NUMERICAL ANALYSIS PROJECT
MANUSCRIPT NA-80-03

JULY 1980

COMPUTATION OF ZEROS OF LINEAR MULTIVARIABLE SYSTEMS

BY

A. EMAMI-NAEINI
P. VAN DOOREN

NUMERICAL ANALYSIS PROJECT
COMPUTER SCIENCE DEPARTMENT
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305

COMPUTATION OF ZEROS OF LINEAR MULTIVARIABLE SYSTEMS

A. Emami-Naeini* and P. Van Dooren**

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 $\{\lambda I - A,B,C,D\}$. In this report we discuss the numerical properties of a new algorithm and compare it with some earlier techniques of computing zeros. The new approach is shown to handle both nonsquare and/or degenerate systems without difficulties whereas earlier methods would either fail or would require special treatment for these cases. The method is also shown to be backward stable in a rigorous sense. Several numerical examples are given in order to compare speed and accuracy of the algorithm with its nearest competitors.

* Information Systems Laboratory, Stanford University.

** Information Systems Laboratory and Department of Computer Science, Stanford University

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.

I. 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, e.g., the study of regulation, robust servomechanism design, and decoupling [1]-[5].

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 Smith-zeros [6] of the system matrix,

$$S(\lambda) = \begin{bmatrix} XI-A & B \\ -C & D \end{bmatrix} \begin{matrix} \}n \\ \}p \end{matrix} \qquad \begin{matrix} \\ \\ n \quad m \end{matrix} \tag{2}$$

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 [7] for an elaborate

discussion). In the sequel we make no special distinction anymore between these four types of zeros since they are all the zeros of a special type of system matrix (2).

In this report we give a 'fast' implementation of a method that was recently designed to tackle the computation of the zeros of an arbitrary state-space system (1). We also compare our algorithm with those of Davison and Wang [1],[8] and Laub and Moore [9], two methods with 'controlled numerical behavior.' We first briefly review the methods.

The first technique, due to Davison and Wang [1], uses the invariance property of zeros under high gain output feedback to determine their locations. Let the full rank output feedback $u = \rho K y$ be applied to the system (assume m = p, i.e., square system). Then the zeros as defined in [8] are the finite eigenvalues of $[A-B(D - \frac{1}{\rho} K^{-1})^{-1}C]$. Equivalently, the zeros are the generalized eigenvalues [9] of the system matrix,

$$S_\rho(\lambda) = \begin{bmatrix} XI-A & B \\ -C & D - \frac{1}{\rho} K^{-1} \end{bmatrix} \qquad (3)$$

where $\rho = \gamma \frac{\|A\|}{\|BC\|}$ and γ is some 'large' number dependent on the accuracy of the computer. Simple examples can be constructed (see e.g. [9]) to show that the large gain ρK introduces considerable loss of accuracy and forces one to use up to quadruple precision. Since the problem has to be solved for several values of gain ρK, it makes the algorithm rather inefficient and at times the distinction between the

true zeros and the 'extraneous' ones may be difficult [1],[9]. In addition,one has to come up with the random output gain matrix K for each problem. In order to handle nonsquare systems, the associated system matrix has to be augmented with randomly generated rows or columns to make $S_\rho(\lambda)$ square. This has to be solved for two different random augmentations in order to determine the zeros.

The second method of computing zeros is discussed by Laub and Moore [9]. The QZ algorithm is used to determine the generalized eigenvalues of the square system matrix,

$$s(X) = \begin{bmatrix} XI-A & B \\ -C & D \end{bmatrix} \qquad (4)$$

The finite generalized eigenvalues of (4) are the zeros of the system. For the nonsquare case, similar techniques to those of Davison and Wang [1] need to be used. Random rows and columns are added to $S(\lambda)$ in order to obtain a square pencil which is then processed by the QZ algorithm. This method should be preferred over the previous one because of the numerical stability of the QZ algorithm.

The above methods may both encounter some difficulties in distinguishing the finite zeros from the infinite zeros (when the latter have a high multiplicity) and, more importantly, $S(\lambda)$ has normal rank $r < \min(p,m)$ [6].

The third technique of computing zeros is based on principles discussed in [10]-[13]. The system matrix (4) can always be transformed by unitary transformations P and Q to the generalized (upper) Schur form,

$$P\,S(A)\,Q = \begin{bmatrix} A_r-\lambda B_r & * & * & * \\ 0 & A_f-\lambda B_f & * & * \\ 0 & 0 & A_i-\lambda B_i & * \\ 0 & 0 & 0 & A_\ell-\lambda B_\ell \end{bmatrix} \tag{5}$$

when

(i) $(A_r-\lambda B_r)$ and $(A_\ell-\lambda B_\ell)$ are nonsquare pencils with no zeros and revealing the right and left minimal indices of $S(\lambda)$.

(ii) $(A_i-\lambda B_i)$ is a regular pencil with no finite zeros and revealing the infinite zeros of $S(\lambda)$.

(iii) $(A_f-\lambda B_f)$ is a regular pencil whose generalized eigenvalues are the finite zeros of S(A).

After this preliminary reduction, the QZ algorithm is applied to the 'finite structure' pencil $(A_f-\lambda B_f)$. 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 $\{\lambda I-\tilde{A},\tilde{B},\tilde{C},\tilde{D}\}$ where $\begin{bmatrix} A & B \\ C & D \end{bmatrix}$ is ϵ-close to $\begin{bmatrix} \tilde{A} & \tilde{B} \\ \tilde{C} & \tilde{D} \end{bmatrix}$ and ϵ is the machine precision of the computer. This does not hold for the method in [1]. While the left and right minimal indices and the multiple infinite zeros [14] are also determined by the decomposition (5), they are exactly the cause of the numerical problems in the two previous methods. This is illustrated by some simple numerical examples later.

This third method is somewhat related to other approaches that can be found in the literature [15]-[19]. 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 report is as follows. In section II the new algorithm for computing zeros is presented and compared with previous methods. In section III we discuss the properties of the new algorithm. In section IV several numerical examples are presented. Some concluding remarks appear in section V. The appendices include the data used in some of the numerical examples as well as a listing of the implementation of the algorithm in FORTRAN.

II. 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.

II.1 Reduction Method:

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

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

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

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

with no restrictions 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 Kronecker canonical form of the pencil $S(\lambda)$ [10]-[13] 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

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

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 11.2. The algorithm is stated in an ALOGOL 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; ν_0=n; δ_0:=0; μ_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};

$$\begin{matrix} \sigma_j \{ \\ \tau_j \{ \end{matrix} \left[\begin{array}{c|c} \overline{C}_{j-1} & \overline{D}_{j-1} \\ \hline \tilde{C}_{j-1} & 0 \end{array}\right] := U_j^* \left[\begin{array}{c|c} C_{j-1} & D_{j-1} \end{array}\right] ; \quad \underline{\text{if}} \;\; \tau_j = 0 \; \underline{\text{then go to}} \; \text{exit\_1};$$

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

$$\underbrace{[0}_{\nu_j} \; \underbrace{S_j]}_{\rho_j} := \tilde{C}_{j-1} V_j ; \quad \underline{\text{if}} \; \rho_j = 0 \; \underline{\text{then go to}} \; \text{exit\_1};$$

if $\nu_j=0$ then go to exit_2;

comment update;

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

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

$\underbrace{}_{\nu_j}$ $\underbrace{}_{\rho_j}$ $\underbrace{}_{m}$

j:=j+1; go to step_j;

exit_1 : comment $\{\lambda I_n - A, B, C, D\}$ and $\{\lambda I_{n_r} - A_r, B_r, C_r, D_r\}$ have the same zeros;

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

stop;

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

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

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

Proof: We prove the result by induction.

Step j of the above algorithm reduces the system matrix of $\{\lambda I_{\nu_{j-1}} - A_{j-1}, B_{j-1}, C_{j-1}, D_{j-1}\}$ to the form:

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

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,

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

and the systems $\{\lambda I - A_k, B_k, C_k, D_k\}$ for $k = j-1, j$ have thus the same Smith 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 $\{\lambda I-A,B,C,D\}$. 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 go to exit;

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 go to exit;

step 3 : comment compress the columns of $[C_{rc}\ D_{rc}]$ to $[0\ D_f]$ and apply the transformation to the system matrix;

n_f:=n_{rc}; if rank = 0 then go to exit;

$$\left[\begin{array}{c|c} A_f & * \\ \hline 0 & D_f \end{array}\right] := \left[\begin{array}{c|c} A_{rc} & B_{rc} \\ \hline D_{rc} & C_{rc} \end{array}\right] W; \quad \left[\begin{array}{c|c} B_f & * \\ \hline 0 & 0 \end{array}\right] := \left[\begin{array}{c|cc} 1 & 1 & 0 \\ \hline 0 & I & 0 \end{array}\right] W;$$

exit : stop;

Theorem 2: The(finite) zeros of the system $\{\lambda I-A,B,C,D\}$ 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

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

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

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

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 $S_{rc}(\lambda)$ to,

$$\begin{matrix} nrc \\ rank \end{matrix} \left\{\left[\begin{array}{c|c} XI_{n_{rc}} - A_{rc} & B_{rc} \\ \hline -C_{rc} & D_{rc} \end{array}\right]\right. \qquad W = \left[\begin{array}{c|c} \lambda B_f - A_f & * \\ \hline 0 & D_f \end{array}\right] \begin{matrix} \}n_f \\ \}rank \end{matrix} \tag{13}$$

$$\begin{matrix} \underbrace{\phantom{XI_{n_{rc}}}}_{n_{rc}} & \underbrace{\phantom{B_{rc}}}_{rank} \end{matrix} \qquad \begin{matrix} \underbrace{}_{n_f} & \underbrace{}_{rank} \end{matrix}$$

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 $S(x)$. ■

The zeros are now computed by the QZ method [20], which decomposes $(\lambda B_f - A_f)$ into,

$$Q(\lambda B_f - A_f)Z = \lambda \begin{bmatrix} \beta_{11} & & * \\ & \ddots & \\ 0 & & \beta_{n_f n_f} \end{bmatrix} - \begin{bmatrix} \alpha_{11} & & * \\ & \ddots & \\ 0 & & \alpha_{n_f n_f} \end{bmatrix} \qquad (14)$$

where Q and Z are unitary.

The ratios $\lambda_i = \frac{a_{ii}}{\beta_{ii}}$ $i = 1,\ldots,n_f$ 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 [12]. In case $S(\lambda)$ was minimal, these are also the infinite transmission zeros and left and right minimal indices of the transfer function $R(\lambda) = C(\lambda I - A)^{-1} B + D$ [14]. Note also 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 $\{\lambda I-A,B,C,D\}$ 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 <u>block</u> triangular form [20]

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

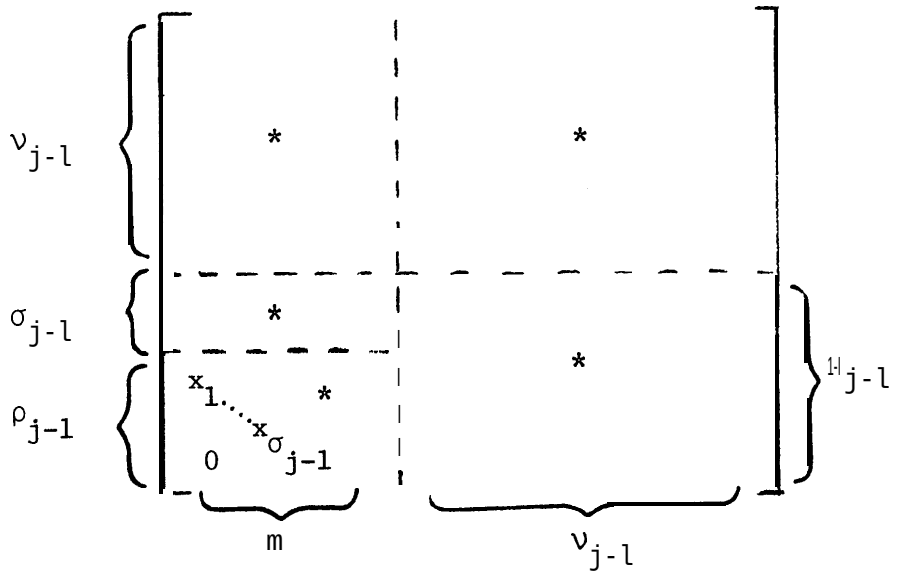
where the $(\lambda B_{ii} - A_{ii})$ blocks are 1×1 or 2×2. The generalized eigenvalues of the 2×2 blocks are complex conjugate pairs and those of the 1×1 blocks are real.

II.2 <u>Details of Implementation</u>

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,

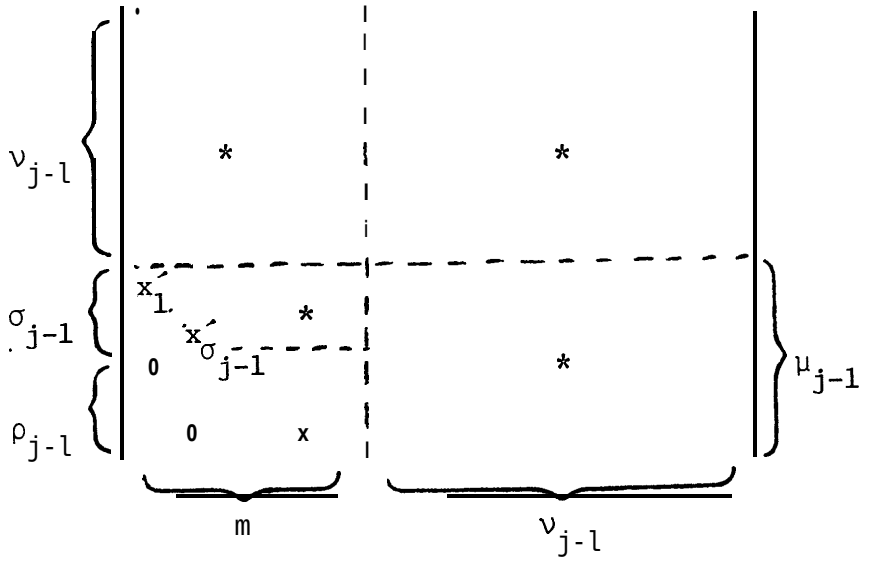
$$\left[\begin{array}{c|c} B_{j-1} & A_{j-1} \\ \hline C_{j-1} & D_{j-1} \end{array}\right] \qquad (16)$$

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



(17)

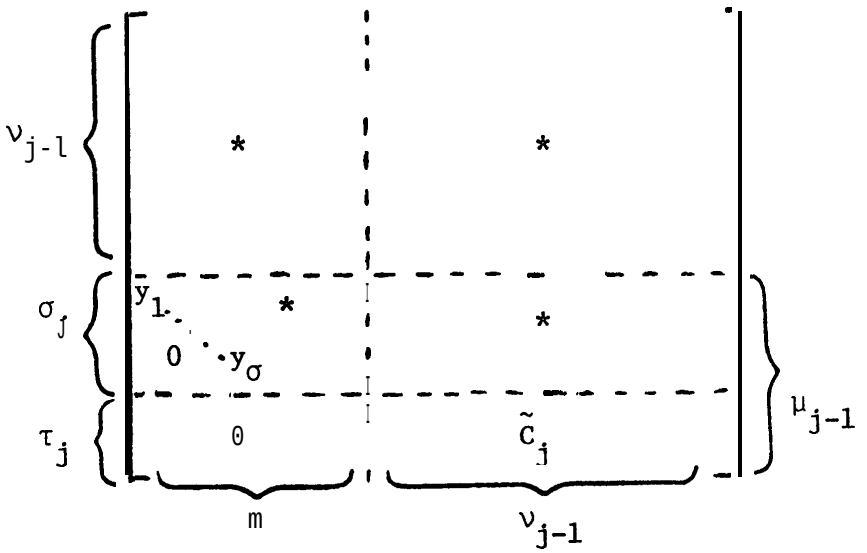
and the x_i's are--nonzero (when j = 1, we have ν_0=n; μ_0 = p; ρ_0 = p; σ_0 = 0). Step j first performs an output transformation $\overset{*}{U}_j$ to compress D_{j-1} to full row rank. Therefore we first use σ_{j-1} Householder transformations without pivoting (since x_i's are nonzero) to reduce (17) to,



(18)

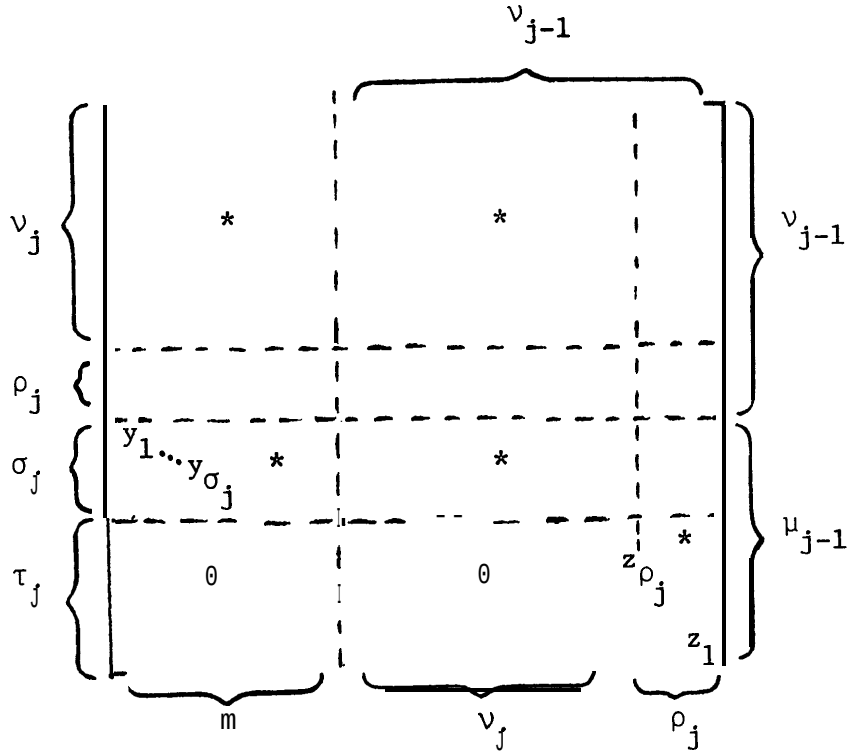
where again the x'_1's are nonzero. We then continue with Householder reduction with column pivoting to reduce X to trapezoidal form, yielding

finally a row compression of D_{j-1}



(19)

where the y_i's 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 $\tilde{C}_j$.



(20)

where z_i's are also nonzero. The last ρ_j columns and τ_j rows can then be discarded giving (17) again for j updated by one. This process is continued until no such 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_{r_c} by reducing a matrix of the form,

$$\left[\begin{array}{c|c} B_{rc} & A_{rc} \\ \hline D_{rc} & C_{rc} \end{array}\right] = \left[\begin{array}{c|c} \begin{matrix} * & \\ * & * \\ 0 & * \end{matrix} & \begin{matrix} * \\ * \\ * \end{matrix} \end{array}\right] \qquad (21)$$

to the form

$$\left[\begin{array}{c|c} A_f & * \\ \hline 0 \quad I & D_f \end{array}\right] = \left[\begin{array}{c|c} \begin{matrix} * & \\ 0 & * \end{matrix} & \begin{matrix} * \\ * \\ * \end{matrix} \\ \hline \begin{matrix} 0 & * \end{matrix} & * \end{array}\right] \qquad (22)$$

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 $[I_{n_{rc}} \quad 0]$.

III. PROPERTIES OF THE ALGORITHM AND COMPARISON

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

1. Speed

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. Using this, and the assumption that $m \leq p$, we obtain 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 $\rho_{j-1}(\nu_{j-1}+m) \leq \rho_{j-1}(n+m)$ (see (17),(18)). For this step we thus have less than

$$2\rho_{j-1}(n + m)m \text{ operations} \tag{23}$$

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

$$2\rho_j(p + m + 2n)n \text{ operations} \tag{24}$$

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

$$2(p + A)(n + m)m + 2\Delta(p + m + 2n)n \tag{25}$$

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

- for step_1 ; less than:

$$2\{(p+\Delta_1)(n+m)m + A_1(p+m+2n)n\} \text{ operations} \qquad (26)$$

with $\Delta_1 = n - n_r$.

- for step_2 ; less than:

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

with $\Delta_2 = n_r - n_{rc}$; $p_r \leq m$.

- for step_3 ; less than:

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

with $m_{rc} \leq m$.

This last step indeed requires m_{rc} Householder transformations working on matrices smaller than $(n_{rc}+m_{rc})(n_{rc}+1)$. 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 $(\lambda B_f - A_f)$ in less than

$$2\Delta(n+m)m + 2\Delta(m+p+2n)n + 4p(n+m)m$$

$$+ 2(n+m)(n+1)m \leq \Delta\{4M^2+2mM\} + 2mM^2 + 4pmM$$

$$\leq 6(\Delta+p)M^2 \text{ operations.} \qquad (29)$$

This operation count is a rather generous upper bound. Notice that $\Delta + p = M - n_{rc}$ is the total reduction of the dimension of S(X) to the pencil $(\lambda B_f - A_f)$. ZEROS thus requires less than $6M^2$ operations per deflation, while the QZ method used on S(X) directly, would require approximately $25M^2$ operations per deflation [9]. Furthermore, the operation count of the QZ method compared favorably to Davison and Wang's method [9].

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 [21]. In the presence of rounding errors, the right hand side of (9) is exactly equal to

$$\begin{bmatrix} \bar{V}_j^* & 0 \\ 0 & \bar{U}_j^* \end{bmatrix} \begin{bmatrix} \lambda I_{\nu_{j-1}} - \bar{A}_{j-1} & \bar{B}_{j-1} \\ -C_{j-1} & I - D_{j-1} \end{bmatrix} \begin{bmatrix} \bar{V}_j & 0 \\ 0 & I_m \end{bmatrix} = \begin{bmatrix} \lambda I_{\nu_j} - A_j & I & * & B_j \\ & & & \\ -c_j & & * & D_j \\ 0 & & -S_j & 0 \end{bmatrix} \tag{30}$$

where $\bar{U}_j$ and $\bar{V}_j$ are still unitary and if ϵ is the machine precision of the computer, then

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

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 *'s in (30) are not computed). The errors performed in each step of REDUCE can be worked back to the original matrix S(X) 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,

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

where Π_{RED} is the sum of the Π_j's in (31). A similar error analysis of step 3 in ZEROS and the QZ decomposition of $(\lambda B_f - A_f)$ yields,

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

$$= \lambda \left[\begin{array}{c|c} \begin{matrix} \beta_{11} & * \\ & \ddots \\ 0 & \beta_{n_f n_f} \end{matrix} & * \\ \hline 0 & 0 \end{array}\right] - \left[\begin{array}{c|c} \begin{matrix} \alpha_{11} & * \\ & \ddots \\ 0 & \alpha_{n_f n_f} \end{matrix} & * \\ \hline 0 & D_f \end{array}\right] \tag{34}$$

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

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

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

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 $\|F\|_2 < 3\ \Pi_e\ \epsilon$ 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

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

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

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

where

$$\text{ii} = \Pi_{RED} + \Pi_{rc} + 3\,\Pi_e \tag{40}$$

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 fl_2 computations [21] in the Householder transformations in order to keep ii close to one.

<u>Additional Features</u>:

The method described in section II requires no assumption on (A,B,C,D) and requires no special treatment for different cases as opposed to [1] and [9]. It handles the case where $r < \min\{m,p\}$ very effectively

and has no difficulty with high multiplicity zeros at infinity under small perturbations. The examples in section IV illustrate these advantages over the other two methods.

Our method has no problems in the presence of a Kronecker structure (left and right null space when $r < \min\{m,p\}$ and infinite zeros) because it separates a pencil $(\lambda B_f - A_f)$ with only a finite structure from the original system matrix $S(\lambda)$. This allows us to determine zeros of ungenerical system matrices, **e.g.**, when $r < \min\{p,m\}$ while the other two methods cannot handle this case properly.

The 'degenerate' case has been pretty much neglected in the past because of its ill-posedness. Recently more attention has been paid to the degenerate problem and justifications have been given, from a numerical and physical point of view, for computing zero of degenerate systems (see [13],[22]).

Another nice property of our approach is that no problem dependent parameter and/or matrix is needed as in method [1]. Moreover, the method is direct and does not require two runs and/or a (delicate) sorting of the 'extraneous' zeros in the nonsquare case.

IV. 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(Z) and all computations were performed in double precision (REAL*8). The driver program RGG of EISPACK [24] was used to call the QZ algorithm and singular values were obtained using the routine DSVDC of LINPACK [25]. For each example, we also compute the singular values σ_i's of the system matrix,

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

at each computed zero λ_o. We refer to the ratio,

$$RBA = \frac{\sigma_{(n+rank)}}{\text{"1}} \tag{42}$$

as the 'relative backward error.' Note that according to the backward error analysis, RBA is of the order of $\tilde{\Pi}\cdot\varepsilon$. Indeed,

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

Hence,

$$\sigma_{(n+rank)}\{S(\lambda_o)\} \approx \tilde{\Pi}\cdot\varepsilon \left\| \begin{matrix} A & B \\ C & D \end{matrix} \right\|_2 \approx \tilde{\Pi}\cdot\varepsilon\, \sigma_1\{S(\lambda_o)\} \tag{44}$$

This allows us to estimate the value of $\tilde{\Pi}$. Note that an ϵ-small

backward error does not imply that the zeros are also computed up to ϵ accuracy. This also depends on the conditioning of each separate zero [21].

EXAMPLE 1: This example is the 16th 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 [26]. The data for this example appears in Appendix I. Since D has rank 4, there are fifteen zeros. The following shows the computed zeros to thirteen significant digits.

Zeros	$RBA = \frac{\sigma_{21}}{\sigma_1} \approx \tilde{\Pi}\,\epsilon$
-829.2490955651110	$< \epsilon$
789.8985828158399	$< \epsilon$
141.2294550203129	$< \epsilon$
- 50.46757476394580 ± j1.0319141604160423297	$< \epsilon$
- 49.63760103236723	$< \epsilon$
- 13.76530730452916 ± j9.110214747547156	$< \epsilon$
.6659561616385485	$< \epsilon$
- 6.710651036803525	$< \epsilon$
- 2.003403155575229	$< \epsilon$
- 23.13366516893961	$< \epsilon$
- 20.55602749379905 ± j1.417353350011828	$< \epsilon$
- 18.95850189406822	$< \epsilon$

EXAMPLE 2: This is a 9th order model of a boiler system [27] and the data is shown in Appendix I. The zeros along with the corresponding relative backward errors, are as follows:

Zeros	$RBA = \frac{\sigma_{11}}{\sigma_1} \approx \tilde{\Pi}\,\epsilon$
-26.39513728882773	$< \epsilon$
- 2.957771985292086 ± j.3352672071870191	$< \epsilon$
.7486064441907556	$< \epsilon$
.09403261020202233	$< \epsilon$
- .009546070612163396	$< \epsilon$

EXAMPLE 3: This is a 6th order example from [1] with,

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

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

Zeros	$RBA = \frac{\sigma_8}{\sigma_1} \approx \tilde{\Pi} \epsilon$
-.6823278038280190	$< \epsilon$
.3411639019140096 ± j1.161541399997251	$< \epsilon$
.9999999999999997	$< \epsilon$

EXAMPLE 4: This is a fifth order model of a drum boiler [28] with,

$$A = \begin{bmatrix} -0.129 & 0.000 & 0.396\times10^{-1} & 0.250\times10^{-1} & 0.191\times10^{-1} \\ 0.329\times10^{-2} & 0.000 & -0.779\times10^{-4} & 0.122\times10^{-3} & -0.621 \\ 0.718\times10^{-1} & 0.000 & -0.100 & 0.887\times10^{-3} & -0.385\times10^{1} \\ 0.411\times10^{-1} & 0.000 & 0.000 & -0.822\times10^{-1} & 0.000 \\ 0.361\times10^{-3} & 0.000 & 0.350\times10^{-4} & 0.426\times10^{-4} & -0.743\times10^{-1} \end{bmatrix}$$

$$B = \begin{bmatrix} 0.000 & 0.1 \\ 0.000 & 0.359\times10^{-4} \\ 0.000 & -0.989\times10^{-2} \\ 0.249\times10^{-4} & 0.000 \\ 0.000 & -0.534\times10^{-5} \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

The zeros and the relative backward errors are:

Zeros	$RBA = \frac{\sigma_7}{\sigma_1} \approx \tilde{\Pi}\,\epsilon$
-.368051203603595	$< \epsilon$
-.06467751189941505	$< \epsilon$

EXAMPLE 5: Consider the system with,

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

corresponding to the transfer function $H(s) = \frac{1}{s^{15}}$. 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 rest are located on a circle with radius $\epsilon^{-1/n}$. Our algorithm has no such difficulty and will indicate that there are no finite zeros.

EXAMPLE 6: None of the previous algorithms can handle the case where the system is degenerate. Consider the case where,

$$S(\lambda) = \left|\begin{array}{ccc:c} x-2 & 1 & 0 & 0 \\ 0 & A & 0 & 0 \\ 1 & 0 & \lambda & 1 \\ \hdashline 0 & 1 & 0 & 0 \end{array}\right|$$

This pencil has right and left Kronecker indices equal to one.
The QZ algorithm [9] will indicate degeneracy (i.e., a $\frac{0}{0}$ 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 [21]-[22]. Our algorithm will extract the singular part of S(X) and will yield a regular pencil containing the single zero at 2.

EXAMPLE 7: Consider the system,

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

with Kronecker right and left indices equal to one. The QZ algorithm will indicate degeneracy whereas in fact there are no zeros as detected by our algorithm. Davison and Wang'-s method [1] will find two zeros at 0.

EXAMPLE 8: This is the model of an electrical network [29] with,

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

$$C = \begin{bmatrix} 0 & 0 & 0 & 10 & 0 \end{bmatrix}$$

Zeros	RBA= $\frac{\|7}{\sigma_1} \approx \tilde{\Pi} \epsilon$
$-.9999999999999994 \pm j.1821927265261758 \times 10^{-7}$	$< \epsilon$

The system actually has two zeros at -1.0, but note that the error is of the order of $\epsilon^{1/2}$ which is to be expected because of the presence of a 2×2 Jordan block.

EXAMPLE 9: Consider the system with,

$$S(\lambda) = \begin{bmatrix} \lambda & & & & -1 \\ -1 & \ddots & & & .05 \\ & \ddots & \ddots & & \vdots \\ 0 & & \ddots & & 0 \\ & & & -1 \quad \lambda & 0 \\ \hline 0 & & 0 & 1 & 0 \end{bmatrix} \Big\} 15$$

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 cannot be discerned from the rest anymore. Our algorithm will compute the zero at 20 with no difficulty.

EXAMPLE 10: This example is taken from [30].

$$A = \begin{bmatrix} 0 & 0 & 0 & -24 & & & & & & & & \\ 1 & 0 & 0 & -50 & & & & & & & & \\ 0 & 1 & 0 & -35 & & & & & & & & \\ 0 & 0 & 1 & -10 & & & & & & & & \\ \hline & & & & 0 & 0 & 0 & -30 & & & & \\ & & & & 1 & 0 & 0 & -61 & & & & \\ & & & & 0 & 1 & 0 & -41 & & & & \\ & & & & 0 & 0 & 1 & -11 & & & & \\ \hline & & & & & & & & 0 & 0 & -15 \\ & & & & & & & & 1 & 0 & -23 \\ & & & & & & & & 0 & 1 & -9 \end{bmatrix} \quad B = \begin{bmatrix} 135 & 18 & 14 & 20 \\ 117 & 42 & 25 & 33 \\ 33 & 30 & 13 & 15 \\ 3 & 6 & 2 & 2 \\ \hline 4 & 10 & 50 & 32 \\ 6 & 17 & 20 & 32 \\ 2 & 8 & 2 & 8 \\ 0 & 1 & 0 & 0 \\ \hline 36 & 0 & 5 & 68 \\ 14 & -10 & 6 & 54 \\ 2 & -2 & 1 & 10 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The system has input decoupling zeros at,

Zeros	$RBA = \frac{\sigma_{11}}{\sigma_1} \approx \tilde{\Pi}\ \epsilon$
-4.999999999999955	$< \epsilon$
-2.999999999999988	$< \epsilon$

and no output decoupling zeros.

EXAMPLE 11: Consider the rectangular system [31],

$$A = \left[\begin{array}{rrrrr} -2 & -6 & 3 & -7 & 6 \\ 0 & -5 & 4 & -4 & 8 \\ 0 & 2 & 0 & 2 & -2 \\ 0 & 6 & -3 & 5 & -6 \\ 0 & -2 & 2 & -2 & 5 \end{array}\right], \quad B = \begin{bmatrix} -2 & 7 \\ -8 & -5 \\ -3 & 0 \\ 1 & 5 \\ -8 & 0 \end{bmatrix}$$

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

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

Zeros	$RBA = \frac{\sigma_7}{\sigma_1} \approx \tilde{\Pi}\ \epsilon$
-3.000000000000000	$< \epsilon$
3.999999999999972	$< \epsilon$

V. CONCLUSIONS

We have presented a new algorithm for computing the zeros of a linear multivariable system. The algorithm is superior to previous methods in that it effectively deals with all cases. It is backward stable and more efficient compared to earlier techniques. The algorithm 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.

Acknowledgement

We gratefully acknowledge the encouragement and useful discussions with Professors G.F. Franklin and L.M. Silverman.

REFERENCES

[1] Davison, E.J. and S.H. Wang, "Properties and calculation of transmission zeros of linear multivariable systems," Automatica 10, 643-658, 1974.

[2] Davison, E.J., "The robust control of a servomechanism for linear time-invariant multivariable systems," IEEE Trans. on AC, Vol. AC-21, No.1, 1976.

[3] Franklin, G.F., "A new formulation of the multivariable robust servo-mechanism problem," Contributions to Workshop on Adaptive Control, Champaign, Illinois, 1978.

[4] Francis, B.A., and W.M. Wonham, "The role of transmission zeros in linear multivariable regulators," Int. J. Control, 22, 657-681, 1975.

[5] Desoer, C.A., and J.D. Schulman, "Zeros and poles of matrix transfer functions and their dynamical interpretation," IEEE Trans. Circ. Sys. CAS-21, 3-8, 1974.

[6] Rosenbrock, H.H., "State-space and multivariable theory," 1970.

[7] MacFarlane, A.G.J. and N. Karcanias, "Poles and zeros of linear multi-variable systems: a survey of the algebraic, geometric, and complex-variable theory," Int. J. Control, 24. 33-74 , 1976.

[8] Davison, E.J. and S.H. Wang, "Remark on a multiple transmission zeros of a system," Automatica 12, 195, 1976.

[9] Laub, A.J., and B.C. Moore, "Calculation of transmission zeros using QZ techniques," Automatica 14, 557-566, 1978.

[10] Van Dooren, P., A. Emami-Naeini and L. Silverman, "Stable extraction of the Kronecker structure of pencils," IEEE Conf.on Decision and Control, 521-524, 1978.

[11] Emami-Naeini, A., "Multivariable poles and zeros," Internal report, Information Systems Laboratory, Stanford University, 1978.

[12] Van Dooren, P., "The computation of Kronecker's canonical form of a singular pencil," Linear Algebra and its Applications, 27, 103-141, 1979.

[13] Van Dooren, P., "The generalized eigenstructure problem in linear system theory," submitted to IEEE Trans. on Automatic Control.

[14] Verghese, G., P. Van Dooren, and T. Kailath, "Properties of the system matrix of a generalized state-space system," Int. J. Contr. Vol. 30, 1979.

[15] Molinari, B.P., "Structural invariants of linear multivariable systems," Int. J. Control, 28, 493-510, 1978.

[16] Aplevich, J.D., "Tableau methods for analysis and design of linear systems," Automatica, 15, 419-429, 1979.

[17] Moylan, P.J., "Stable inversion of linear systems," IEEE Trans. Aut. Contr. Vol. AC-22, 74-78, 1977.

[18] Jordan, D. and L.F. Godbout, "On the computation of the canonical pencil of a linear system," IEEE Trans. Aut. Contr., AC-22, 126-128, 1977.

[19] Thompson, G.L. and R.L. Weil, "The roots of matrix pencils $AY = \lambda BY$: existence, calculations and relations to game theory," Lin. Alg. and Appl., Vol. 5, 207-226, 1972.

[20] Moler, C.B. and G.W. Stewart, "An algorithm for generalized matrix eigenvalue problems" SIAM J. Numer. Anal. 10, 241-256, 1973.

[21] Wilkinson, J.H., The Algebraic Eigenvalue Problem, Oxford University Press, London, 1965.

[22] Wilkinson, J.H., "Linear differential equations and Kronecker canonical form," in Recent Advances in Numerical Analysis, ed. C. de Boor, G. Golub, Academic Press, New York, 1978.

[23] Wilkinson, J.H., "Kronecker canonical form and the QZ algorithm," NPL Report DNACS 10/78, 1978.

[24] Garbow, B.S. et al., "Matrix eigensystem routines - EISPACK guide extension. Lecture Notes in Computer Science," Vol. 51, Springer-Verlag, Berlin, 1977.

[25] Dongarra, J.J. et al. "LINPACK User's Guide," Society for Industrial and Applied Mathematics, 1979.

[26] Sain, M.K., et al. "Alternatives for linear multivariable control," National Engineering Consortium, Inc. Chicago, 1978.

[27] Axelby, G., A.J. Laub and E.J. Davison, "Further discussion on the calculation of transmission zeros," Automatica, 403-405, 1978.

[28] Bengtsson, G., "A theory for control of linear multivariable systems," Report 7341, Lund Institute of Technology, 1973.

[29] Kaufman, I., "On poles and zeros of linear systems," IEEE Trans. Circ. Theory, 20, 93-101, 1973.

[30] Kalman, R.E., "Mathematical description of linear dynamical systems," SIAM J. of Control, Vol. 1, No. 2, 153-192, 1963.

[31] Kouvaritakis, B. and A.G.J. MacFarlane, "Geometric approach to the analysis and synthesis of system zeros," Part 1 and 2, Int. J. Control, 23, 149-181, 1976.

APPENDIX I

Jet engine (example 1):

F100 MODEL ALT=0.0 PLA=83

THE A MATRIX

-4.328	.1714	5.376	401.6	-724.6	-1.933	1.020	-.9820
-.4402	-5.643	127.5	-233.5	-434.3	26.59	2.040	-2.592
1.038	6.073	-165.0	-4.483	1049.	-82.45	-5.314	5.097
.5304	-.1086	131.3	-578.3	102.0	-9.240	-1.146	-2.408
.8476E-02	-.1563E-01	.5602E-01	1.573	-10.05	.1952	-.8804E-02	-.2110E-01
.8350	-.1249E-01	-.3567E-01	-.6074	37.65	-19.79	-.1813	-.2962E-01
.6768	-.1264E-01	-.9683E-01	-.3567	80.24	-.8239E-01	-20.47	-.3928E-01
-.9696E-01	.8666	16.87	1.951	-102.3	29.66	.5943	-19.97
-.8785E-02	-.1636E-01	.1847	.2169	-8.420	.7003	.5666E-01	6.623
-.1298E-03	-.2430E-03	.2718E-02	.3214E-02	-.1246	.1039E-01	.8395E-03	.9812E-01
-1.207	-6.717	26.26	12.49	-1269.	103.0	7.480	36.84
-.2730E-01	-.4539	-52.72	193.5	-28.09	2.243	.1794	9.750
-.1206E-02	-.2017E-01	-2.343	8.835	-1.248	.9975E-01	.8059E-02	.4333
-.1613	-.2469	-24.05	23.38	146.3	1.638	.1385	4.486
-.1244E-01	.3020E-01	-.1198	-.4821E-01	5.675	-.4525	19.81	.1249
-1.653	1.831	-3.822	113.4	341.4	-27.34	-2.040	-.6166
.9990	1.521	-4.062	9.567	10.08	-.6017	-.1312	.9602E-01
11.32	10.90	-4.071	-.5739E-01	-.6063	-.7488E-01	-.5936	-.9602E-01
-.9389E-02	.1352	5.638	.2246E-01	.1797	.2407E-01	1.100	.2743E-01
-3.081	-4.529	5.707	-.2346	-2.111	-.2460	-.4686	-.3223
.2090E-02	-.5256E-01	-.4077E-01	-.9182E-02	-.8178E-01	.3428E-01	.4995E-02	-.1256E-01
-.1953E-01	-.1622	-.6439E-02	-.2346E-01	-.2201	-.2514E-01	-.3749E-02	-.3361E-01
.1878E-01	-.2129	-.9337E-02	-.3144E-01	-.2919	-.3370E-01	.8873E-01	-.4458E-01
.2253E-01	.1791	.8371E-02	.2645E-01	.2560	.2835E-01	-.3749E-01	.3635E-01
-49.99	.6760E-01	39.46	.4991E-02	.8983E-01	.5349E-02	.0	.1372E-01
-.6666	-.6657	.5847	.6654E-04	.1347E-02	.7131E-04	.0	.2057E-03
.2854	2.332	-47.65	.3406	3.065	.3624	-.4343	.4681
-9.627	-9.557	38.48	-50.01	.1011	.1203E-01	-.4686E-01	.1715E-01
-.4278	-.4245	1.710	-2.000	-1.996	.5349E-03	-.1999E-02	.7544E-03
-4.414	-4.354	17.66	-3.113	-3.018	-19.77	-.4999E-01	.1509E-01
-.1127E-02	-.6760E-02	.1835E-01	-.9981E-03	-.1347E-01	-.1070E-02	-20.00	-.2057E-02
.5004	-.1437	-2.416	-.1073	-1.078	30.63	19.89	-50.16

THE B MATRIX

-.4570E-01	-451.6	-105.8	-1.596	851.5
.1114	-546.1	-6.575	-107.8	3526.
.2153	1362.	13.46	20.14	-.6777E+05
.3262	208.0	-2.888	-1.653	-269.1
.9948E-02	-98.39	.5069	-.1940	-94.70
.2728E-01	71.62	9.608	-.3160	-184.1
.1716E-01	71.71	8.571	.7989	-515.2
-.7741E-01	-141.2	-.8215	39.74	1376.
.3855E-01	-7.710	-.4371E-01	-.1024	-6684.
.5707E-03	-.1144	-.6359E-03	-.1432E-02	-99.02
5.727	-1745.	-8.940	-17.96	.8898E+05
.1392	-24.30	-.2736	-.3403	-6931.
.6172E-02	-1.082	-.1183E-01	-.1452E-01	-307.7
.6777E-01	16.60	.3980	.2311E-01	-2588.
.1880E-02	9.147	-.8241	.8984E-01	-32.31
.1677	435.8	-89.94	4.900	-295.5

THE C MATRIX

.4866	-.6741	5.392	95.42	24.03	10.52	.8190	-.4492
.1383E-01	.2789E-05	.0	.0	-.1081E-01	-.5545E-04	.4722E-04	.0
.0	.0	.0	.0	.0	.0	.0	.0
.7418E-04	.5496E-05	.4790E-05	.1478E-03	-.1504E-01	-.6503E-04	.8820E-04	.4999E-05
.1538E-04	.1201E-03	-.2579E-02	-.1609E-03	.1618E-01	-.1071E-02	-.9561E-04	-.5503E-05
.5195	.8437	-1.863	.5709E-01	.4815	3.428	2.161	.7681E-01
.0	.0	.0	.0	.0	.0	.0	.0
.0	.0	1.000	.0	.0	.0	.0	.0
.3434E-05	.2727E-04	.1128E-05	.4002E-05	.3673E-04	.4290E-05	-.4958E-05	.5609E-05
-.3732E-05	-.2996E-04	-.1234E-05	-.4380E-05	-.4024E-04	-.4721E-05	.5324E-05	-.6103E-05

THE D MATRIX

-.6777E-01	-420.5	32.97	-1.824	1245.
.1282E-03	.3353	.6804	-.5605E-04	-.1199E-01
.0	.0	.0	.0	.0
.1030E-05	-.1193E-01	-.5806E-02	.6015E-04	.4463E-01
.8109E-05	.2328E-01	.1178E-03	-.5538E-02	-.1039

Boiler (example 2):

MATRIX A

-3.93	-3.15×10^{-3}	0	0	0	4.03×10^{-5}	0	0	0
3.68×10^{2}	-3.05	3.03	0	0	-3.77×10^{-3}	0	0	0
2.74×10^{1}	7.87×10^{-2}	-5.96×10^{-2}	0	0	-2.81×10^{-4}	0	0	0
-6.47×10^{-2}	-5.20×10^{-5}	0	-2.55×10^{-1}	-3.35×10^{-6}	3.60×10^{-7}	6.33×10^{-5}	1.94×10^{-4}	0
3.85×10^{3}	1.73×10^{1}	-1.28×10^{1}	-1.26×10^{4}	-2.91	-1.05×10^{-1}	1.27×10^{1}	4.31×10^{1}	0
2.24×10^{4}	1.80×10^{1}	0	-3.56×10^{1}	-1.04×10^{-4}	-4.14×10^{-1}	9.00×10^{1}	5.69×10^{1}	0
0	0	2.34×10^{-3}	0	0	2.22×10^{-4}	-2.03×10^{-1}	0	0
0	0	0	-1.27	1.00×10^{-3}	7.86×10^{-5}	0	-7.17×10^{-2}	0
-2.20	-1.77×10^{-3}	0	-8.44	-1.11×10^{-4}	1.38×10^{-5}	1.49×10^{-3}	6.02×10^{-3}	-1.00×10^{-10}

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1.56 & 0 \\ 0 & -5.13\times10^{-6} \\ 8.28 & -1.55 \\ 0 & 1.78 \\ 2.33 & 0 \\ 0 & -2.45\times10^{-2} \\ 0 & 2.94\times10^{-5} \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

APPENDIX II

```
      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=l,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=l,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 xl                    | -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
      11 =NU+MU
      J1=MNU+1
      IO=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
      ROl=RO
      MNU=M+NU
      NUMU=NU+MU
      IF(M.EQ.0) GO TO 120
```

```
      RO1=RO1+1
      IROW=NU
      IF(SIGMA.LE.l) GO TO 40
C***  COMPRESS ROWS OF D. FIRST EXPLOIT TRIANGULAR SHAPE ***
      Ml=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
      ROl=ROl-1
   45 IF(SIGMA.EQ.M) GO TO 60
      DO 55 ICOL=SIGMA,M
        DUM=0.DO
        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=l,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 ***
      I1=NU+SIGMA
      MM1=M+1
      N1=NU
      IF(TAU.EQ.l) GO TO 140
      DO 135 I=1,TAU
        DUM=0.DO
        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=ROl-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 220
      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.DO
      DO 10 I=1,K
   10   S=S+DUMMY(I)*DUMMY(I)
      ALFA=DSQRT(S)
      IF (ALFA.LE.EPS) RETURN
      ZERO=.FALSE.
```

```
241.            DUM1=DUMMY(J)
242.            IF(DUM1.GT.0.D0) ALFA=-ALFA
243.            DUMMY(J)=DUM1-ALFA
244.            S=1.D0/(S-ALFA*DUM1)
245.            RETURN
246.            END
247.            SUBROUTINE TR1(A,MDIMA,U,S,I1,I2,J1,J2)
248.      C***  THIS ROUTINE PERFORMS THE HOUSEHOLDER TRANSFORMATION H=I-S.UU'
249.      C***  ON THE ROWS I1+1 TO I1+I2 OF A, THIS FROM COLUMNS J1 TO J2.
250.      C***
251.            REAL*8 A(MDIMA,1),U(I2),S,INPROD,Y
252.            DO 20 J=J1,J2
253.              INPROD=0.D0
254.              DO 10 I=1,I2
255.         10     INPROD=INPROD+U(I)*A(I1+I,J)
256.              Y=INPROD*S
257.              DO 20 I=1,I2
258.         20     A(I1+I,J)=A(I1+I,J)-U(I)*Y
259.            RETURN
260.            END
261.            SUBROUTINE TR2(A,MDIMA,U,S,I1,I2,J1,J2)
262.      C***  THIS ROUTINE PERFORMS THE HOUSEHOLDER TRANSFORMATION H=I-S.UU'
263.      C***  ON THE COLUMNS Jl+l TO J1+J2 OF A, THIS FROM ROWS 11 TO 12.
264.      C***
265.            REAL*8 A(MDIMA,1),U(J2),S,INPROD,Y
266.            DO 20 I=I1,I2
267.            INPROD=0.D0
268.            DO 10 J=1,J2
269.         10   INPROD=INPROD+U(J)*A(I,J1+J)
270.            Y=INPROD*S
271.            DO 20 J=1,J2
272.         20   A(I,J1+J)=A(I,J1+J)-U(J)*Y
273.            RETURN
274.            END
```