecture Notes in Computer Science No. 51, Springer, Berlin.

Golub, G., V. Klema and G. Stewart (1976). Rank degeneracy and least squares problems. Report STAN-CS-76-559, Computer Science Department, Stanford University, Stanford, California.

Jordan, D. and L. F. Godbout (1977). On the computation of the canonical pencil of a linear system. *IEEE Trans Aut. Control*, **AC-22**, 126.

Kalman, R. E. (1963). Mathematical description of linear dynamical systems, *SIAM J. Control*, **1**, 153.

Kaufman, I. (1973). On poles and zeros of linear systems. *IEEE Trans Circuit Theory*, **CT-20**, 93.

Kouvaritakis, B. and A. G. J. MacFarlane (1976). Geometric approach to the analysis and synthesis of system zeros, parts 1 and 2. *Int. J. Control*, **23**, 149.

Laub, A. J. and B. C. Moore (1978). Calculation of transmission zeros using *QZ* techniques. *Automatica*, **14**, 557.
MacFarlane, A. G. J. and N. Karcanias (1976). Poles and zeros of linear multivariable systems: a survey of the algebraic, geometric and complex-variable theory. *Int. J. Control*, **24**, 33.
McMillan, B. (1952). Introduction to formal realization theory, parts 1 and 2. *Bell System Tech. J.*, **31**, 217, 541.
Moler, C. B. and G. W. Stewart (1973). An algorithm for generalized matrix eigenvalue problems. *SIAM J. Num. Anal.*, **10**, 241.
Molinari, B. P. (1978). Structural invariants of linear multivariable systems. *Int. J. Control*, **28**, 493.
Moore, B. C. and L. Silverman (1974). A time domain characterization of the invariant factors of a system transfer function. *Proceedings Joint Automatic Control Conference.*, 186.
Moylan, P. J. (1977). Stable inversion of linear systems. *IEEE Trans Aut. Control*, **AC-22**, 74.
Porter, B. (1979). Computation of the zeros of linear multivariable systems. *Int. J. Systems Sci.* **10**, 1427.
Rosenbrock, H. H. (1970). *State-space and Multivariable Theory.* Nelson, London.
Sain, M. K., J. L. Peczkowski, J. L. Melga (Eds) (1978). *Alternatives for Linear Multivariable control.* National Engineering Consortium Inc., Chicago.
Silverman, L. M. (1976). Discrete Riccati equations: alternative algorithms, asymptotic properties, and system theoretic interpretations, in *Control and Dynamic Systems*, Vol. 12. Academic Press, New York.
Thompson, G. L. and R. L. Weil (1972). The roots of matrix pencils $AY = \lambda BY$: existence, calculations and relations to game theory. *Lin. Alg. & Appl.* **5**, 207.
Van Dooren, P. (1979). The computation of Kronecker's canonical form of a singular pencil. *Lin. Alg. & Appl.* **27**, 103.
Van Dooren, P. (1981). The generalized eigenstructure problem in linear system theory. *IEEE Trans Aut. Control.* **AC-26**, 111.
Van Dooren, P., A. Emami-Naeini and L. Silverman (1979). Stable extraction of the Kronecker structure of pencils. *Proceedings IEEE Conference on Decision and Control*, 521.
Verghese, G., P. Van Dooren and T. Kailath (1979). Properties of the system matrix of a generalized stage-space system. *Int. J. Control*, **30**, 235.
Wilkinson, J. H. (1965). *The Algebraic Eigenvalue Problem.* Oxford University Press, Oxford.
Wilkinson, J. H. (1978). Linear differential equations and Kronecker canonical form, in *Recent Advances in Numerical Analysis.* Academic Press, New York.
Wilkinson, J. H. (1979). Kronecker's canonical form and the *QZ* algorithm. *Lin. Alg. & Appl.*, **28**, 285.

APPENDIX: SUBROUTINE CODES FOR COMPUTING ZEROS

The basic subroutine codes are listed in FORTRAN for the convenience of the reader, but it must be emphasized that they have not been developed by a commercial software firm to be transportable, machine independent, PFORT verified, or essentially error free. Consequently, they must be used with some caution and risk by the reader. Note also that only subroutines are listed, and a main program is needed to call them and provide an interface with the user's computer and I/O devices. Also, as indicated in Section 4, other programs preferably from EISPACK and LINPACK, are required to call the QZ algorithm and to obtain singular values—if the numerical solutions to the examples given in Section 4 are to be obtained and compared, including the RBA as defined in equation (42).

With respect to the program listing, the authors have noted that they "regard the program as a guideline for implementation of the algorithm by software experts". We would be interested in receiving comments about the inclusion of the program code with this or any other paper, we will welcome any account of the experiences anyone has in attempting to use this algorithm.

Editor-in-Chief

```
      SUBROUTINE ZEROS(A,B,C,D,M,NMAX,N,PMAX,P,MAX,EPS,BF,AF,NU,
     *RANK,SUM,DUMMY)
C***  THIS ROUTINE EXTRACTS FROM THE SYSTEM MATRIX OF A STATE-SPACE
C***  SYSTEM {A(N,N),B(N,M),C(P,N),D(P,M)} A REGULAR PENCIL
C***  {λ.BF(NU,NU)-AF(NU,NU)} WHICH HAS THE NU INVARIANT ZEROS OF
C***  THE SYSTEM AS GENERALIZED EIGENVALUES. THE ROUTINE ZEROS
C***  REQUIRES THE SUBROUTINES REDUCE, HOUSH, PIVOT, TR1 AND TR2.
C***  THE PARAMETERS IN THE CALLING SEQUENCE ARE (STARRED INPUT
C***  PARAMETERS ARE ALTERED BY THE SUBROUTINE) :
C***    INPUT:
C***   *A,B,C,D    THE SYSTEM DESCRIPTOR MATRICES
C***    M,N,P      THE NUMBER OF INPUTS, STATES AND OUTPUTS
C***    PMAX,NMAX  THE FIRST DIMENSION OF C,D AND A,B RESPECTIVELY
C***    MAX        THE FIRST DIMENSION OF AF,BF
C***    EPS        THE ABSOLUTE TOLERANCE OF THE DATA(NOISE LEVEL),IT
C***               SHOULD BE LARGER THAN THE MACH. ACC.*NORM(A,B,C,D)
C***    OUTPUT:
C***    BF,AF      THE COEFFICIENT MATRICES OF THE REDUCED PENCIL
C***    NU         THE NUMBER OF (FINITE) INVARIANT ZEROS
C***    RANK       THE NORMAL RANK OF THE TRANSFER FUNCTION
C***    WORKING SPACE:
C***    SUM        A VECTOR OF DIMENSION AT LEAST MAX(M,P)
C***    DUMMY      A VECTOR OF DIMENSION AT LEAST MAX(M,N,P)
C***
      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL ZERO
      INTEGER P,PMAX,PP,RANK,RO,SIGMA
      DIMENSION A(NMAX,N),B(NMAX,M),C(PMAX,N),D(PMAX,M),AF(MAX,1),
     *BF(MAX,1),SUM(1),DUMMY(1)
      MM=M
      NN=N
      PP=P
C* CONSTRUCT THE COMPOUND MATRIX | B A | OF DIMENSION (N+P)X(M+N)
C*                            | D C |
      IF(MM.EQ.0) GO TO 15
      DO 10 I=1,NN
      DO 10 J=1,MM
```

```
      10      BF(I,J)=B(I,J)
      15 DO 20 I=1,NN
           DO 20 J=1,NN
      20      BF(I,J+MM)=A(I,J)
         IF(PP.EQ.0) GO TO 45
         IF(MM.EQ.0) GO TO 35
         DO 30 I=1,PP
           DO 30 J=1,MM
      30      BF(I+NN,J)=D(I,J)
      35 DO 40 I=1,PP
           DO 40 J=1,NN
      40      BF(I+NN,J+MM)=C(I,J)
C* REDUCE THIS SYSTEM TO ONE WITH THE SAME INVARIANT ZEROS AND WITH
C* D FULL ROW RANK MU (THE NORMAL RANK OF THE ORIGINAL SYSTEM).
C*
      45 RO=PP
         SIGMA=0
         CALL REDUCE(BF,MAX,MM,NN,PP,EPS,RO,SIGMA,MU,NU,SUM,DUMMY)
         RANK=MU
         IF(NU.EQ.0) RETURN
C* PERTRANSPOSE THE SYSTEM.
         NUMU=NU+MU
         MNU=MM+NU
         NUMU1=NUMU+1
         MNU1=MNU+1
         DO 50 I=1,NUMU
           DO 50 J=1,MNU
      50      AF(MNU1-J,NUMU1-I)=BF(I,J)
         IF (MU.EQ.MM) GO TO 55
         PP=MM
         NN=NU
         MM=MU
C* REDUCE THE SYSTEM TO ONE WITH THE SAME INVARIANT ZEROS AND WITH
C* D SQUARE INVERTIBLE.
C*
         RO=PP-MM
         SIGMA=MM
         CALL REDUCE(AF,MAX,MM,NN,PP,EPS,RO,SIGMA,MU,NU,SUM,DUMMY)
         IF(NU.EQ.0) RETURN
C* PERFORM A UNITARY TRANSFORMATION ON THE COLUMNS OF |λI-A  B | IN
C*                            |λBF-AF X|                  | -C  D |
C* ORDER TO REDUCE IT TO |  0    Y| WITH Y AND BF SQUARE INVERTIBLE
C*
         MNU=MM+NU
      55 DO 70 I=1,NU
           DO 60 J=1,MNU
      60      BF(I,J)=0.D0
      70   BF(I,I+MM)=1.D0
         IF(RANK.EQ.0) RETURN
         NU1=NU+1
         I1=NU+MU
         J1=MNU+1
         I0=MM
         DO 90 I=1,MM
           I0=I0-1
           DO 80 J=1,NU1
      80      DUMMY(J)=AF(I1,I0+J)
         CALL HOUSH(DUMMY,NU1,NU1,EPS,ZERO,S)
         CALL TR2(AF,MAX,DUMMY,S,1,I1,I0,NU1)
         CALL TR2(BF,MAX,DUMMY,S,1,NU,I0,NU1)
      90 I1=I1-1
         RETURN
         END
         SUBROUTINE REDUCE(ABCD,MDIMA,M,N,P,EPS,RO,SIGMA,MU,NU,SUM,
        *DUMMY)
C***  THIS ROUTINE EXTRACTS FROM THE (N+P)X(M+N) SYSTEM [ B A ]
C***                                                [ B'A']      [ D C ]
C***  A (NU+MU)X(M+NU) 'REDUCED' SYSTEM [ D'C'] HAVING THE SAME
C***  TRANSMISSION ZEROS BUT WITH D' OF FULL ROW RANK. THE SYSTEM
C***  {A',B',C',D'} OVERWRITES THE OLD SYSTEM. EPS IS THE NOISE
C***  LEVEL. SUM(MAX(P,M)) AND DUMMY(MAX(P,N)) ARE WORKING ARRAYS.
C***
         IMPLICIT REAL*8 (A-H,O-Z)
         INTEGER TAU,P,RO,RO1,SIGMA
         LOGICAL ZERO
         DIMENSION ABCD(MDIMA,1),DUMMY(1),SUM(1)
         MU=P
         NU=N
      10 IF(MU.EQ.0) RETURN
```

```
      RO1=RO
      MNU=M+NU
      NUMU=NU+MU
      IF(M.EQ.0) GO TO 120
      RO1=RO1+1
      IROW=NU
      IF(SIGMA.LE.1) GO TO 40
C***  COMPRESS ROWS OF D. FIRST EXPLOIT TRIANGULAR SHAPE ***
      M1=SIGMA-1
      DO 30 ICOL=1,M1
        DO 20 J=1,RO1
   20     DUMMY(J)=ABCD(IROW+J,ICOL)
        CALL HOUSH(DUMMY,RO1,1,EPS,ZERO,S)
        CALL TR1(ABCD,MDIMA,DUMMY,S,IROW,RO1,ICOL,MNU)
   30 IROW=IROW+1
C***  CONTINUE WITH HOUSEHOLDER WITH PIVOTING ***
   40 IF(SIGMA.NE.0) GO TO 45
      SIGMA=1
      RO1=RO1-1
   45 IF(SIGMA.EQ.M) GO TO 60
      DO 55 ICOL=SIGMA,M
        DUM=0.D0
        DO 50 J=1,RO1
   50     DUM=DUM+ABCD(IROW+J,ICOL)*ABCD(IROW+J,ICOL)
   55   SUM(ICOL)=DUM
   60   DO 100 ICOL=SIGMA,M
C***  PIVOT IF NECESSARY  ***
        IF(ICOL.EQ.M) GO TO 80
        CALL PIVOT (SUM,DUM,IBAR,ICOL,M)
        IF(IBAR.EQ.ICOL) GO TO 80
        SUM(IBAR)=SUM(ICOL)
        SUM(ICOL)=DUM
      DO 70 I=1,NUMU
         DUM=ABCD(I,ICOL)
         ABCD(I,ICOL)=ABCD(I,IBAR)
   70    ABCD(I,IBAR)=DUM
C***  PERFORM HOUSEHOLDER TRANSFORMATION ***
   80   DO 90 I=1,RO1
   90     DUMMY(I)=ABCD(IROW+I,ICOL)
        CALL HOUSH(DUMMY,RO1,1,EPS,ZERO,S)
        IF(ZERO) GO TO 120
        IF(RO1.EQ.1)  RETURN
        CALL TR1(ABCD,MDIMA,DUMMY,S,IROW,RO1,ICOL,MNU)
        IROW=IROW+1
        RO1=RO1-1
        DO 100 J=ICOL,M
  100     SUM(J)=SUM(J)-ABCD(IROW,J)*ABCD(IROW,J)
  120 TAU=RO1
      SIGMA=MU-TAU
C***  COMPRESS THE COLUMNS OF C ***
      IF(NU.LE.0) GO TO 220
      I1=NU+SIGMA
      MM1=M+1
      N1=NU
      IF(TAU.EQ.1) GO TO 140
      DO 135 I=1,TAU
        DUM=0.D0
        DO 130 J=MM1,MNU
  130     DUM=DUM+ABCD(I1+I,J)*ABCD(I1+I,J)
  135   SUM(I)=DUM
  140 DO 200 RO1=1,TAU
      RO=RO1-1
        I=TAU-RO
        I2=I+I1
C***  PIVOT IF NECESSARY  ***
      IF(I.EQ.1) GO TO 160
      CALL PIVOT(SUM,DUM,IBAR,1,I)
      IF(IBAR.EQ.I) GO TO 160
      SUM(IBAR)=SUM(I)
      SUM(I)=DUM
      DO 150 J=MM1,MNU
        DUM=ABCD(I2,J)
        ABCD(I2,J)=ABCD(IBAR+I1,J)
  150   ABCD(IBAR+I1,J)= DUM
C***  PERFORM HOUSEHOLDER TRANSFORMATION ***
  160 DO 170 J=1,N1
  170   DUMMY(J)=ABCD(I2,M+J)
      CALL HOUSH(DUMMY,N1,N1,EPS,ZERO,S)
      IF(ZERO) GO TO 210
      IF(N1.EQ.1) GO TO 200
```

```
      CALL TR2(ABCD,MDIMA,DUMMY,S,1,I2,M,N1)
      MN1=M+N1
      CALL TR1(ABCD,MDIMA,DUMMY,S,0,N1,1,MN1)
      DO 190 J=1,I
  190   SUM(J)=SUM(J)-ABCD(I1+J,MN1)*ABCD(I1+J,MN1)
      MNU=MNU-1
  200 N1=N1-1
      RO=TAU
  210 NU=NU-RO
      MU=SIGMA+RO
      IF (RO.EQ.0) RETURN
      GO TO 10
  220 MU=SIGMA
      NU=0
      RETURN
      END
      SUBROUTINE PIVOT(NORM,MAX,IBAR,I1,I2)
C***  THIS SUBROUTINE COMPUTES THE MAXIMAL ELEMENT (MAX) OF THE
C***  VECTOR NORM(I1,...,I2) AND ITS LOCATION IBAR
      REAL*8 NORM(1),MAX
      IBAR=I1
      MAX=NORM(1)
      I11=I1+1
      IF(I11.GT.I2) RETURN
      DO 10 I=I11,I2
        IF(MAX.GE.NORM(I)) GO TO 10
        MAX=NORM(I)
        IBAR=I
   10   CONTINUE
      RETURN
      END
      SUBROUTINE HOUSH(DUMMY,K,J,EPS,ZERO,S)
C***  THIS ROUTINE CONSTRUCTS A HOUSEHOLDER TRANSFORMATION H=I-S.UU'
C***  THAT 'MIRRORS' A VECTOR DUMMY(1,..,K) TO THE JTH UNIT VECTOR
C***    IF NORM(DUMMY)<EPS, ZERO IS PUT EQUAL TO .TRUE.
C***  UPON RETURN U IS STORED IN DUMMY
      REAL*8 DUMMY(K),S,ALFA,DUM1,EPS
      LOGICAL ZERO
      ZERO=.TRUE.
      S=0.D0
      DO 10 I=1,K
   10   S=S+DUMMY(I)*DUMMY(I)
      ALFA=DSQRT(S)
      IF (ALFA.LE.EPS) RETURN
      ZERO=.FALSE.
      DUM1=DUMMY(J)
      IF(DUM1.GT.0.D0) ALFA=-ALFA
      DUMMY(J)=DUM1-ALFA
      S=1.D0/(S-ALFA*DUM1)
      RETURN
      END
      SUBROUTINE TR1(A,MDIMA,U,S,I1,I2,J1,J2)
C*** THIS ROUTINE PERFORMS THE HOUSEHOLDER TRANSFORMATION H=I-S.UU'
C*** ON THE ROWS I1+1 TO I1+I2 OF A, THIS FROM COLUMNS J1 TO J2.
C***
      REAL*8 A(MDIMA,1),U(I2),S,INPROD,Y
      DO 20 J=J1,J2
        INPROD=0.D0
        DO 10 I=1,I2
   10     INPROD=INPROD+U(I)*A(I1+I,J)
        Y=INPROD*S
        DO 20 I=1,I2
   20     A(I1+I,J)=A(I1+I,J)-U(I)*Y
      RETURN
      END
      SUBROUTINE TR2(A,MDIMA,U,S,I1,I2,J1,J2)
C***  THIS ROUTINE PERFORMS THE HOUSEHOLDER TRANSFORMATION H=I-S.UU'
C***  ON THE COLUMNS J1+1 TO J1+J2 OF A, THIS FROM ROWS I1 TO I2.
C***
      REAL*8 A(MDIMA,1),U(J2),S,INPROD,Y
      DO 20 I=I1,I2
      INPROD=0.D0
      DO 10 J=1,J2
   10   INPROD=INPROD+U(J)*A(I,J1+J)
      Y=INPROD*S
      DO 20 J=1,J2
   20   A(I,J1+J)=A(I,J1+J)-U(J)*Y
      RETURN
      END
```