NUMERICAL ANALYSIS PROJECT
MANUSCRIPT NA-80-02

JULY 1980

A GENERALIZED EIGENVALUE APPROACH FOR SOLVING RICCATI EQUATIONS

BY

P. VAN DOOREN

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

A GENERALIZED EIGENVALUE APPROACH FOR SOLVING RICCATI EQUATIONS*

P. Van Dooren#

Abstract

A numerically stable **algorithm** is derived to compute orthonormal bases for any deflating subspace of a regular pencil $\lambda B-A$. The method is based on an update of the QZ-algorithm, in order to obtain any desired ordering of eigenvalues in the quasi-triangular forms constructed by this algorithm.

As applications we discuss a new approach to solve Riccati equations arising in linear system theory. The computation of deflating subspaces with specified spectrum is shown to be of crucial importance here.

---

* This research was supported by the National Science Foundation under Grant ENG78-10003 and by the U.S.Air Force under Grant AFOSR-79-0094.

Dept. El. Eng. and Dept. Comp. Sc., Stanford University, Stanford, CA 94305.

Present address: Philips Research Lab., Av. Van Becelaere 2, Box 8, B-1170 Brussels, Belgium

I. Introduction

The computation of deflating subspaces with specified spectrum has not received a great deal of attention until it was recently applied to the solution of the optimal control problem of a linear discrete time system [5][15]. Before the development of reliable software for the generalized eigenvalue problem [13][16], these problems were often reduced to an equivalent standard eigenvalue problem and gave rise to the computation of invariant subspaces with specified spectrum [8][14] [17][21]. The matrix involved in this standard eigenvalue problem does not consist of given data but has to be computed, which, unfortunately, requires inverses of possibly ill-conditioned matrices. In [5][15] the use of a generalized eigenvalue problem is recommended as a safer alternative, and the attention is drawn to the absence of appropriate software for computing deflating subspaces of a regular pencil.
In this paper we try to fill this gap and we also exploit this new tool in a class of related problems arising in linear system theory. We thereby develop a new approach to tackle these problems in a numerically sound way.

In the **rest** of this section we briefly review some notions that we will need in later sections. The material covered here can be found e.g. in [13][18][19][20].
Notations will be as follows. We use uppercase for matrices and lowercase for vectors and scalars. $\mathbb{R}$ and $\mathbb{C}$ are the fields of real and complex numbers, respectively. We use A^* (resp. x^*) for the conjugate transpose of a complex matrix A (resp. vector x) and A' (resp. x') for the transpose of a real matrix A (resp. vector x). $\|.\|_2$ denotes the spectral. norm

of a matrix and the Euclidean norm of a vector. A complex (real) square matrix A is called unitary (orthogonal) when $A^*A=I$ ($A'A=I$). Wnen no explicit distinction is made between the complex and real case, we use the term unitary and the notation A^* for the real case as well.

Recently, more attention has been paid to the generalized eigenvalue problem (GEP) :

$$Ax = \lambda Bx \tag{1}$$

where B is not necessarily invertible but where the pencil $\lambda B-A$ is regular, i.e. :

$$\det.(\lambda B-A) \not\equiv 0 \tag{2}$$

When the coefficients of the matrices A and B belong to $\mathbb{C}$, there exist unitary transformations Q and Z reducing the nxn pencil AB-A to the upper triangular form :

$$Q^*(\lambda B-A)Z = \lambda\hat{B}-\hat{A} = \lambda \begin{bmatrix} \hat{b}_{11} & \cdots & * \\ & \ddots & \vdots \\ 0 & & \hat{b}_{nn} \end{bmatrix} - \begin{bmatrix} \hat{a}_{11} & \cdots & * \\ & \ddots & \vdots \\ 0 & & \hat{a}_{nn} \end{bmatrix} \tag{3}$$

The ratios $\lambda_i=\hat{a}_{ii}/\hat{b}_{ii}$ are called the generalized eigenvalues of the pencil $\lambda B-A$. The set $\{\lambda_1,\ldots,\lambda_n\}$ is called the spectrum of $\lambda B-A$ and is denoted by $\Lambda(B,A)$; it may contain repeated elements. Notice that λ_i may be infinite (when $\hat{b}_{ii}=0$) but it is never undetermined (i.e. $\lambda_i=0/0$) since $\hat{a}_{ii}=\hat{b}_{ii}=0$ implies $\det.(\lambda\hat{B}-\hat{A}) \equiv 0$ and hence $\det.(\lambda B-A) \equiv 0$. As a consequence the matrix $\hat{a}_{ii}B-\hat{b}_{ii}A$ is singular. The vectors x_i satisfying

$$(\hat{a}_{ii}B-\hat{b}_{ii}A)x_i = 0 \tag{4}$$

are called generalized eigenvectors of XB-A corresponding to λ_i.
If the eigenvalue $\lambda_i=\hat{a}_{ii}/\hat{b}_{ii}$ has a larger multiplicity than the number of independent solutions x_i of (4) then one can define generalized principal vectors $\tilde{x}_i$ of $\lambda B-A$ corresponding to λ_i. Since we do not need this concept in the sequel, we do not go into further details about it.

In the real case the decomposition (3) also exists but involves complex matrices Q, Z, $\hat{A}$ and $\hat{B}$ when $\Lambda(B,A)$ contains complex elements. Under orthogonal transformations Q and Z, $\lambda B-A$ can be transformed to the quasi upper triangular form :

$$Q'(\lambda B-A)Z = \lambda\hat{B}-\hat{A} = \lambda\begin{bmatrix}\hat{B}_{11} & & * \\ & \ddots & \\ 0 & & \hat{B}_{kk}\end{bmatrix} - \begin{bmatrix}\hat{A}_{11} & & * \\ & \ddots & \\ 0 & & \hat{A}_{kk}\end{bmatrix} \qquad (5)$$

where the diagonal pencils $\lambda\hat{B}_{ii}-\hat{A}_{ii}$ have sizes $d_i=1$ or 2, and the $\hat{B}_{ii}$ are upper triangular. If $d_i=1$ then $\Lambda(\hat{B}_{ii},\hat{A}_{ii})$ is real (maybe infinite). If $d_i=2$ then $\Lambda(\hat{B}_{ii},\hat{A}_{ii})$ contains two (finite) complex conjugate numbers. The spectrum of $\lambda B-A$ is the union of the sets $\Lambda(\hat{B}_{ii},\hat{A}_{ii})$, as can be seen from an additional (unitary) reduction of (5) to (3).

An algorithm has been derived recently to obtain decompositions of the type (3) and (5) in a numerically stable way [13].

When B=I, (1) boils down to the standard eigenvalue problem (SEP):

$$Ax = xx \qquad (6)$$

Tt is readily verified that the decompositions (3) and (5) then reduce to the classical Schur decompositions of the real or complex matrix A, respectively. We therefore call (3) and (5) generalized Schur decompositions of the regular pencil XB-A. Tn the sequel we drop the term "generalized" when no confusion is possible from the context.

The notion of eigenvector in the GEP can be extended to the notion of deflating subspace X of a regular pencil XB-A, satisfying :

$$\dim.(BX + AX) = \dim.X \tag{7}$$

where dim.S denotes the dimension of a subspace S. Let X have dimension ℓ and suppose that the ℓ first columns of the unitary matrices Q and Z, partitioned as

$$Z = [\underbrace{Z_1}_{\ell} | \underbrace{Z_2}_{n-\ell}] \; ; \; Q = [\underbrace{Q_1}_{\ell} | \underbrace{Q_2}_{n-\ell}] \tag{8}$$

span the spaces X and $AX+BX$, respectively. Then it follows from (7) that $Q_2^* A Z_1 = Q_2^* B Z_1 = 0$, or :

$$Q^*(\lambda B - A)Z = \lambda \begin{bmatrix} \hat{B}_{11} & \hat{B}_{12} \\ \hline 0 & \hat{B}_{22} \end{bmatrix} - \begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} \\ \hline 0 & \hat{A}_{22} \end{bmatrix} \begin{matrix} \}\ell \\ \}n-\ell \end{matrix} \tag{9}$$

Conversely, if (8)(9) hold then the columns of Z_1 span a deflating subspace X according to (7). For $\ell=1$, X is an eigenvector of $\lambda B-A$ corresponding to the eigenvalue $\Lambda(\hat{B}_{11},\hat{A}_{11})$. For any ℓ, $\Lambda(\hat{B}_{11},\hat{A}_{11})$ is a subset of $\Lambda(B,A)$ and is denoted as $\Lambda(B,A)|_X$ (the spectrum of XB-A restricted to X). The deflating subspace X is uniquely determined by $\Lambda(B,A)|_X$ when this subset is disjoint from the rest of $\Lambda(B,A)$ (X is then spanned by the eigenvectors and principal vectors corresponding to the spectrum $\Lambda(B,A)|_X$). All this also holds for the real case.

For the case $B=I$, definition (7) of a deflating subspace reduces to the definition of an invariant subspace X of A, since di.m. $(X+AX)=\dim.X$ is equivalent to $AX \subset X$. Notice also that in the SEP, Q is equal to Z in (8)(9).

It follows now immediately from (9) that the Z matrix in the Schur decomposition (3) yields orthonormal bases for deflating subspaces of dimension 1 to n-l, since the right hand side of (3) has a block partitioning of the type (9) for $\ell=1,\ldots,n-1$. This also holds for the 'real' Schur decomposition (5) for these ℓ that are conformable with the block partitioning in (5), namely :

$$\ell = \sum_{j=1}^{i} d_j \qquad \text{for } i=1,\ldots,k-1 \qquad (10)$$

In this paper we consider the computation of a deflating subspace X with prescribed spectrum $\Lambda(B,A)|_X=\{\mu_1,\ldots,\mu_\ell\}$. From the above it follows that the ℓ first columns of Z in (3) form an orthonormal basis for such a space X if and only if the sets $\{\lambda_i=\hat{a}_{ii}/\hat{b}_{ii} \mid i=1,\ldots,\ell\}$ and $\{\mu_i \mid i=1,\ldots,\ell\}$ are equal except for the ordering of their elements. In the real case, this also holds for the matrix Z in (5) when ℓ satisfies (10). The complex elements in $\{\mu_i \mid i=1,..,\ell\}$ must therefore appear in conjugate pairs.

The problem thus reduces to obtaining decompositions of the type (3) and (5) but with prescribed ordering of the eigenvalues occurring on diagonal. In the next section we show how to solve this problem by deriving a method to interchange the order of the eigenvalues in the decompositions (3) and (5), which were previously obtained by the QZ-algorithm. The method is proved to be numerically stable. In section III we apply this new tool to derive new methods for solving Ficcati equations arising in linear system theory. In these methods, deflating subspaces with specified spectrum (namely all the eigenvalues inside the unit. circle or all the eigenvalues in the left half plane) have to

be computed. In Section IV we give some numerical examples and a FORTRAN program implementing the reordering is given in Appendix.

II. Reordering

It is clear that the **1x1** and 2x2 diagonal blocks in the decompositions (3) and (5) can be reordered in an arbitrary way by using a method to interchange two consecutive blocks only. This idea was e.g. used in the SEP to obtain standard Schur forms with an arbitrary ordering of the eigenvalues [8][17][21]. The method described hereafter can be viewed as a generalization of it to the GEP.

We thus want to find unitary transformations Q and Z such that

$$Q^*AZ = Q^*\left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline 0 & A_{22} \end{array}\right] Z = \hat{A} = \left[\begin{array}{c|c} \hat{A}_{11} & \hat{A}_{12} \\ \hline 0 & \hat{A}_{22} \end{array}\right] \qquad (11a)$$

$$Q^*BZ = Q^*\left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline 0 & B_{22} \end{array}\right] Z = \hat{B} = \left[\begin{array}{c|c} \hat{B}_{11} & \hat{B}_{12} \\ \hline 0 & \hat{B}_{22} \end{array}\right] \qquad (11b)$$

where $\Lambda(B_{11},A_{11})=\Lambda(\hat{B}_{22},\hat{A}_{22})$ and $\Lambda(B_{22},A_{22})=\Lambda(\hat{B}_{11},\hat{A}_{11})$, and where the dimensions d_1 and d_2 are either **1** or 2.

Moreover, we want the transformations Q and Z to be numerically stable. In order to prove this we use a standard error analysis [23] of (possibly complex) transformations of the type:

$$G^*y = G^*\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ 0 \end{bmatrix} \qquad (12a)$$

where G is the (possibly complex) Givens transformation :

$$G = \left[\begin{array}{cc|c} c & \bar{s} & \\ s & -\bar{c} & \mathbf{1} \\ \hline & & \mathbf{1} \end{array}\right] \quad c\bar{c}+s\bar{s}=1 \qquad (12b)$$

constructed to annihilate y_2. Let $\tilde{c}$, $\tilde{s}$ (defining $\tilde{G}$) and $\tilde{y}_1$ be the computed versions of c, s and $\hat{y}_1$, respectively , and let ε be the machine

precision of the computer, then a backward error analysis yields (for a standard construction of such transformations):

$$\tilde{G}^*(y+e_y) = \begin{bmatrix} \tilde{y}_1 \\ 0 \end{bmatrix}; \|e_y\|_2 \leqslant 6 \bullet E\|y\|_2 \qquad (13)$$

Here we assume that the 0 element is not computed but put equal to zero. Wnen performing the transformation $G^*z=\hat{z}$ for an arbitrary vector z, we have similarly:

$$\tilde{G}^*(z+e_z) = \begin{bmatrix} \tilde{z}_1 \\ \tilde{z}_2 \end{bmatrix}; \|e_z\|_2 \leqslant 6.\varepsilon\|z\|_2 \qquad (14)$$

In the sequel G_{ij} denotes the class of matrices representing Givens transformations between columns or rows i and j. We prove that by using transformations in this class for the reduction (11), the backward error can be bounded with respect to

$$\Delta = \max\{\|A\|_2, \|B\|_2\} \qquad (15)$$

Case I: $d_1=d_2=1$:

This may occur in both decompositions (3) and (5). We thus assume that the matrices can be complex. We have the following configuration :

$$Q^*AZ = Q^*\begin{bmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{bmatrix} Z = \begin{bmatrix} \hat{a}_{11} & \hat{a}_{12} \\ 0 & \hat{a}_{22} \end{bmatrix} = \hat{A} \qquad (16a)$$

$$Q^*BZ = Q^*\begin{bmatrix} b_{11} & b_{12} \\ 0 & b_{22} \end{bmatrix} Z = \begin{bmatrix} \hat{b}_{11} & \hat{b}_{12} \\ 0 & \hat{b}_{22} \end{bmatrix} = \hat{B} \qquad (16b)$$

We can assume without loss of generality that $|b_{22}| \geqslant |a_{22}|$ (if this is not the case the role of A and B should be interchanged).

A construction of Q and Z such that the order of the eigenvalues is

interchanged, follows then immediately from (8)(9). Indeed, we have $\Lambda(b_{22},a_{22})=\Lambda(\hat{b}_{11},\hat{a}_{11})$ if the first column zl of Z is an eigenvector of $\lambda B-A$ corresponding to $\Lambda(b_{22},a_{22})$ or :

$$(a_{22}B-b_{22}A)Z = \left[\begin{array}{c|c} 0 & * \\ 0 & \end{array}\right] \tag{17}$$

Notice that the last row of $H=(a_{22}B-b_{22}A)$ is zero :

$$H = \begin{bmatrix} x_1 & x \\ 0 & 0 \end{bmatrix} \tag{18}$$

In order to obtain (17) we thus can choose a $Z \in G_{12}$ annihilating xl in (18). It follows from (17) that Bzl and Azl are parallel, and (16) is then obtained by choosing a $Q \in G_{12}$ annihilating x2 in Fzl:

$$Q^*BZ = Q^*\begin{bmatrix} x & x \\ x_2 & x \end{bmatrix} = \begin{bmatrix} x & x \\ 0 & x \end{bmatrix} \tag{19}$$

The assumption $|b_{22}|/|a_{22}|=|\hat{b}_{11}|/|\hat{a}_{11}|\geq 1$ implies that $\hat{b}_{11}\neq 0$, and Q^*Az_1 can then only be parallel to Q^*Bz_1 if Q*AZ is indeed upper triangular.

We now prove the numerical stability of the method. As in (13)(14), computed elements are denoted by upper tilden (~). Using the analysis (13)(14) above, it is easy to prove that all the $\varepsilon_i, i=1,..,9$ below are of the order of the machine accuracy ε of the computer.

An error analysis of (17)(18) yields :

$$(a_{22}B-b_{22}A+F)\tilde{Z} = \begin{bmatrix} 0 & x \\ 0 & 0 \end{bmatrix} \quad \text{for } \|F\|_2 = \varepsilon_1 \|a_{22}B-b_{22}A\|_2 \tag{20}$$

and of (19) :

$$\tilde{Q}^*(B+E_b)\tilde{Z} = \begin{bmatrix} \tilde{b}_{11} & \tilde{b}_{12} \\ 0 & \tilde{b}_{22} \end{bmatrix} \quad \text{for} \quad \|E_b\|_2 = \varepsilon_2 \|B\|_2 \tag{21}$$

We prove that there also exists a backward error Ea such that

$$\tilde{Q}^*(A+E_a)\tilde{Z} = \begin{bmatrix} \tilde{a}_{11} & \tilde{a}_{12} \\ 0 & \tilde{a}_{22} \end{bmatrix} \quad \text{for } \|E_a\|_2 = \varepsilon_3 \|A\|_2 \tag{22}$$

An error analysis of Q^*AZ using (14) yields

$$\tilde{Q}^*(A+E_c)\tilde{Z} = \begin{bmatrix} \tilde{a}_{11} & \tilde{a}_{12} \\ \tilde{a}_{21} & \tilde{a}_{22} \end{bmatrix} \quad \text{for } \|E_c\|_2 = \varepsilon_4 \|A\|_2 \tag{23}$$

We only have to prove that $\tilde{a}_{21} \simeq \varepsilon\Delta$ in order to obtain (22) by putting $\tilde{a}_{21}$ equal to zero.

Let us therefore denote the (2,1) elements of $\tilde{Q}^*(a_{22}B-b_{22}A)\tilde{Z}$, $\tilde{Q}^*B\tilde{Z}$ and $\tilde{Q}^*A\tilde{Z}$ by η_1, η_2 and η_3, respectively. They clearly satisfy the **relation** :

$$a_{22}\cdot\eta_2 - b_{22}\cdot\eta_3 = \eta_1 \tag{24}$$

From (20) and (21) it follows that :

$$|\eta_1| \leqslant \varepsilon_5\{|a_{22}|\|B\|_2 + |b_{22}|\|A\|_2\} \tag{25a}$$

$$|\eta_2| \leqslant \varepsilon_6\|B\|_2 \tag{25b}$$

$$|\tilde{a}_{21}| \leqslant |\eta_3| + \varepsilon_7\|A\|_2 \tag{25c}$$

Using (25) and the assumption $|b_{22}| \geqslant |a_{22}|$ in (24) we obtain :

$$\begin{aligned} |\eta_3| &\leqslant |\eta_2||a_{22}|/|b_{22}| + |\eta_1|/|b_{22}| \\ &\leqslant \varepsilon_6\Delta + \varepsilon_5\{\Delta+\Delta\} = \varepsilon_8\Delta \end{aligned} \tag{26a}$$

$$|\tilde{a}_{21}| \leqslant (\varepsilon_8+\varepsilon_7)\Delta = \varepsilon_9\Delta \tag{26b}$$

This shows the importance of the assumption $|b_{22}| \geqslant |a_{22}|$ in order to guarentee the stability of the algorithm. In case $|b_{22}| < |a_{22}|$, Q is constructed to reduce A to triangular form instead of B, and a similar analysis is then possible.

Case II: $d_1=2$, $d_2=1$:

We now have the following configuration (all matrices are real) :

$$Q'AZ = Q'\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{bmatrix} Z = \begin{bmatrix} \hat{a}_{11} & \hat{a}_{12} & \hat{a}_{13} \\ 0 & \hat{a}_{22} & \hat{a}_{23} \\ 0 & \hat{a}_{32} & \hat{a}_{33} \end{bmatrix} = \hat{A} \qquad (27a)$$

$$Q'BZ = Q'\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ 0 & b_{22} & b_{23} \\ 0 & 0 & b_{33} \end{bmatrix} Z = \begin{bmatrix} \hat{b}_{11} & \hat{b}_{12} & \hat{b}_{13} \\ 0 & \hat{b}_{22} & \hat{b}_{23} \\ 0 & 0 & \hat{b}_{33} \end{bmatrix} = \hat{B} \qquad (27b)$$

We assume that $|b_{33}| \geq |a_{33}|$. If this not the case, we can always interchange the role of A and B by transforming the first two columns of A and the last two columns of $\hat{A}$ in order to annihilate a_{21} and $\hat{a}_{32}$ and to create b_{21} and $\hat{b}_{32}$.

It follows again from (8)(9) that $\Lambda(b_{33},a_{33})=\Lambda(\hat{b}_{11},\hat{a}_{11})$ if the first column z1 of Z is an eigenvector of XB-A corresponding to $\Lambda(b_{33},a_{33})$. Therefore we have (with R any invertible row transformation):

$$R'(a_{33}B-b_{33}A)Z = \begin{bmatrix} 0 & & \\ 0 & & * \\ 0 & & \end{bmatrix} \qquad (28)$$

Notice that the last row of $H=(a_{33}B-b_{33}A)$ is zero and that we can choose $R \in G_{12}$ to annihilate the (2,1) element of H. We then have

$$RH = \begin{bmatrix} x_2 & x & x \\ 0 & x_1 & x \\ 0 & 0 & 0 \end{bmatrix} \qquad (29)$$

In order to obtain (28) we thus can choose $Z=Z_1 \cdot Z_2$ with $Z_1 \in G_{23}$ and $Z_2 \in G_{12}$ annihilating x_1 and x_2, respectively. Q is then constructed to

have $\hat{B}=Q'BZ$ in upper triangular form. We therefore take $Q=Q_1.Q_2$ where $Q_1 \in G_{23}$ is chosen to annihilate the (3,2) element created by Z_1 (i.e. $Q_1'BZ_1$ is upper triangular) and where $Q_2 \in G_{12}$ is chosen to annihilate the (2,1) element created by Z_2 (i.e. $Q_2'Q_1'BZ_1Z_2$ is upper triangular). $\hat{B}$ now satisfies (27b). Since $|b_{33}|/|a_{33}|=|\hat{b}_{11}|/|\hat{a}_{11}|\geq 1$, we have $\hat{b}_{11}\neq 0$ and because of (26), $Q'Az_1$ and $Q'Bz_1$ are parallel. This ensures that $\hat{A}=Q'AZ$ also satisfies (27a).

We now prove the numerical stability of the method.

Using (13)(14) it can be checked that all. the ε_i, i=1,..,4 below are of the order of the machine accuracy ε. An error analysis of (28)(29) yields :

$$\tilde{R}'(a_{33}B-b_{33}A+F)\tilde{Z}_1\tilde{Z}_2 = \begin{bmatrix} 0 & x & x \\ 0 & 0 & x \\ 0 & 0 & 0 \end{bmatrix} \quad \text{for } \|F\|_2=\varepsilon_1\|a_{33}B-b_{33}A\|_2 \qquad (30)$$

and of the constructed product $Q_2'Q_1'BZ_1Z_2$

$$\tilde{Q}_2'\tilde{Q}_1'(B+E_b)\tilde{Z}_1\tilde{Z}_2 = \begin{bmatrix} \tilde{b}_{11} & \tilde{b}_{12} & \tilde{b}_{13} \\ 0 & \tilde{b}_{22} & \tilde{b}_{23} \\ 0 & 0 & \tilde{b}_{33} \end{bmatrix} \quad \text{for } \|E_b\|_2=\varepsilon_2\|B\|_2 \qquad (31)$$

We prove that there also exists a backward error E_a such that

$$\tilde{Q}_2'\tilde{Q}_1'(A+E_a)\tilde{Z}_1\tilde{Z}_2 = \begin{vmatrix} a_{11} & \tilde{a}_{12} & \tilde{a}_{13} \\ 0 & a_{22} & \tilde{a}_{23} \\ 0 & \tilde{a}_{32} & \tilde{a}_{33} \end{vmatrix} \quad \text{for } \|E_a\|_2=\varepsilon_3\|A\|_2 \qquad (32)$$

An error analysis of $Q_2'Q_1'AZ_1Z_2$ yields

$$\tilde{Q}_2'\tilde{Q}_1'(A+E_c)\tilde{Z}_1\tilde{Z}_2 = \begin{vmatrix} \tilde{a}_{11} & \tilde{a}_{12} & \tilde{a}_{13} \\ \tilde{a}_{21} & \tilde{a}_{22} & \tilde{a}_{23} \\ \tilde{a}_{31} & \tilde{a}_{32} & \tilde{a}_{33} \end{vmatrix} \quad \text{for } \|E_c\|=\varepsilon_4\|A\|_2 \qquad (33)$$

We only have to prove that the elements $\tilde{a}_{i1}$, i=2,3 are ε-small, in order to obtain (32) by putting $\tilde{a}_{i1}=0$, i=2,3. This is easily proved using a similar reasoning to (24)-(26). Here again the assumption $|b_{33}| \geq |a_{33}|$ is crucial in the proof of backward stability. Therefore, in the case $|b_{33}|<|a_{33}|$ the role of B and A have to be interchanged.

Case III: $d_1=1$, $d_2=2$:

This case is dual to the previous case and can be reduced to it by pertransposition (transposition over the anti-diagonal).

Case IV: $d_1=d_2=2$:

A detailed configuration of (11) is then

$$Q'AZ = Q'\begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & a_{43} & a_{44} \end{vmatrix} Z = \begin{vmatrix} \hat{a}_{11} & \hat{a}_{12} & \hat{a}_{13} & \hat{a}_{14} \\ \hat{a}_{21} & \hat{a}_{22} & \hat{a}_{23} & \hat{a}_{24} \\ 0 & 0 & \hat{a}_{33} & \hat{a}_{34} \\ 0 & 0 & \hat{a}_{43} & \hat{a}_{44} \end{vmatrix} = \hat{A} \qquad (34a)$$

$$Q'BZ = Q'\begin{vmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ 0 & b_{22} & b_{23} & b_{24} \\ 0 & 0 & b_{33} & b_{34} \\ 0 & 0 & 0 & b_{44} \end{vmatrix} Z = \begin{bmatrix} \hat{b}_{11} & \hat{b}_{12} & \hat{b}_{13} & \hat{b}_{14} \\ 0 & \hat{b}_{22} & \hat{b}_{23} & \hat{b}_{24} \\ 0 & 0 & \hat{b}_{33} & \hat{b}_{34} \\ 0 & 0 & 0 & \hat{b}_{44} \end{bmatrix} = \hat{B} \qquad (34b)$$

where all the elements are real and B and $\hat{B}$ are invertible. In order to have $\Lambda(B_{22},A_{22})=\Lambda(\hat{B}_{11},\hat{A}_{11})$ the first two columns of Z must span the deflating subspace of $\lambda B-A$ corresponding to $\Lambda(B_{22},A_{22})$ or, equivalently, the two (complex) eigenvectors corresponding to the eigenvalues λ_2 and $\bar{\lambda}_2$ of $\Lambda(B_{22},A_{22})$. Such a Z also satisfies

$$(\lambda_2 I - B^{-1}A)(\bar{\lambda}_2 I - B^{-1}A)Z = \left|\begin{array}{cc|cc} 0 & 0 & & \\ 0 & 0 & & \\ 0 & 0 & * & \\ 0 & 0 & & \end{array}\right| \qquad (35)$$

and could be constructed through (35). Unfortunately, this approach is not recommended from a numerical point of view because of the occurrence of B^{-1} and of the product $(\lambda_2 I - B^{-1}A)(\bar{\lambda}_2 I - B^{-1}A)$. An error analysis of (35) would yield a negligible relative error for this product but not for A and B individually.

A different approach is therefore recommended here, namely the double shift QZ-step. Implicitly this is a double shift QR-step working on the matrix AR^{-1}, but the actual implementation avoids the construction of AB^{-1} and works instead directly on B and A [13]. For our 4x4 pencil (34) the scheme can be implemented economically with Givens rotations:

-Construct $Q_1 \in G_{23}$ and $Q_2 \in G_{12}$ according to the 'double shift technique' and construct $Z_1 \in G_{23}$ and $Z_2 \in G_{12}$ such that $Q_2'Q_1'BZ_1Z_2$ is upper triangular. $Q_2'Q_1'AZ_1Z_2$ and $Q_2'Q_1'BZ_1Z_2$ then look like:

$$\left[\begin{array}{cccc|c} x & x & x & x & \\ x & x & x & x & \\ x_4 & x & x & x & \\ x_3 & x_5 & x & x & \end{array}\right] \qquad \left|\begin{array}{cccc} x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \end{array}\right| \qquad (36)$$

-Construct $Q_3 \in G_{34}$, $Q_4 \in G_{23}$ and $Q_5 \in G_{34}$ annihilating x_3, x_4 and x_5, respectively, in (36). Construct $Z_3 \in G_{34}$, $Z_4 \in G_{23}$ and $Z_5 \in G_{34}$ such that $Q'BZ$, with $Q = Q_1Q_2Q_3Q_4Q_5$ and $Z = Z_1Z_2Z_3Z_4Z_5$, is upper triangular. $Q'AZ$ is now upper Hessenberg and $Q'BZ$ upper triangular. This form is clearly maintained by a QZ-step.

In order to obtain (34) we want moreover that $\hat{a}_{32}=0$ and $\Lambda(\hat{B}_{22},\hat{A}_{22})=\Lambda(B_{11},A_{11})$. According to the properties of the double shift method [13], this will be the case when $\{\lambda_1,\bar{\lambda}_1\}=\Lambda(B_{11},A_{11})$ is chosen to determine the double shift (i.e. Q_1 and Q_2), and if in addition $a_{32}\neq 0$. Since in (34) the latter is not satisfied we first perform a QZ-step with random shift such that $a_{32}\neq 0$, and we then perform a second QZ-step with double shift based on $\{\lambda_1,\bar{\lambda}_1\}$.

The numerical properties of the QZ-step are discussed in [13]. The algorithm is backward stable, but under the presence of rounding errors the element $\hat{a}_{32}$ may not be negligible. Several QZ-steps with double snift $\{\lambda_1,\bar{\lambda}_1\}$ are then performed and $\hat{a}_{32}$ is shown to converge very fast to zero [13]. Only in pathological cases more than one step is required to obtain $|\hat{a}_{32}|\leqslant\varepsilon\Delta$.

Operation count

The combination of a pair of left and right Givens transformations Q_i, Z_1, requires approximately 12n operations (1 operation ❑ 1 addition + 1 multiplication). The number of operations for the different cases is then (for Case IV we assume only 2 QZ-steps are needed):

Case I: 12n

Case II and III: 32n (average)

Case IV: 120n

Since Cases II and III correspond to 2 interchanges of eigenvalues and Case IV to 4 interchanges, we finally have an average of 20n operations for interchanging two adjacent eigenvalues.

When a deflating subspace with specified spectrum $\{\mu_1,\ldots,\mu_\ell\}$ has to be computed and a QZ decomposition is already available, then at most

$\ell.(n-\ell)<n^2/4$ such interchanges are required (namely when all $\mu_i, i=1,..,\ell$ are in the bottom right corner). A reasonable estimate is thus $5n^3$ ops. for computing a specific deflating subspace from a QZ decomposition, while the latter requires approximately $25n^3$ ops..

In order to obtain all possible orderings of eigenvalues in the QZ decomposition, and thus all possible deflating subspaces (if no eigenvalues are repeated), n! such interchanges are required [9]. This is to be expected since it is a combinatorial problem.

III. Riccati equations

In this section we apply the above ideas to the solution of certain Riccati equations arising in linear system theory. We first briefly restate the four problems we will focus on and we refer to the literature for a more complete discussion. We will everywhere assume that the matrices involved are real since this is usually the case in practice. Extensions to the complex case are trivial.

Problem I . Optimal control: continuous time case [10][11][12][24]

Given the stabilizable system

$$\dot{x}(t) = \underset{nn}{A}\, x(t) + \underset{nm}{B}\, u(t) \tag{37}$$

find the control $u(t)=-Kx(t)$ minimizing the functional

$$J = \int_0^\infty [x'(t)\underset{nn}{Q}x(t) + u'(t)\underset{mm}{R}u(t)]\, dt \tag{38}$$

where (A,Q) is detectable, $Q \geqslant 0$ and $R \geqslant 0$.

When R is invertible this problem reduces to the computation of the unique nonnegative definite solution P of the algebraic Riccati equation

$$Q + A'P + PA - PBR^{-1}B'P = 0 \tag{39}$$

K .is then equal to R-1B'P. Equivalently [12], one can compute the invariant subspace X_s of the matrix

$$H = \begin{bmatrix} A & -BR^{-1}B' \\ -Q & -A' \end{bmatrix} \tag{40}$$

where $\Lambda(H)|_{X_s}$ contains all the stable eigenvalues (i.e. $Re(\lambda)<0$) of H. If $\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$ is a basis for this subspace then $P=X_2X_1^{-1}$.

Problem II. Optimal control problem: discrete time case [5][7][15]

Given the stabilizable system

$$x_{i+1} = F_{nn}x_i + G_{nm}u_i \tag{41}$$

find the control $u_i = -Kx_i$ minimizing the functional

$$J = \sum_{i=0}^{\infty} [x_i'Q_{nn}x_i + u_i'R_{mm}u_i] \tag{42}$$

where (F,Q) is detectable, $Q \geq 0$ and $R > 0$.

When R is invertible [7], this problem can again be converted to the computation of the unique nonnegative definite solution P of the (discrete time) algebraic Riccati equation :

$$P = F'PF - F'PG(R+G'PG)^{-1}G'PF + Q \tag{43}$$

K is then equal to R-1G'P. This is also equivalent to solving for the 'stable' deflating subspace X_s of the pencil [5][15]

$$\lambda \begin{bmatrix} I & GR^{-1}G' \\ 0 & F' \end{bmatrix} - \begin{bmatrix} F & 0 \\ -Q & I \end{bmatrix} \tag{44}$$

where this time the stable eigenvalues are those inside the unit circle. If $\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$ is a basis for X_s then $P = X_2X_1^{-1}$.

Problem III. Spectral factorization: continuous time case [2]

Given an mxm 'positive real' rational matrix Z(s), i.e.

$$Z(s)+Z^*(s) \geq 0 \text{ in } Re(s)>0 \tag{45}$$

find a 'spectral factorization'

$$Z(s)+Z'(-s) = R(s).R'(-s) \tag{46}$$

where R(s) has only stable poles (i.e. Re(s)<0).

When Z(s) is given by a minimal realization $C(sI_n-A)^{-1}B+D$ and (D+D') is invertible, then this problem reduces to the computation of the unique positive definite solution of the algebraic Riccati equation [2]:

$$B(D+D')^{-1}B'+P[A-B(D+D')^{-1}C]'+[A-B(D+D')^{-1}C]P+PC'(D+D')^{-1}CP=0 \tag{47}$$

This is again equivalent to the computation of the stable invariant subspace X_s of the matrix

$$H = \begin{bmatrix} A-B(D+D')^{-1}C & B(D+D')^{-1}B' \\ -C'(D+D')^{-1}C & -[A-B(D+D')^{-1}C]' \end{bmatrix}_I \tag{48}$$

Problem TV. Spectral factorization: discrete time case [1][4]

Given an mxm 'positive real' discrete time matrix 7(z), i.e.

$$Z(z)+Z^*(z) \geq 0 \quad \text{for } |z|>1 \tag{49}$$

find a spectral factorization

$$Z(z)+Z'(z^{-1})=R(z).R'(z^{-1}) \tag{50}$$

where R(z) has only stable poles (i.e. poles inside the unit circle). Again, wnen Z(z) is given by a minimal realization $H(zI_n-F)^{-1}G+J$ and (J+J') is invertible, the problem can be reduced to the computation of the unique positive definite solution P of the (discrete time) Riccati equation [4]:

$$P = FPF' + (G-FPH')(J+J'-HPH')^{-1}(G'-HPF') \tag{51}$$

In analogy to (43)(44), one can prove that this is equivalent to computing the stable deflating subspace X_s of

$$\lambda \begin{bmatrix} I & -G(J+J')^{-1}G' \\ 0 & F'-H'(J+J')^{-1}G' \end{bmatrix} - \begin{bmatrix} F-G(J+J')^{-1}H & 0 \\ -H'(J+J')^{-1}H & I \end{bmatrix} \tag{52}$$

This, however, was not found in the literature.

Note that in order to be able to write down the Riccati equations, certain matrices need to be invertible. This also holds for the equivalent SEP's and GEP's, since they are derived from the Riccati equations. Yet, if the matrices to be inverted happen to be badly conditioned, each of these approaches may encounter serious numerical difficulties when computing these inverses. We now present a way to circumvent this by an embedding trick.

Let D be invertible in the pencil

$$\begin{matrix} \begin{bmatrix} \lambda E-A & B \\ \lambda F-C & D \end{bmatrix} & \begin{matrix} \}p \\ \}m \end{matrix} \\ \underbrace{\qquad\qquad}_{P} \quad \underbrace{\qquad}_{m} \end{matrix} \tag{53}$$

then

$$\begin{bmatrix} I & -BD^{-1} \\ 0 & I \end{bmatrix} \cdot \begin{bmatrix} \lambda E-A & B \\ \lambda F-C & D \end{bmatrix} = \begin{bmatrix} \lambda(E-BD^{-1}F)-(A-BD^{-1}C) & 0 \\ \lambda F-C & D \end{bmatrix} \tag{54}$$

Let U be an orthogonal transformation reducing $\begin{bmatrix} B \\ D \end{bmatrix}$ to $\begin{bmatrix} 0 \\ \tilde{D} \end{bmatrix}$ with $\tilde{D}$ mxm and invertible. Partition U conformably with (53), then we have

$$\begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix} \cdot \begin{bmatrix} \lambda E-A & B \\ \lambda F-C & D \end{bmatrix} = \begin{bmatrix} \lambda\tilde{E}-\tilde{A} & 0 \\ * & \tilde{D} \end{bmatrix} \tag{55}$$

Since the rows of $[U_{11}\,|\,U_{12}]$ and $[\,I\,|-BD^{-1}]$ both are a basis for the left null space of $\begin{bmatrix} B \\ D \end{bmatrix}$, they are related by an invertible row transformation which clearly must be U_{11}:

$$U_{11}[\,I\,|-BD^{-1}] = [U_{11}\,|\,U_{12}] \tag{56}$$

From (54) and (55) it then follows that

$$U_{11}[\lambda(E-BD^{-1}F)-(A-BD^{-1}C)] = \lambda\tilde{E}-\tilde{A} \tag{57}$$

Therefore, the deflating subspaces of $\lambda\tilde{E}-\tilde{A}$ and of $\lambda(E-BD^{-1}F)-(A-BD^{-1}C)$ are the same. According to (7), deflating subspaces of a regular pencil are indeed not affected by an invertible row transformation on the pencil. This trick was originally applied in [22] (with E=I and F=0) for developing a stable way to compute the deflating subspaces of $\lambda I-(A-BD^{-1}C)$ or, in other words, the invariant subspaces of $A-BD^{-1}C$. This can now be applied to the above four problems. In each of them the pencil (53) takes the form (we always have p=2n)

Problem I :

$$\lambda\begin{bmatrix} & & \\ I & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} A & 0 & B \\ -Q & -A' & 0 \\ 0 & B' & R \end{bmatrix} \tag{58}$$

Problem II :

$$\lambda\begin{bmatrix} I & 0 & 0 \\ 0 & F' & 0 \\ 0 & G' & 0 \end{bmatrix} - \begin{bmatrix} F & 0 & -G \\ -Q & T & 0 \\ 0 & 0 & R \end{bmatrix} \tag{59}$$

Problem III :

$$\lambda\begin{bmatrix} & & \\ & & \\ I & II & 0 \end{bmatrix} \quad \begin{bmatrix} A & 0 & B \\ 0 & -A' & C' \\ C & -B' & D+D' \end{bmatrix} \tag{60}$$

Problem IV :

$$\lambda \begin{bmatrix} I & 0 & 0 \\ 0 & F' & 0 \\ 0 & G' & 0 \end{bmatrix} - \begin{bmatrix} F & 0 & -G \\ 0 & I & -H' \\ H & 0 & -(J+J') \end{bmatrix} \qquad (61)$$

For each of these pencils a $2n \times 2n$ pencil $\lambda\tilde{E}-\tilde{A}$ can thus be derived via (55) and its stable deflating subspace X_s is the one required in the above four problems. This procedure does not involve the inversion of a possibly ill-conditioned matrix. Only orthogonal transformations are used as well in the construction of $\lambda\tilde{E}-\tilde{A}$ as in the computation of the deflating subspace X_s. This guarentees the numerical stability of the method. Unfortunately this is not completely satisfactory yet, since the performed errors do not necessarily respect the structure of the pencils (58)-(61). A (unsuccesful) attempt to restrict the orthogonal transformations to those respecting the structure of the matrices they act upon, can be found in the literature for Problems I and III but in the formulation (40) and (48), respectively [14].

A important remark here is that in the new formulation (58)-(61) no inverses occur anymore and that perhaps this new formulation also gives the correct answer when these inverses do not exist. This would follow from limiting arguments if both the exact solution of the problem and the computed solution from the GEP's (58)-(61) are continuous. This is true for the eigenvalue problem if the spectrum $\Lambda(\tilde{E},\tilde{A})|_{X_s}$ is separated from the rest of the spectrum of $\lambda\tilde{E}-\tilde{A}$ [20], and this holds under the assumptions made in each problem (stabilizability, detectability, positive realness). The continuity of the solution $R(s)$ of Problem III is discussed in [2 ,p.243]. It also holds for the more general 'minimal factorization problem' [3] for which the above embedding trick was

originally derived [22]. It is therefore reasonable to assume that it also holds for the other three Problems. This is still under current investigation.

During the elaboration of this research, the author's attention was drawn to the work of A. Emami-Naeini and G. Franklin [6]. Via an independent approach they arrive to the same form (59). No proof is provided, though, that the method also works for singular R.

IV. Computations

In this section we give two examples illustrating the reordering of eigenvalues in order to compute a certain deflating subspace with prescribed spectrum. We use a PDP11-34 computer with double precision. The machine precision is then $\varepsilon = 1.5\ 10^{-17}$. Two routines are used for the reordering of the Schur form (see the Appendix for a listing). EXCHQZ exchanges two adjacent blocks in a real Schur form and ORDER uses this routine to reorder all the eigenvalues inside the unit circle to the top or bottom of the real Schur form, depending on the value of a parameter IFIRST. This last routine is easily adapted for any region which is symmetric with respect to the real axis. This condition is necessary because the pencils considered are real and complex conjugate eigenvalues need thus to stay together in the real Schur form.

Example I

$$A-\lambda B = \left|\begin{array}{cccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1.3 & .2 & 4 & 6 & 0 & 0 & 0 \\ 0 & -.2 & .3 & 0 & 0 & 0 & .5 & 0 \\ 0 & 0 & 0 & .5 & 0 & 2 & 0 & 3 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 2.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & -10 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{array}\right| - \lambda \left|\begin{array}{cccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right| \qquad (62)$$

The first four eigenvalues $\{0, .3-j.2, .3+j.2, .5\}$ are inside the unit circle. The last four ones $\{\infty, 4-j5, 4+j5, 2\}$ are outside the unit circle. Calling ORDER with IFIRST=1 interchanges the order of these sets of eigenvalues. The first four columns of the transformation Z required for this, then span the unstable subspace X_U of $A-\lambda B$:

```
 0.0000000000000000d 00  0.0000000000000000d 00  0.0000000000000000d 00  0.4472135954999579d 00
 0.0000000000000000d 00  0.1074176896329408d 00  0.4536659636818238d-01  0.0000000000000000d 00
 0.0000000000000000d 00 -0.6787906375520325d-01 -0.1235432403982217d 00  0.0000000000000000d 00
 0.0000000000000000d 00  0.1029985626780152d 00 -0.1992532259759830d 00  0.0000000000000000d 00
 0.1000000000000000d 01  0.0000000000000000d 00  0.0000000000000000d 00  0.0000000000000000d 00
 0.0000000000000000d 00 -0.1485334378807009d 00 -0.9610814937405690d 00  0.0000000000000000d 00
 0.0000000000000000d 00 -0.9752861049841944d 00  0.1389224422842769d 00  0.0000000000000000d 00
 0.0000000000000000d 00  0.0000000000000000d 00  0.0000000000000000d 00  0.8944271909999159d 00
```

When again calling ORDER but now with IFIRST=-1 we retrieve the ordering of $A-\lambda B$ and the four first columns of the updated Z look like

```
 0.1000000000000000d 01  0.0000000000000000d 00  0.0000000000000000d 00  0.0000000000000000d 00
 0.0000000000000000d 00 -0.9682688602071642d 00 -0.2499108127975237d 00  0.3620467925763759d-16
 0.0000000000000000d 00 -0.2499108127975237d 00  0.9682688602071642d 00 -0.1119431427711831d-15
 0.0000000000000000d 00  0.2168404344971009d-17  0.1154675313697062d-15  0.9999999999999999d 00
 0.0000000000000000d 00 -0.3134519117986896d-17  0.1040834085586084d-16  0.4491253335510017d-16
 0.0000000000000000d 00  0.1040834085586084d-16  0.1543903893619358d-15 -0.5898059818321144d-16
 0.0000000000000000d 00 -0.3469446951953614d-17 -0.6357765153945499d-15  0.0000000000000000d 00
 0.0000000000000000d 00  0.0000000000000000d 00  0.0000000000000000d 00  0.0000000000000000d 00
```

This is ε-close to the real stable deflating subspace X_s of $A-\lambda B$, which is spanned by $\begin{bmatrix} I_4 \\ 0_4 \end{bmatrix}$. This result is to be expected because of the numerical stability of our method and because the space X_s of $A-\lambda B$ is well conditioned. When the gap between the spectrums $\Lambda(B,A)|_{X_s}$ and $\Lambda(B,A)|_{X_u}$ is large, both spaces X_s and X_u are indeed well conditioned (see [20]).

Example IX

Consider Problem II with

$$F = \begin{bmatrix} 2 & -1 \\ 1 & 0 \end{bmatrix}, \quad G = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad Q = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = \begin{bmatrix} 0 \end{bmatrix} \qquad (63)$$

The pencil (59) then looks like

$$\lambda \begin{vmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{vmatrix} - \begin{vmatrix} 2 & -1 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix} \tag{64}$$

An orthogonal row transformation can then be constructed in order to construct a deflated pencil $\lambda\tilde{E}-\tilde{A}$ following (55):

$$\lambda \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{65}$$

The QZ-algorithm permutes the two Last columns of (65) to obtain the real Schur form :

$$\lambda \begin{vmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{vmatrix} - \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{vmatrix} \tag{66}$$

which displays the eigenvalues $\{\infty, \infty, 0, 0\}$. Tn order to obtain the stable subspace X_s of $\lambda\tilde{E}-\tilde{A}$ we reorder these eigenvalues and obtain as a basis for X_s:

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 0 & -\sqrt{2}/2 \\ -\sqrt{2}/2 & 0 \\ 0 & -\sqrt{2}/2 \\ -\sqrt{2}/2 & 0 \end{bmatrix} + O(\varepsilon) \tag{67}$$

We then find, up to machine accuracy, the answer P=I .One can check that

this is the correct answer to Problem IT, by using another method [7]. This example illustrates that the embedding trick gives a correct result even when R is singular. Moreover, the problem is perfectly well conditioned as well for the construction of $\lambda\tilde{E}-\tilde{A}$, as for the computation of X_s and P.

We finally want to draw the attention to the fact that the number of operations required for the construction of $\lambda\tilde{E}-\tilde{A}$ from the pencils (58)-(61), is comparable to the amount of work required to construct the pencils (40)(44)(48)(52). From then on the new approach takes the same amount of computations for Problems IT and IV and only slightly more (less than the double) for Problems I and III. The stability of the method and its better conditioning therefore make this new approach particularly attractive.

Acknowledgements

I want to thank A. Emami-Naeini, G. Franklin and A. Laub for drawing my attention to this problem and for several helpful1 discussions. A. Emami-Naeini also suggested Example II.

Appendix

```
      subroutine order(a,b,z,nmax,n,eps,fail,ifirst,ind)
      implicit real*8 (a-h,o-z)
      logical fail
      dimension a(nmax,n),b(nmax,n),z(nmax,n),ind(1)
c Given the upper triangular matrix b and upper Hessenberg matrix a
c with 1x1 or 2x2 diagonal. blocks, this routine reorders the diagonal
c blocks along with their generalized eigenvalues by constructing
c equivalence transformations qt and zt. After reordering, the eigenvalues
c outside the unit circle appear first if ifirst=1 and last if ifirst=-1.
c Order requires the subroutine exchqz. The parameters in the calling
c sequence are (starred parameters are altered by the subroutine) :
c   *a,*b    the matrix pair whose blocks are to be reordered.
c   *z       upon return this array is multiplied by the column
c            transformation zt.
c    nmax    the first dimension of a, b and z
c    n       the order of a, b and z
c    eps     the required absolute accuracy of the result
c   *fail    a logical variable which is false on a normal return,
c            true otherwise (when exchqz fails)
c    ifirst  an integer equal to +1 or -1 (see above)
c   *ind     an integer working array of dimension at least n
      fail=.true.
      num=0
      l=0
      ls=1
c*** determine size and stability of blocks ***
 10   l=l+ls
      if(l.gt.n) go to 50
      is=-ifirst
      l1=l+1
      if(l1.gt.n) go to 20
      if(a(l1,l).eq.0.d0) go to 20
c* 2x2 block *
      ls=2
      if(dabs(a(l,l)*a(l1,l1)-a(l1,l)*a(l,l1)).lt.dabs(b(l,l)*b(l1,l1)))
     * is=ifirst
      go to 30
c* 1x1  block *
 20   ls=1
      if(dabs(a(l,l)).lt.dabs(b(l,l)))is=ifirst
 30   num=num+1
      ind(num)=ls*is
      go to 10
c*** reorder blocks ***
 50   l2=1
      i=0
 60   i=i+1
      if(ind(i).gt.0) go to 70
      l2=l2-ind(i)
      go to 60
 70   k=i
 80   l2=l2+ind(k)
 85   k=k+1
      if(k.gt.num) go to 100
```

```
      if(ind(k).gt.0) go to 80
c* interchange block k before block i *
      istep=k-i
      ls2=-ind(k)
      l=l2
        do 90 ii=1,istep
        ifirst=k-ii
        ls1=ind(ifirst)
        l=l-ls1
        call exchqz (a,b,z,nmax,n,l,ls1,ls2,eps,fail)
        if (fail) return
 90     ind(ifirst+1)=ind(ifirst)
      ind(i)=-ls2
      i=i+1
      l2=l2+ls2
      go to 85
 100  fail=.false.
      return
      end

      subroutine exchqz(a,b,z,nmax,n,l,ls1,ls2,eps,fail)
      implicit real*8 (a-h,o-z)
      dimension a(nmax,n),b(nmax,n),z(nmax,n),u(3,3)
      logical fail,altb
c Given the upper triangular matrix b and upper Hessenberg matrix a
c with consecutive ls1xls1 and ls2xls2 diagonal blocks (ls1,ls2.le.2)
c starting at row/column 1, exchqz produces equivalence transforma-
c tions qt and zt that exchange the blocks along with their generalized
c eigenvalues. Exchqz requires the subroutines rotc, rotr and giv.
c The parameters in the calling sequence are (starred parameters are
c altered by the subroutine):
c   *a,*b   the matrix pair whose blocks are to be interchanged
c   *z      upon return this array is multiplied by the column
c           transformation zt.
c    nmax   the first dimension of a, b and z
c    n      the order of a, b and z
c    l      the position of the blocks
c    ls1    the size of the first block
c    ls2    the size of the second block
c    eps    the required absolute accuracy of the result
c   *fail   a logical variable which is false on a normal return,
c           true otherwise (when a(l+ls2,l+ls2-1) cannot be assumed
c           zero)
      fail=.false.
      l1=l+1
      ll=ls1+ls2
      if (ll.gt.2) go to 50
c*** interchange 1x1 and 1x1 blocks ***
      f=dmax1(dabs(a(l1,l1)),dabs(b(l1,l1)))
      altb=.true.
      if(dabs(a(l1,l1)).ge.f) altb=.false.
      sa=a(l1,l1)/f
      sb=b(l1,l1)/f
      f=sa*b(l,l)-sb*a(l,l)
c* compute z *
      g=sa*b(l,l1)-sb*a(l,l1)
```

```
      call giv(f,g,d,e)
      call rotc(a,nmax,n,l,l1,1,l1,d,e)
      call rotc(b,nmax,n,l,l1,1,l1,d,e)
      call rotc(z,nmax,n,l,l1,1,n,d,e)
c* compute q *
      if (altb) call giv(b(l,l),b(l1,l),d,e)
      if (.not.altb) call giv(a(l,l),a(l1,l),d,e)
      call rotr(a,nmax,n,l,l1,l,n,d,e)
      call rotr(b,nmax,n,l,l1,l,n,d,e)
      a(l1,l)=0.d0
      b(l1,l)=0.d0
      return
c*** interchange 1x1 and 2x2 blocks ***
 50   l2=l+2
      if(lsl.eq.2) go to 100
      g=dmax1(dabs(a(l,l)),dabs(b(l,l)))
      altb=.true.
      if(dabs(a(l,l)).lt.g) go to 60
      altb=.false.
      call giv(a(l1,l1),a(l2,l1),d,e)
      call rotr(a,nmax,n,l1,l2,l1,n,d,e)
      call rotr(b,nmax,n,l1,l2,l1,n,d,e)
c** compute q and z **
 60   sa=a(l,l)/g
      sb=b(l,l)/g
      do 80 j=1,2
        lj=l+j
        do 80 i=1,3
          li=l+i-1
 80       u(i,j)=sa*b(li,lj)-sb*a(li,lj)
      call giv(u(3,1),u(3,2),d,e)
      call rotc(u,3,3,1,2,1,3,d,e)
c* ql *
      call giv(u(1,1),u(2,1),d,e)
      u(2,2)=-u(1,2)*e+u(2,2)*d
      call rotr(a,nmax,n,l,l1,l,n,d,e)
      call rotr(b,nmax,n,l,l1,l,n,d,e)
c* z1 *
      if (altb) call giv(b(l1,l),b(l1,l1),d,e)
      if (.not.altb) call giv(a(l1,l),a(l1,l1),d,e)
      call rotc(a,nmax,n,l,l1,1,l2,d,e)
      call rotc(b,nmax,n,l,l1,1,l2,d,e)
      call rotc(z,nmax,n,l,l1,1,n,d,e)
c* q2 *
      call giv(u(2,2),u(3,2),d,e)
      call rotr(a,nmax,n,l1,l2,l,n,d,e)
      call rotr(b,nmax,n,l1,l2,l,n,d,e)
c* z2 *
      if (altb) call giv (b(l2,l1),b(l2,l2),d,e)
      if (.not.altb) call giv(a(l2,l1),a(l2,l2),d,e)
      call rotc(a,nmax,n,l1,l2,1,l2,d,e)
      call rotc(b,nmax,n,l1,l2,1,l2,d,e)
      call rotc(z,nmax,n,l1,l2,1,n,d,e)
      if (altb) go to 90
      call giv(b(l,l),b(l1,l),d,e)
      call rotr(a,nmax,n,l,l1,l,n,d,e)
```

```
      call rotr(b,nmax,n,1,l1,1,n,d,e)
 90   a(l2,1)=0.d0
      a(l2,l1)=0.d0
      b(l1,1)=0.d0
      b(l2,1)=0.d0
      b(l2,l1)=0.d0
      return
c*** interchange 3x3 and 1x1 blocks ***
 100  if(ls2.eq.2) go to 150
      g=dmax1(dabs(a(l2,l2)),dabs(b(l2,l2)))
      altb=. true.
      if(dabs(a(l2,l2)).lt.g) go to 120
      altb=. false.
      call giv(a(1,1),a(l1,1),d,e)
      call rotr(a,nmax,n,1,l1,1,n,d,e)
      call rotr(b,nmax,n,1,l1,1,n,d,e)
c** compute q and z **
 120  sa=a(l2,l2)/g
      sb=b(l2,l2)/g
      do 130 i=1,2
        li=l+i-1
        do 130 j=1,3
          lj=l+j-1
 130      u(i,j)=sa*b(li,lj)-sb*a(li,lj)
      call giv (u(1,1),u(2,1),d,e)
      call rotr (u,3,3,1,2,1,3,d,e)
c* z1 *
      call giv (u(2,2),u(2,3),d,e)
      u(1,2)=u(1,2)*e-u(1,3)*d
      call rotc (a,nmax,n,l1,l2,1,l2,d,e)
      call rotc (b,nmax,n,l1,l2,1,l2,d,e)
      call rotc (z,nmax,n,l1,l2,1,n,d,e)
c* ql *
      if (altb) call giv (b(l1,l1),b(l2,l1),d,e)
      if (.not.altb) call giv (a(l1,l1),a(l2,l1),d,e)
      call rotr (a,nmax,n,l1,l2,1,n,d,e)
      call rotr (b,nmax,n,l1,l2,1,n,d,e)
c* z2 *
      call giv (u(1,1),u(1,2),d,e)
      call rotc (a,nmax,n,1,l1,1,l2,d,e)
      call rotc (b,nmax,n,1,l1,1,l2,d,e)
      call rotc (z,nmax,n,1,l1,1,n,d,e)
c* q2 *
      if(altb) call giv(b(1,1),b(l1,1),d,e)
      if(.not.altb) call giv(a(1,1),a(l1,1),d,e)
      call rotr(a,nmax,n,1,l1,1,n,d,e)
      call rotr(b,nmax,n,1,l1,1,n,d,e)
      if(altb) go to 140
      call giv(b(l1,l1),b(l2,l1),d,e)
      call rotr(a,nmax,n,l1,l2,l1,n,d,e)
      call rotr(b,nmax,n,l1,l2,l1,n,d,e)
 140  a(l1 ,1)=0.d0
      a(l2,1)=0.d0
      b(l1,1)=0.d0
      b(l2,1)=0.d0
      b(l2,l1)=0.d0
```

```
      return
c*** interchange 2x2 and 2x2 blocks ***
 150  l3=l+3
      ammbmm=a(1,1)/b(1,1)
      anmbmm=a(l1,1)/b(1,1)
      amnbnn=a(1,l1)/b(l1,l1)
      annbnn=a(l1,l1)/b(l1,l1)
      bmnbnn=b(1,l1)/b(l1,l1)
      do 180 it1=1,3
        u(1,1)=1.d0
        u(2,1)=1.d0
        u(3,1)=1.d0
        do 180 it2=1,10
c* q1,q2 *
          call giv(u(2,1),u(3,1),d,e)
          call rotr(a,nmax,n,l1,l2,1,n,d,e)
          call rotr(b,nmax,n,l1,l2,l1,n,d,e)
          u(2,1)=d*u(2,1)+e*u(3,1)
          call giv(u(1,1),u(2,1),d,e)
          call rotr(a,nmax,n,1,l1,1,n,d,e)
          call rotr(b,nmax,n,1,l1,1,n,d,e)
c* z1,z2 *
          call giv(b(l2,l1),b(l2,l2),d,e)
          call rotc(a,nmax,n,l1,l2,1,l3,d,e)
          call rotc(b,nmax,n,l1,l2,1,l2,d,e)
          call rotc(z,nmax,n,l1,l2,1,n,d,e)
          call giv(b(l1,1),b(l1,l1),d,e)
          call rotc(a,nmax,n,1,l1,1,l3,d,e)
          call rotc(b,nmax,n,1,l1,1,l1,d,e)
          call rotc(z,nmax,n,1,l1,1,n,d,e)
c* q3,z3,q4,z4,q5,z5 *
          call giv(a(l2,1),a(l3,1),d,e)
          call rotr(a,nmax,n,l2,l3,1,n,d,e)
          call rotr(b,nmax,n,l2,l3,l2,n,d,e)
          call giv(b(l3,l2),b(l3,l3),d,e)
          call rotc(a,nmax,n,l2,l3,1,l3,d,e)
          call rotc(b,nmax,n,l2,l3,1,l3,d,e)
          call rotc(z,nmax,n,l2,l3,1,n,d,e)
          call giv(a(l1,1),a(l2,1),d,e)
          call rotr(a,nmax,n,l1,l2,1,n,d,e)
          call rotr(b,nmax,n,l1,l2,l1,n,d,e)
          call giv(b(l2,l1),b(l2,l2),d,e)
          call rotc(a,nmax,n,l1,l2,1,l3,d,e)
          call rotc(b,nmax,n,l1,l2,1,l2,d,e)
          call rotc(z,nmax,n,l1,l2,1,n,d,e)
          call giv(a(l2,l1),a(l3,l1),d,e)
          call rotr(a,nmax,n,l2,l3,l1,n,d,e)
          call rotr(b,nmax,n,l2,l3,l2,n,d,e)
          call giv(b(l3,l2),b(l3,l3),d,e)
          call rotc(a,nmax,n,l2,l3,1,l3,d,e)
          call rotc(b,nmax,n,l2,l3,1,l3,d,e)
          call rotc(z,nmax,n,l2,l3,1,n,d,e)
c* test of convergence *
          if(dabs(a(l2,l1)).le.eps) go to 190
          a11b11=a(1,1)/b(1,1)
          a12b22=a(1,l1)/b(l1,l1)
```

```
          a21b11=a(11,1)/b(1,1)
          a22b22=a(11,11)/b(11,11)
          b12b22=b(1,11)/b(11,11)
          u(1,1)=((ammbmm-a11b11)*(annbnn-a11b11)-amnbnn*anmbmm
     &          +anmbmm*bmnbnn*a11b11)/a21b11+a12b22-a11b11*b12b22
          u(2,1)=(a22b22-a11b11)-a21b11*b12b22-(ammbmm-a11b11)
     &          -(annbnn-a11b11)+anmbmm*bmnbnn
 180      u(3,1)=a(12,11)/b(11,11)
      fail=.true.
      return
 190  a(12,1)=0.d0
      a(12,11)=0.d0
      a(13,1)=0.d0
      a(13,11)=0.d0
      b(11,1)=0.d0
      b(12,1)=0.d0
      b(12,11)=0.d0
      b(13,1)=0.d0
      b(13,11)=0.d0
      b(13,12)=0.d0
      return
      end

      subroutine rotc(h,nmax,n,11,12,m1,m2,d,e)
      real*8 h(nmax,n),d,e,s
c This routine performs the Givens rotation | e  d | on columns 11,12
c of h(nmax,n), this from rows ml to m2.     |-d  e |
      do 10 i=m1,m2
        s=d*h(i,11)+e*h(i,12)
        h(i,11)=e*h(i,11)-d*h(i,12)
 10     h(i,12)=s
      return
      end

      subroutine rotr(h,nmax,n,11,12,m1,m2,d,e)
      real*8 h(nmax,n),d,e,s
c This routine performs the Givens rotation | d  e | on rows 11,12
c of h(nmax,n), this from columns ml to m2.  I-e  d
      do 10 j=m1,m2
        s=d*h(11,j)+e*h(12,j)
        h(12,j)=-e*h(11,j)+d*h(12,j)
 10     h(11, j)=s
      return
      end

      subroutine giv(a,b,d,e)
      implicit real*8 (a-e)
c This routine computes d=a/sqrt(a*a+b*b) and e=b/sqrt(a*a+b*b)
      c=dmax1(dabs(a),dabs(b))
      d=a/c
      e=b/c
      c=dsqrt(d*d+e*e)
      d=d/c
      e=e/c
      return
      end
```

References

[1] B. Anderson, J. Moore, "Optimal filtering", Prentice Hall, New Jersey, 1979

[2] B. Anderson, S. Vongpanitlerd, "Network analysis and synthesis. A modern systems approach", Prentice Hall, New Jersey, 1972

[3] H. Bart, I. Gohberg, M. Kaashoek, P. Van Dooren, "Factorizations of transfer functions", to appear in SIAM J. Contr. & Opt.

[4] M. Denham, "On the factorization of discrete-time rational spectral density matrices.", IEEE Trans. Aut. Contr., Vol AC-20, pp.535-537, Aug. 1975

[5] A. Emami-Naeini, G. Franklin, "Design of steady state quadratic loss optimal digital controls for systems with a singular system matrix", in Proceedings 13th Asilomar Conf. Circ. Syst. & Comp., pp.370-374, Nov. 1979

[6] A. Emami-Naeini, G. Franklin, "Deadbeat control & tracking", in preparation

[7] G. Franklin, J. Powell, "Digital control of dynamic systems", Addison-Wesley, New york, 1979.

[8] G. Golub, J. Wilkinson, "Ill-conditioned eigensystems and the computation of the Jordan canonical form", SIAM Rev., Vol. 18, pp.578-619, Oct. 1976

[9] S. Johnson, "Generation of permutations by adjacent transposition", Math. Comp., Vol. **17**, No 83, pp.282-285, 1963

[10] T. Kailath, "Linear systems", Prentice Hall, New Jersey, 1980

[11] H. Kwakernaak, R. Sivan, "Linear optimal control systems", Wiley-Tnterscience, 1972

[12] A. Laub, "A Schur method for solving algebraic Riccati equations", IEEE Trans. Aut. Contr., Vol AC-34, pp.913-921, Dec. **1979**

[13] C. Moler, G. Stewart, "An algorithm for generalized matrix eigen-value problem", SIAM J. Num. Anal., Vol. 10, pp.241-256, April 1973

[14] C. Paige, C. Van Loan, "A Hamiltonian-Schur decomposition", Intern. Rept., Dept. Comp. Sc., Cornell Univ., New York, 1979

[15] T. Pappas, A. Laub, N. Sandell Jr., "On the numerical solution of the discrete time algebraic Riccati equation'", to appear in IEEE Trans. Aut. Contr.

[16] G. Peters, J. Wilkinson, "$Ax=\lambda Bx$ and the generalized eigenproblem", SIAM J. Num. Anal., Vol. 7, pp.479-492, Dec. 1970

[17] A. Ruhe, "An algorithm for numerical determination of the structure of a general matrix", BIT, Vol. 10, pp.196-216, 1970

[18] G. Stewart, "On tne sensitivity of the eigenvalue problem $Ax=\lambda Bx$", SIAM J. Num. Anal., Vol. 9, pp.669-686, Dec. 1972

[19] G. Stewart, "Introduction to matrix computation", Acad. Press, New York, 1973

[20] G. Stewart, "Error and perturbation bounds for subspaces associated with certain eigenvalue problems", SIAM Rev., Vol.15, pp.727-764, Oct. 1973

[21] G. Stewart, "Algorithm 506: HQR3 and EXCHNG. Fortran subroutines for calculating and ordering the eigenvalues of a real upper Hessenberg matrix", ACM TOMS, Vol.2, pp.275-280, Sept. 1976

[22] P. Van Dooren, "Tne generalized eigenstructure problem in linear system theory", subm. to IEEE Trans. Aut. Contr.

[23] J. Wilkinson, "Tne algebraic eigenvalue problem", Oxford University Press, London, 1965

[24] M . Wonna"On a matrix Ficcati equation of stochastic control", SIAM J. Contr., Vol.6, pp.681-697, Nov. 1968