A numerical implementation of the algorithm of Kung and Lin for AAK model reduction

Johan Decorte
Adhemar Bultheel
Marc Van Barel

Report TW 81, July 1986
Typeset in LaTeX 2007



Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

A numerical implementation of the algorithm of Kung and Lin for AAK model reduction

Johan Decorte
Adhemar Bultheel
Marc Van Barel

Report TW 81, July 1986
Typeset in LaTeX 2007

Department of Computer Science, K.U.Leuven

Abstract

A numerical implementation of the method of Kung and Lin is described. It is a method to find the minimal degree approximant, given an approximation error bound for the Hankel-norm approximation criterion for multivariable systems. After a brief review of the theory, all non standard computations are described in detail. Some comments on its numerical performance are given, guided by two illustrative examples.

A numerical implementation of the algorithm of Kung and Lin for AAK model reduction*

Johan Decorte
Adhemar Bultheel
Marc Van Barel

July 1986
Typeset in LaTeX 2007

Abstract

A numerical implementation of the method of Kung and Lin is described. It is a method to find the minimal degree approximant, given an approximation error bound for the Hankel-norm approximation criterion for multivariable systems. After a brief review of the theory, all non standard computations are described in detail. Some comments on its numerical performance are given, guided by two illustrative examples.

1 Introduction

Consider a discrete linear causal system

$$x_{k+1} = Ax_k + Bu_k y_k = Cx_k + Du_k.$$

$u_k \in \mathbb{C}^m$, $x_k \in \mathbb{C}^n$, $y_k \in \mathbb{C}^r$, $A \in \mathbb{C}^{n\times n}$, $B \in \mathbb{C}^{n\times m}$, $C \in \mathbb{C}^{r\times n}$ and $D \in \mathbb{C}^{r\times m}$.
m is the number of inputs and r is the number of outputs. n is the number of states or the dimension of the state space. If this number is minimal, it is called the McMillan degree of the system.

$$H(z) = C(zI - A)^{-1}B + D$$

is the transfer function of the system. The z-transforms of the inputs and outputs are related by

$$Y(z) = H(z)U(z).$$

The problem is to approximate this system by another system with a lower McMillan degree, say $k < n$. Two possible problems can be considered.

*This paper is presented at the SIAM conference on linear algebra in signals, systems and control, August 12-14, 1986, Boston, Massachusetts.

1. Minimal Norm Approximation (MNA)
   Given: $H(z)$ of McMillan degree n and $k < n$.
   Find: an approximant $\tilde{H}(z)$ of $H(z)$ of McMillan degree $\leq k$ such that $\|H(z)-\tilde{H}(z)\|_H$ is minimal.

2. Minimal Degree Approximation (MDA)
   Given: $H(z)$ of McMillan degree n and $\rho \in \mathbb{R}$
   Find: an approximant $\tilde{H}(z)$ of $H(z)$ with $\|H(z)-\tilde{H}(z)\|_H \leq \rho$ and of McMillan degree as low as possible.

The norm used to measure the approximation error is the so called Hankel norm $\|\cdot\|_H$. For stable transfer functions $H(z) = C(zI - A)^{-1}B$, $|\lambda_i(A)| < 1$, this norm is defined as

$$\|H(z)\|_H = \|\Gamma[H(z)]\|_2 = \sigma_{\max}(\Gamma[H(z)]),$$

i.e. it is the l_2 norm (= maximal singular value) of the (block) Hankel matrix $\Gamma[H(z)] = [CA^{i+j}B]_{i,j=0}^{\infty}$. If $H(z)$ is not stable, $\|\cdot\|_H$ is not a norm, but we can define

$$\|H(z)\|_H = \|[H(z)]_-\|_H$$

with $[H(z)]_-$ the stable part of $H(z)$.
This type of approximation has been studied extensively by Adamjan, Arov and Krein [1,2] and is therefore called an AAK approximation. Since the end of the seventies, many papers appeared on this subject and related problems of Pick-Nevanlinna and Schur. In [12] a bibliography of 189 references is listed. In the next section we shall briefly recall some theoretical aspects of the approach of Kung and Lin [8] to the AAK problem. A numerical implementation of their algorithm is tested and commented upon in the next sections.

2 The algorithm of Kung and Lin: Theory

We shall start with some notations and definitions.
Let $\{F_i\}_{i\in\mathbb{Z}}$ be a complex square summable matrix sequence. Then we set

$$F(z) = \sum_{-\infty}^{\infty} F_k z^{-k},$$

$$\tilde{F}(z) = z^{-1}F(z^{-1}) = \sum_{-\infty}^{\infty} F_k z^{k-1}$$

and

$$F^*(z) = \sum_{-\infty}^{\infty} F_k^* z^{-k}$$

where the upper * denotes complex conjugate transpose.
If $F(z)$ is a polynomial matrix of degree n, then

$$\hat{F}(z) = z^n F(z^{-1}).$$

Suppose $H(z) = \sum_1^\infty H^k z^{-k}$ is a stable transfer function with $H_k \in \mathbb{C}^{p\times p}$. Let $\Gamma[H(z)] = [H_{i+j}]_{i,j=0}^\infty$ be the Hankel matrix associated with $H(z)$. It has a singular value decomposition (SVD)

$$\Gamma = \sum_{k=1}^\infty \sigma_i \zeta^{(i)} \eta^{(i)*} = U\Sigma V^*.$$

$U = [\zeta^{(1)}, \zeta^{(2)}, \ldots]$, $V = [\eta^{(1)}, \eta^{(2)}, \ldots]$. These are the left, resp. right singular vectors and the matrix $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots)$ contains the corresponding singular values. $\zeta^{(i)}(z) = \sum_1^\infty \zeta_k^{(i)} z^{-k}$ is the z-transform of $\zeta^{(i)} = [\zeta_1^{(i)T}, \zeta_2^{(i)T}, \ldots]^T$. Similarly, $\eta^{(i)}(z)$ represents the z-transform of $\eta^{(i)}$.
With these notations we can give the solution of the MDA problem. The following theorem gives the solution for a square $p \times p$ transfer function if the approximation error ρ is equal to a singular value of $\Gamma[H(z)]$ which has multiplicity p.

Theorem 1 *Suppose we have a stable transfer function*

$$H(z) = \frac{1}{a(z)} N(z)$$

with $a(z)$ a monic polynomial of degree N and $N(z)$ a $p \times p$ polynomial matrix of degree $\leq N-1$. Suppose the SVD

$$\Gamma[H(z)] = \sum_{k=1}^\infty \sigma_k \zeta^{(k)} \eta^{(k)*}$$

satisfies

$$\sigma_k > \rho = \sigma_{k+1} = \ldots = \sigma_{k+p} > \sigma_{k+p+1}.$$

Then the MDA solution $H^{(k)}(z)$ with approximation error ρ satisfies

$$\mathrm{rank}\, \Gamma[H^{(k)}(z)] = k$$

$$H^{(k)}(z) = [T(z)\hat{P}^{-1}(z)]_- = [\hat{Q}^{*-1}(z) W^*(z)]_-$$

where

$$X(z) = [\eta^{(k+1)}(z), \ldots, \eta^{(k+p)}(z)] = \frac{1}{a^*(z)} P(z),$$

$$Y(z) = [\zeta^{(k+1)}(z), \ldots, \zeta^{(k+p)}(z)] = \frac{1}{a(z)} Q(z).$$

$P(z)$ and $Q(z)$ are polynomial matrices of degree $\leq N-1$ and so are

$$T(z) = \frac{N(z)\hat{P}(z) - \rho \hat{a}^*(z) Q(z)}{a(z)}$$

and

$$W(z) = \frac{N^*(z)\hat{Q}(z) - \rho \hat{a}(z) P(z)}{a^*(z)}.$$

A proof of this theorem is given in [8].
The more general case where $\rho \in (\sigma_{k+1}, \sigma_k)$ can be found from the previous characterization via an extension of the Hankel matrix $\Gamma[H(z)]$.
If

$$H(z) = \sum_{k=1}^{\infty} H_k z^{-k}$$

and $\underline{H}(z) = H_0 z^{-1} + H(z)z^{-1}$, then $\Gamma[\underline{H}(z)]$ is called an extension of $\Gamma[H(z)]$.
Kung and Lin also prove the following theorem.

Theorem 2 *For a given $H(z)$ and a given $\rho \in (\sigma_{k+1}, \sigma_k)$, where $\sigma_1 \geq \sigma_2 \geq \ldots$ are the singular values of $\Gamma[H(z)]$, we can almost always find a matrix H_0 such that the extension $\Gamma[\underline{H}(z)]$ of $\Gamma[H(z)]$ has singular values*

$$\underline{\sigma}_1 \geq \ldots \geq \underline{\sigma}_k > \rho = \underline{\sigma}_{k+1} = \underline{\sigma}_{k+2} = \cdots = \underline{\sigma}_{k+p} > \underline{\sigma}_{k+p+1} \geq \cdots$$

With theorems 1 and 2 we can now formulate a theorem to characterise the MDA solution for $\rho \in (\sigma_{k+1}, \sigma_k)$.

Theorem 3 *Given $H(z)$ as in theorem 1 and $\rho \in (\sigma_{k+1}, \sigma_k)$ where $\sigma_1 \geq \sigma_2 \geq \ldots$ are the singular values of $\Gamma[H(z)]$. Construct the extension $\Gamma[\underline{H}(z)]$ as in theorem 2 and generate $\underline{T}(z)$, $\underline{P}(z)$, $\underline{Q}(z)$ and $\underline{W}(z)$ for the extension $\Gamma[\underline{H}(z)]$ as in theorem 1. Then*

$$H_{MDA}(z) = [z\underline{T}(z)\hat{\underline{P}}^{-1}(z)]_- = [\hat{\underline{Q}}^{*-1}(z)\underline{W}^*(z)z]_-$$

and

- $\|H(z) - H_{MDA}(z)\|_H \leq \rho$
- *The McMillan degree of $H_{MDA}(z) = k$.*

A constructive proof is given in [8].
Notes

1. It should be emphasized that theorem 3 only gives a solution for the MDA problem when $\rho \in (\sigma_{k+1}, \sigma_k)$, i.e. if ρ is <u>not</u> a singular value of $\Gamma[H(z)]$! This is a weak point of this characterization since ρ will rarely be equal to a singular value, but it can well be within a numerical neighbourhood and in that case, numerical difficulties may be expected.

2. If $H(z)$ is non-square, we can easily bring it into a square form by adding zero rows or columns and afterwards dropping the corresponding rows or columns in the solution. E.g. if $r \leq m$ replace $H(z)$ by

   $$\overline{H}(z) = \begin{bmatrix} 0_{(m-r)\times m} \\ H(z) \end{bmatrix},$$

   construct the solution $\overline{H}_{MDA}(z)$ for $\overline{H}(z)$ and set

   $$H_{MDA}(z) = [0 \;\; I_r]\overline{H}_{MDA}(z).$$

If $r > m$, replace $H(z)$ by

$$\overline{H}(z) = [0_{r\times(r-m)} \quad H(z)]$$

construct the solution $\overline{H}_{MDA}(z)$ for $\overline{H}(z)$ and set

$$H_{MDA}(z) = \overline{H}_{MDA}(z) \begin{bmatrix} 0 \\ I_m \end{bmatrix}.$$

In both cases, $H_{MDA}(z)$ still satisfies 1 and 2 of theorem 3.
From now on we shall suppose that $H(z)$ is $p \times p$ with $p = \max(m, r)$.

From the proofs of theorems 2 and 3, which are constructive, Kung and Lin derive the following algorithm, given $H(z)$, a $p \times p$ stable transfer function and ρ the approximation error bound.

1. Write $H(z)$ as $H(z) = a^{-1}(z)N(z)$ with $N(z)$ a polynomial matrix of degree $\leq N-1$ and $a(z)$ a monic polynomial of degree N.

2. Solve the polynomial matrix equation

   $$V(z)Z(z) = 0$$

   given

   $$V(z) = \begin{bmatrix} a(z)I_p & -N(z) & \rho\hat{a}^*(z)I_p & 0 \\ 0 & \rho a(z)I_p & -\hat{N}^*(z) & \hat{a}^*(z)I_p \end{bmatrix} \in \mathbb{C}^{2p\times 4p}[z]$$

   ($\hat{N}(z) = z^n N(z^{-1})$) to find

   $$Z(z) = \begin{bmatrix} R'(z) & R''(z) \\ \underline{\hat{P}}'(z) & \underline{\hat{P}}''(z) \\ \underline{Q}'(z) & \underline{Q}''(z) \\ \hat{R}'(z) & \hat{S}''(z) \end{bmatrix} \in \mathbb{C}^{4p\times 2p}[z].$$

3. Solve the algebraic Riccati equation

   $$R_0{}' - R_0'' H_0^* + H_0 \underline{\hat{P}}_0' - H_0 \underline{\hat{P}}_0'' H_0^* = 0$$

   where $(R_0{}')^* = R_0{}'$, $(\underline{\hat{P}}_0'')^* = \underline{\hat{P}}_0''$ and $(\underline{\hat{P}}_0')^* = -R_0''$.

4. Find the MDA approximant as one of the following expressions

   $$\begin{aligned} H_{MDA}(z) &= [(R'(z) - R''(z)H_0^*)(\underline{\hat{P}}'(z) - \underline{\hat{P}}''(z)H_0^*)^{-1}]_- \\ &= [(\underline{\hat{Q}}'^*(z) - H_0\underline{\hat{Q}}''^*(z))^{-1}(S'^*(z) - H_0 S''^*(z))]_- \end{aligned}$$

3 Numerical aspects

In this section we shall give a detailed description of the numerical implementation of the different steps of the algorithm. Although the theory was given for complex data, we shall suppose for practical applications that the data are real.

3.1 Write $H(z)$ in the form $N(z)/a(z)$

We suppose that the system is given by its state space description (A, B, C). We shall bring $H(z) = C(zI - A)^{-1}B$ into the form $H(z) = N(z)/a(z)$ following the method of Varga [13] and Verhaegen [14]. The computation of the *rtimesm* transfer function $H(z)$ is reduced to the computation of $r \times m$ scalar $h_{ij}(z) = c_i^T(zI - A)^{-1}b_j$ where c_i^T is the i-th row of C and b_j is the j-th column of B. In what follows we drop the indices for notational convenience. Thus we compute $h(z) = c^T(zI - A)^{-1}b$.

3.1.1 The scalar minimal realization

To obtain a minimal realization we shall have to isolate the observable and the controllable part of the system (A, b, c^T).

3.1.1.1 The controllable part This is found by using orthogonal similarity transformations (so that poles and zeros are not altered) to bring (A, b, c^T) into the form

$$\tilde{A} = \begin{bmatrix} \tilde{A}_1 & 0 \\ \tilde{A}_3 & \tilde{A}_2 \end{bmatrix}, \quad \tilde{b} = \begin{bmatrix} 0 \\ \tilde{b}_2 \end{bmatrix}, \quad \tilde{c}^T = [\tilde{c}_1^T \ , \ \tilde{c}_2^T]$$

such that $[\tilde{A}, \tilde{b}]$ is in lower Hessenberg form. $(\tilde{A}_2, \tilde{b}_2, \tilde{c}_2^T)$ will then be the controllable part. This is done by the following algorithm:

$ab = [A \ , \ b] \in \mathbb{R}^{n \times (n+1)}$
rank=0
if $\|b\| \leq \epsilon$ **then** stop {there is no controllable part}
rank=rank+1
if $n = 1$ **then** stop
for $l = n + 1$ **downto** 3 **do**
begin

- Use Householder transformations Q to make the first $l-2$ elements of column l of ab equal to zero.
- $A = QAQ^T$; $b = Qb$; $c^T = c^TQ^T$.
  if $\max(|ab(i, l-1)|, i = 1, 2, \ldots, l-2) \leq \epsilon$ **then**
  begin
  $\tilde{A}_2 = \begin{bmatrix} 0 & 0 \\ 0 & I_{rank} \end{bmatrix} A \begin{bmatrix} 0 & 0 \\ 0 & I_{\text{rank}} \end{bmatrix}$
  $\tilde{b}_2 = \begin{bmatrix} 0 & I_{\text{rank}} \end{bmatrix} b$
  $\tilde{c}_2^T = c^T \begin{bmatrix} 0 \\ I_{\text{rank}} \end{bmatrix}$
  stop
  end if
- rank $= rank + 1$

end for

Note that $\tilde{A}_2$ will be in lower Hessenberg form.

3.1.1.2 The observable part This is completely analogous to the previous computations. Similarity transformations are now used to bring $(\tilde{A}_2, \tilde{b}_2, \tilde{c}_2^T)$ into the form

$$\hat{A}_2 = \begin{bmatrix} \hat{A}_{21} & \hat{A}_{23} \\ 0 & \hat{A}_{22} \end{bmatrix}, \quad \hat{b}_2 = \begin{bmatrix} \hat{b}_{21} \\ \hat{c}_{22} \end{bmatrix}, \quad \hat{c}_2^T = [0 \;\; \hat{c}_{22}^T].$$

This can be done by the previous algorithm since we have to find the controllable part of the dual system $(\tilde{A}_2^T, \tilde{c}_2, \tilde{b}_2^T)$. $\hat{A}_{22}$ will be in upper Hessenberg form. Again we simplify the notation $(\hat{A}_{22}, \hat{b}_{22}, \hat{c}_{22}^T)$ to (A, b, c^T).

3.1.2 Computation of the poles

Since (A, b, c^T) is now observable and controllable, we can find the poles of $h(z) = c^T(zI - A)^{-1}b$ as the eigenvalues of A. This can be comfortably done with a standard routine like HQR from EISPACK [10]. Note that A is already in its Hessenberg form.

3.1.3 Computation of the zeros

We use the method of Laub [9]. The zeros are the finite solutions of the generalized eigenvalue problem

$$\begin{bmatrix} A & b \\ c^T & 0 \end{bmatrix} x = \mu \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} x.$$

Note that the left hand side matrix is already in upper Hessenberg form and the right hand side matrix is upper triangular, so that the standard QZ algorithm from EISPACK [10] is directly applicable. The routines QZIT and QZVAL will reduce these matrices to triangular form. Suppose the diagonal elements of the left hand side matrix are $\alpha_1, \ldots, \alpha_{q+1}$ and the diagonal elements of the right hand side matrix are $\beta_1, \ldots, \beta_{q+1}$. Then in general we have three possibilities:

- $\beta_i \geq 0$, then $\mu_i = \alpha_i/\beta_i$ is a zero.
- $\beta_i = 0$ and $\alpha_i > 0$, then there is a zero at infinity.
- $\beta_i = \alpha_i = 0$ which is excluded in our situation.

3.1.4 Computation of the gain factors

If we write $h(z)$ as

$$h(z) = k \frac{\prod_{i=1}^{nz}(z - \mu_i)}{\prod_{j=1}^{q}(z - \lambda_j)},$$

then the gain k is the only unknown parameter left. It can be found as follows: choose $|z_0| > \max(|\lambda_i|, |\mu_j|)$ and set

$$k = c^T(z_0 I - A)^{-1} b \frac{\prod_{j=1}^{q}(z_0 - \lambda_j)}{\prod_{i=1}^{nz}(z_0 - \mu_i)}.$$

The following algorithm avoids the computation of the inverse:

Choose $|z_0| > \max(|\lambda_j|, |\mu_i| : i = 1, \ldots, nz, j = 1, \ldots, q)$
$H = z_0 I - A$ {H is upper Hessenberg}
{Suppose $H = [\eta_{ij}]$, $b = [\beta_i]$, $c^T = [\gamma_i]$}
if $q > 1$ **then**
  begin
    Find the orthogonal transformation Q
    such that QH is upper triangular
    $\{(QH)^{-1}(Qb) = H^{-1}b\}$
    $H = QH$, $b = Qb$

  end if

$$k = \frac{\gamma_q \beta_q}{\eta_{qq}} \frac{\prod_{j=1}^{q}(z_0 - \lambda_j)}{\prod_{i=1}^{nz}(z_0 - \mu_i)} \qquad \{\gamma_1 = \cdots = \gamma_{q-1} = 0\}$$

3.1.5 Computation of $a(z)$ and $N(z)$

Given the poles, zeros and gain factors of each scalar entry of the transfer function, it is easy to compute the denominator $a(z)$ as the least common multiple of the denominators of each entry. The numerator $N(z)$ is then easily found. It is also in this step that $N(z)$ is made square by adding zero rows or columns.
This concludes our description of the first step of the algorithm.

3.2 Solution of the system $V(z)Z(z) = 0$

We know that $V(z)$ can be written as

$$V(z) = \sum_{i=0}^{N} V_i z^i \quad , \quad V_i \in \mathbb{R}^{2p \times 4p}.$$

Thus $Z(z)$ will also have degree N and thus we define

$$Z(z) = \sum_{i=0}^{N} Z_i z^i \quad , \quad Z_i \in \mathbb{R}^{4p \times 2p}.$$

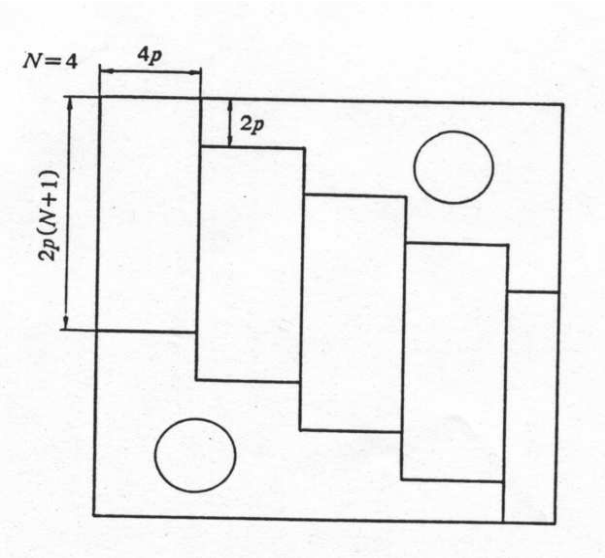
$V(z)Z(z) = 0$ is equivalent with

$$\begin{bmatrix} V_0 & 0 & \cdot & \cdot & \cdot & 0 \\ V_1 & V_0 & 0 & & & \cdot \\ \cdot & V_1 & \cdot & & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & & \cdot & 0 \\ V_N & V_{N-1} & \cdot & \cdot & \cdot & V_0 \\ 0 & V_N & & & \cdot & \cdot \\ \cdot & 0 & \cdot & & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & & & \cdot \\ 0 & 0 & 0 & 0 & 0 & V_N \end{bmatrix} \begin{bmatrix} Z_0 \\ Z_1 \\ \cdot \\ \cdot \\ \cdot \\ Z_N \end{bmatrix} = 0.$$

If ρ is not equal to a singular value of $\Gamma[H(z)]$, then there always exists a solution with Z_N of the form

$$Z_N = \left[\begin{array}{c} \star \\ \hline I_{2p} \end{array}\right]. \tag{3.1}$$

Thus we can bring the last $2p$ columns of the left hand side matrix of the homogeneous system to the right hand side so that we obtain a system of linear equations with $2p$ right hand sides and with a coefficient matrix of the form



(3.2)

The size of this square matrix is $2p(2N+1)$. All the blocks are identical and the last block is only the left part of it. N is bounded by the McMillan degree which may be rather large, (otherwise model reduction would not be in order). A reasonable value is $N = 15$ and $p = 3$, so that the system is of the order $2p(2N + 1) = 186$. Thus ordinary Gauss-elimination is not appropriate. Using the special structure of the system, Kung [6,7] designed an efficient algorithm to solve it. For its formulation we need some notation.
x/Y denotes the projecting vector when $x \in \mathbb{R}^n$ is orthogonally projected onto the space spanned by the vectors $Y = [y_1, \ldots, y_q]$, $y_i \in \mathbb{R}^n$. I.e.

$$x/Y = [x - \sum_{i=1}^{q} \alpha_i y_i : \|x - \sum_{i=1}^{q} \alpha_i y_i\|_2 \quad \text{is minimal}]$$

Its normalized version is

$$x|Y = \begin{cases} (x/Y)\ \|x/Y\|_2^{-1} & \text{if } \|x/Y\|_2 \neq 0 \\ 0 & \text{otherwise.} \end{cases}$$

If $X = [x_1, \ldots, x_m]$ is a matrix, we set

$$X|Y = [z_1, \ldots., z_m]$$

with

$$z_i = [x_i|Y, x_1, \ldots, x_{i-1}].$$

The numerical computation of $X|Y$ is done by the following projection procedure

1. Perform a QR decomposition of $[Y, X]$

$$Q[Y, X] = \left[\frac{U}{0_{n-(q+m)\times(q+m)}}\right] \qquad \text{with } U \text{ upper triangular}$$

2. Compute P regular upper triangular such that

$$UP = \begin{bmatrix} I_q^{(1)} & 0_{q\times m} \\ 0_{m\times q} & I_m^{(2)} \end{bmatrix}$$

with $I_q^{(1)}$ and $I_m^{(2)}$ identity matrices with certain diagonal elements equal to zero because of the zero diagonal elements in U. In our application, because (3.2) is nonsingular, $I_q^{(1)}$ and $I_q^{(2)}$ will be ordinary identity matrices. Therefore, P is just the inverse of the triangular matrix U. Set

$$P = \begin{bmatrix} P_{11} & P_{12} \\ 0_{m\times q} & P_{22} \end{bmatrix}.$$

3. Define

$$T := YP_{12} + XP_{22}.$$

Then $T = X|Y$.

With an obvious use of notation we denote the polynomial matrix associated with the first block column of the matrix (3.2) as $V(z)$ and for the next block columns we get $zV(z), z^2V(z), \ldots$. Define

$$\phi_k(z) := z^kV(z)|[z^{k-1}V(z), ..., zV(z), V(z)] \quad , k \geq 1 \tag{3.3}$$

and $\alpha_k(z)$ by

$$\phi_k(z) = V(z)\alpha_k(z). \tag{3.4}$$

Since the matrix (3.2) has full rank, the last $2p$ columns of $\phi_N(z)$ will be zero. Thus by equating the last $2p$ columns in (3.4) we find that $Z(z)$ will be the last $2p$ columns of $\alpha_N(z)$. Kung shows that with the auxiliary matrix

$$\psi_k(z) := V(z)|[z^kV(z), ..., zV(z)] \quad , \quad k \geq 1$$

we have

$$\begin{cases} \phi_{k+1} = [z\phi_k(z)|\psi_k(z) \\ \psi_{k+1}(z) = \psi_k(z)|[z\phi_k(z)] \\ \phi_0(z) = \psi_0(z) = \tilde{V}(z) \end{cases}$$

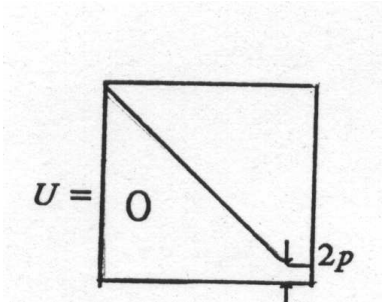
with $\tilde{V}(z) = V(z)P_{-1}$ the orthonormalized version of $V(z)$. I.e. $P_{-1} = U^{-1}$ if U is the upper triangular part in the QR decomposition of the first block column of (3.2).
This recursion can be reformulated as

$$\begin{cases} \phi_{k+1}(z) = z\phi_k(z)P_k^{(22)} + \psi_k(z)P_k^{(12)} \\ \psi_{k+1}(z) = \psi_k(z)Q_k^{(22)} + z\phi_k(z)Q_k^{(12)} \end{cases} \tag{3.5}$$

where both the P_k and Q_k matrices are generated as the P matrix in the projection procedure above. In this application $q = m = 2p$. P_k is generated when projecting $z\phi_k(z)$ onto $\psi_k(z)$ and Q_k is generated when we project $\psi_k(z)$ onto $z\phi_k(z)$. If we introduce $\beta_k(z) = V(z)\psi_k(z)$, then it can be seen that $\alpha_k(z)$ and $\beta_k(z)$ also satisfy recursion (3.5) with initial conditions $\alpha_0(z) = \beta_0(z) = P_{-1}$. Thus we can compute recursively $\phi_i(z)$, $\psi_i(z)$, $\alpha_i(z)$, $\beta_i(z)$, $i = 0, 1, ..., N$ and find $Z(z)$ as the last $2p$ columns of $\alpha_N(z)$.
We are left with one problem viz. it is not guaranteed that Z_N will have the form (3.1). This is realized as follows. Suppose we computed in step $N-1$ of the recursion

$$Q[\psi_{N-1}(z), z\phi_{N-1}(z)] \;=\; \left[\frac{U}{0}\right] \quad , \quad U \text{ upper triangular.}$$

Now $[z\phi_{N-1}(z)]|\psi_{N-1}(z) = \phi_N(z) = z^N V(z)|[z^{N-1}V(z), ..., V(z)]$. Because the last $2p$ columns of $[V(z), zV(z), ..., z^N V(z)]$ are linearly dependent on the previous ones, U will have the form



and

$$UP = \begin{bmatrix} I_{4p}^{(1)} & 0 \\ 0 & I_{4p}^{(2)} \end{bmatrix}.$$

Since we are only interested in the last $2p$ columns of P, we subdivide U and P as

$$\begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0_{2p\times 4p} & U_{22} & U_{23} \\ 0_{2p\times 4p} & 0_{2p\times 2p} & 0_{2p\times 2p} \end{bmatrix} \begin{bmatrix} X_{4p\times 6p} & D_{4p\times 2p} \\ Y_{2p\times 6p} & E_{2p\times 2p} \\ 0_{2p\times 6p} & F_{2p\times 2p} \end{bmatrix} = \begin{bmatrix} I_{6p} & 0 \\ 0 & I_{2p} \end{bmatrix}$$

which gives

$$U_{11}D + U_{12}E + U_{13}F = 0_{4p\times 2p} \tag{3.6}$$

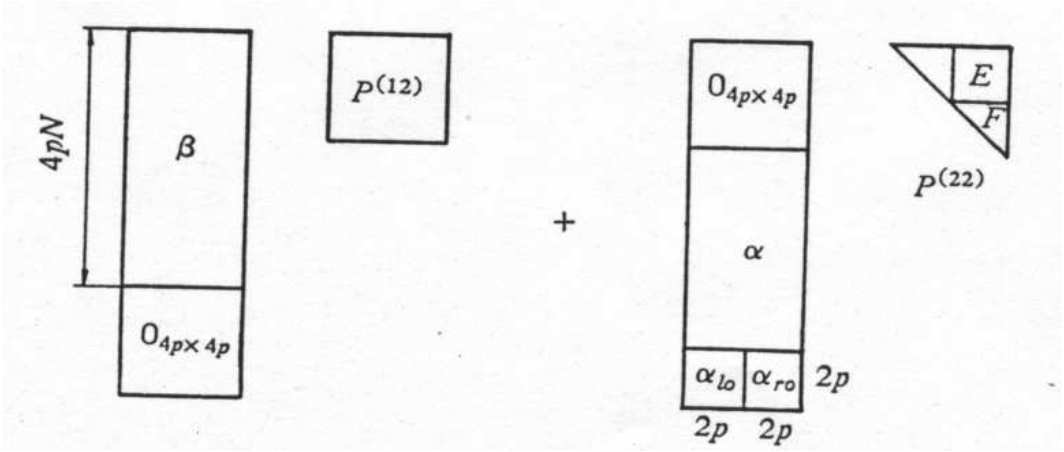
$$U_{22}E + U_{23}F = 0_{2p} \tag{3.7}$$

$$0F = 0. \tag{3.8}$$

So F is arbitrary (but regular) and E and F are then found from the other relations. F will be fixed by condition (3.1) as follows : Write

$$\alpha_N(z) = \beta_{N-1}(z)P_{N-1}^{(12)} + z\alpha_{N-1}(z)P_{N-1}^{(22)}$$

explicitly as



Taking the last $2p$ rows and $2p$ columns gives

$$\alpha_{l0}E + \alpha_{r0}F = I_{2p}.$$

Together with (3.7), this gives

$$\begin{bmatrix} \alpha_{l0} & \alpha_{r0} \\ U_{22} & U_{23} \end{bmatrix} \begin{bmatrix} E \\ F \end{bmatrix} = \begin{bmatrix} I_{2p} \\ 0_{2p} \end{bmatrix}$$

with U_{22} and F upper triangular. Note that the i-th column of the solution of this system has only its first $2p + i$ elements which are nonzero. Thus to solve the system we only need one LU decomposition (SGECO routine from LINPACK [4]) and $2p$ calls of the backsubstitution routine (SGESL from LINPACK) with an increasing dimension.
Once E and F are found, D can be solved from the triangular system (3.6)

$$U_{11}D = -U_{12}E - U_{13}F$$

(the routine STRSL from LINPACK).

This projection method requires $O(N^2p^3)$ operations as opposed to a Gaussian elimination which requires $O(N^3p^3)$ operations. The memory requirements is $O(Np^2)$ as opposed to $O(N^2p^2)$.

3.3 Solution of the algebraic Riccati equation

This is a well known problem. Laub [9] described an excellent procedure for it which is implemented in the SLICE routine RILAC [17]. We don't elaborate on this topic.

3.4 Computation of $H_{MDA}(z)$

Kung and Lin [8] originally propose to construct $H_{MDA}(z) = [\underline{\hat{Q}}^{*-1}(z)S^*(z)]_-$ by partial fraction expansion and thus isolate the stable part. It is well known that this is an ill conditioned problem so that numerical difficulties may be expected. Therefore we propose another procedure which is :

1. Compute a state space realization (A, B, C, D) from $Q(z)$ and $S(z)$ such that $C(zI - A)^{-1}B + D = \hat{Q}^{*-1}(z)S^*(z)$.
2. Isolate the stable part of (A, B, C, D).

We choose here a left MFD $[\underline{\hat{Q}}^{*-1}(z)S^*(z)]_-$ instead of the right MFD $[R(z)\underline{\hat{P}}^{-1}(z)]_-$ because

$$\underline{Q}(z) = \underline{Q}'(z) - \underline{Q}''(z)H_0^* \qquad \text{and} \qquad \hat{S}(z) = \hat{S}'(z) - \hat{S}''(z)H_0^*$$

where $\underline{Q}'(z)$, $\underline{Q}''(z)$, $\hat{S}'(z)$ and $\hat{S}''(z)$ form the lower part of $Z(z)$. Therefore

$$\begin{bmatrix} \underline{Q}'_N & \underline{Q}''_N \\ \hat{S}'_N & \hat{S}''_N \end{bmatrix} = I_{2p}.$$

Thus we directly find for the highest degree coefficients $\underline{Q}_N = I_p$ and $\hat{S}_N = -H_0^* = -H_0$.

3.4.1 Computation of a state space realization

To compute a state space realization from the MFD $D^{-1}(z)N(z)$, we should first bring $D(z)$ in its row reduced form. This means that the matrix formed by the highest degree coefficients of each row of $D(z)$ should be of full row rank.
Thus if $d_1, d_2, ..., d_p$ are the highest degrees of the rows of $D(z)$, then we can write

$$D(z) = S(z)D_{hr} + D_{lr}(z)$$

with $S(z) = \text{diag}(z_1^d, ..., z_p^d)$. by a unimodular similarity transformation it is always possible to bring $D(z)$ in its row reduced form. This is done as follows. Suppose that $\text{rank}\, D_{hr} = k$. Then we can always find a permutation matrix E and an orthogonal matrix Q such that

$$ED_{hr}Q = \begin{bmatrix} R_{11} & 0 \\ R_{12} & 0 \end{bmatrix}$$

with $R_{11} \in \mathbb{R}^{k\times k}$ lower triangular. The rows $k+1, ..., k+p$ of ED_{hr} are then linear combinations of the first k rows. Thus if

$$ED_{hr} = \begin{bmatrix} D_1 \\ D_2 \end{bmatrix} \quad , \quad D_1 \in \mathbb{R}^{k\times p} \quad , \quad D_2 \in \mathbb{R}^{(p-k)\times p}$$

then there exists an $A = R_{12}R_{11}^{-1}$ such that $D_2 = AD_1$. Thus $p - k$ of the highest degree coefficients can be annihilated. This procedure is repeated on what remains of $D(z)$ until D_{hr} has full row rank.
Even if $D(z)$ is row reduced, this procedure can be applied to improve the condition of D_{hr}.

In our examples, $D(z)$ always turned out to be row reduced and the condition of D_{hr} was not too bad so that we didn't implement this.

The computation of (A, B, C, D) from $D^{-1}(z)N(z)$ with $D(z)$ row reduced is now a classical problem. It is based on the structure theorem of Wolovich [18], [16]. It is implemented in the SLICE routine PMXSS [17]. The result is found in Luenberger canonical form and (A, C) is observable. To make (A, B, C, D) minimal, (i.e. also controlable), we use another SLICE routine SSXMR. The method is described in [11].

3.4.2 Isolation of the stable part

In what follows, the transfer functions need not be square. Therefore we again use the original number of inputs (m) and outputs (r). We know at this moment (A, B, C) such that $\overline{H}_{MDA}(z) = C(zI - A)^{-1}B$ (we drop D again). If $r \leq m$, then $H_{MDA}(z)$ is obtained by dropping the first $m - r$ rows in $\overline{H}_{MDA}(z)$. I.e. we drop the first $m - r$ rows in C. If $r > m$, we drop the first $r - m$ columns from B.
We are now after the minimal realization of the stable part of the system (A, B, C). Our method is the discrete analog of an algorithm of Glover [5]. It goes as follows :

1. Find the orthogonal matrix V_1 such that $V_1^T A V_1 = \tilde{A}$ is in real upper Schur form.

2. Find the orthogonal matrix V_2 such that

$$V_2^T \tilde{A} V_2 = \begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} \\ 0 & \tilde{A}_{22} \end{bmatrix}$$

   with all the eigenvalues of A_{11} in the open unit disc and all eigenvalues of A_{22} outside the open unit disc. If the number of eigenvalues inside is k, then k is the McMillan degree of the stable part.

3. Solve $\tilde{A}_{11}X - X\tilde{A}_{22} + \tilde{A}_{12} = 0$ for $X \in \mathbb{R}^{k\times(n-k)}$.

4. If $T = V_1 V_2 \begin{bmatrix} I & X \\ 0 & I \end{bmatrix}$, then set

$$A = T^{-1}\tilde{A}T = \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix}$$

$$T^{-1}B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}$$

$$CT = [C_1, C_2]$$

   and (A_{11}, B_1, C_1) will be the stable part.

Note that the routines to perform steps 1 and 2 were also needed to solve the Riccati equation. The solution of the Sylvester equation in step 3 can be done with the SLICE routine SYHSC [17].

4 Numerical performance

We give here two examples on which the algorithm is tested.

Example 1 [8]

This is an artificial 4-th order discrete system with 2 inputs and 2 outputs. Its state space formulation is

$$A = \begin{bmatrix} 0.5 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.5 & 0.0 & 0.0 \\ 0.0 & 0.0 & -0.5 & 1.0 \\ 0.0 & 0.0 & 0,0 & -0.5 \end{bmatrix} \quad , \quad B = \begin{bmatrix} 1.0 & 1.0 \\ 1.5 & 0.0 \\ 0.0 & 7.75 \\ -0.25 & -0.75 \end{bmatrix}$$

$$C = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.8333333 & 1.0 & 9.0 \end{bmatrix} \quad , \quad D = 0.$$

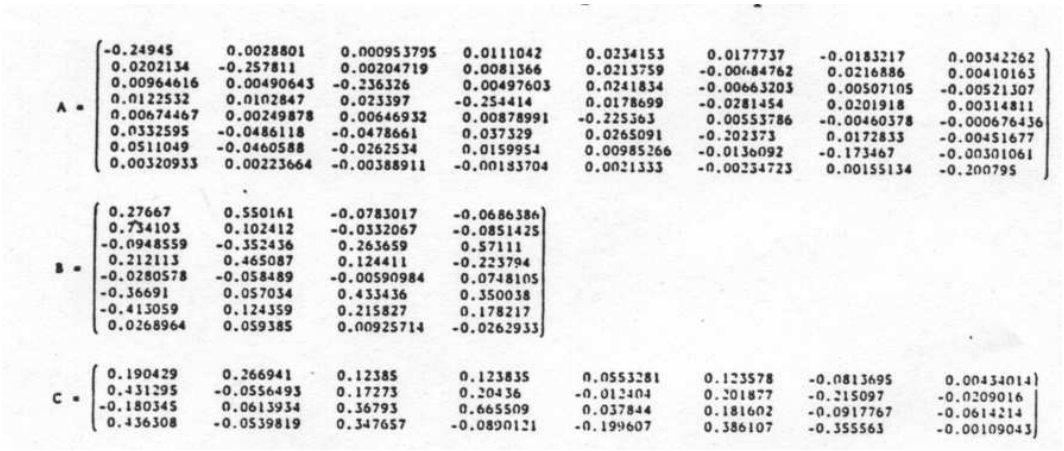
The eigenvalues of A (i.e. λ_i) and the Hankel singular values (σ_i) are given below

i	1	2	3	4
λ_i	0.500	0.500	-0.500	-0.500
σ_i	5.56	3.82	1.33	1.04

Table 1:

Example 2 [3]

This is a 8-th order continuous system with 4 inputs and 4 outputs



The continuous system (A, B, C, D) is transformed into a discrete system $(\hat{A}, \hat{B}, \hat{C}, \hat{D})$ by the well known transformation

$$\hat{A} = (I + A)(I - A)^{-1}$$
$$\hat{B} = \sqrt{2}(I - A)^{-1}B$$
$$\hat{C} = \sqrt{2}C(I - A)^{-1}$$
$$\hat{C} = D + C(I - A)^{-1}B$$

i	1	2	3	4	5	6	7	
λ_i	-0.200	-0.200	-0.200	-0.200	0.200	0.200	0.200	0.200
σ_i	1.13	0.527	0.186	0.178	0.0116	0.0108	0.00346	0.000270

Table 2:

Example 1		Example 2	
ρ	order	ρ	order
2.00	2	0.400	2
1.40	2	0.200	2
1.30	3	0.0116	4
1.25	3	0.0115	5
1.25	3	0.00700	6

Table 3:

The algorithm of Kung is then applied and afterwards, the inverse transformations can be done on the reduced system.
The eigenvalues (λ_i) of $\hat{A}$ and the Hankel singular values (σ_i) are given in table 2. The values of ρ that were tested are given in table 3. Table 3 also displays the order of the reduced system as it should be found by the algorithm.

From computer runs of the algorithm on these examples we can draw some conclusions about its numerical performance. Although the algorithm consists of orthogonal transformations, the solution of systems and elementary matrix operations which can all be implemented in a numerically stable way, some steps may cause trouble. E.g.
Rank determination.

1. The degrees of $a(z)$ and $N(z)$ in the transfer function have to be found numerically. This is a rather difficult problem for numerical computations (computation of poles and zeros of each entry). Not withstanding the presence of a double pole (ex. 1) or a 4-th order pole (ex. 2), the degrees were found correctly.

2. In the last step, when computing a minimal realization, a numerical rank has to be computed. This rank defines the McMillan degree of the reduced system. This failed in one situation (ex. 2 with $\rho = 0.0115$) which gave a McMillan degree 4 instead of 5. The reason is probably that ρ is too close to a singular value $\sigma_5 = 0.01159$.

For these 3 problems, the choice of a tolerance is critical. An automatic fail-safe setting of them is rather difficult and an interactive setting may be desirable.
System solution.
The systems we have to solve are found in the following steps:

1. $V(z)Z(z) = 0$: a system of order $2p(2N + 1)$

2. when solving the Riccati equation : a system of order p

3. when computing the realization (A, B, C, D) from a MFD, D_{hr}^{-1} has to be computed of order p.

These systems may be ill conditioned. The LINPACK routine gives an estimate of the inverse of the condition number : RCOND. If 1.0 + RCOND.eq.1.0 then the system is singular. Table 4 gives RCOND for the 3 systems described above for example 2. Because

ρ	1	2	3
0.400	0.46e-02	0.43	0.76
0.200	0.35e-03	0.25	0.70
0.0116	0.35e-06	0.25	0.29e-01
0.0115	0.51e-05	0.37	0.13e-01
0.0700	0.14e-03	0.33	0.23

Table 4: RCOND

system 1 is singular when ρ is equal to a Hankel singular value, the inverse condition number is very small when ρ is close to a singular value (see $\rho = 0.0115$ and $\rho = 0.0116$). We see that systems 2 and 3 cause no problem at all.
It turns out that the solution of system 1 is the most critical step in the algorithm. It is also the most time and memory consuming step. The projection method as we described it did already do a lot of saving in that respect, but a lot of structure was shown during the computations which may indicate that still more efficient algorithms do exist.
One may doubt if the projection method is a stable method for this system. In our opinion it is reasonably stable, since we replaced it by Gaussian elimination with optimal pivoting (which is possible here since the dimension (≈ 40) is not too large) and the same results were obtained except when ρ was too close to a singular value (see example 2 with $\rho = 0.0115$ and $\rho = 0.0116$). In these two cases the Gaussian elimination was better than the projection method. The loss of accuracy when solving system 1 is really a matter of conditioning which is intrinsically connected with the method of Kung which requires that ρ is not a Hankel singular value.
Determination of the stable part.

Kung and Lin expect the determination of the stable part of the reduced system to be the most difficult step. Their proposal was to do it with partial fraction decomposition. This step then requires $O(N^3p^3)$ operations and is numerically questionable. Our way of solving this problem by finding a state space realization seems to have circumvented these drawbacks completely.

We also ran these two examples with the routine GLOVER from SYCOT [15]. This routine implements the method of Glover [5] for Hankel-norm approximation. The routine is in a number of aspects different from the routine we have implemented.

- It is written for continuous time systems. Because the transformation between continuous and discrete systems is relatively easy, this causes no problem.

- The data for the algorithm are supposed to be a minimal balanced state space description of the system. Balancing and making the realization minimal are standard problems that can be solved easily.

- The routine is written for the MNA problem but it can easily be adapted for the MDA problem. Since the Hankel singular values are known from the balancing procedure, we can find k such that $\sigma_k > \rho \geq \sigma_{k+1}$. You then just look for the MNA approximant of McMillan degree k.

- The result is a reduced system given by its state space description instead of a matrix fraction description. Because of our implementation of the last step in the Kung and Lin method via state space realization, this difference is not essential.

This makes it possible to compare the performance of both algorithms. Our conclusion is that the method of Glover always gives better results than the Kung and Lin method. This discrepancy can be partly explained because the method of Glover will always take the ideal ρ ($= \sigma_{k+1}$) for an approximant of degree k, while in the Kung and Lin method, ρ may be closer to σ_k and still be in (σ_{k+1}, σ_k). If we take ρ very close to a singular value in the method of Kung and Lin in order to compare with the method of Glover, that method will always run into numerical problems as we have seen in example 2 for $\rho = 0.0115$ and $\rho = 0.0116$.

In general one may conclude that the Hankel norm criterion for approximating a system is rather good. Two drawbacks could be formulated:

1. The Hankel norm gives an absolute approximation criterion. Indeed, if $\|H(z) - H_{MDA}(z)\|_H \leq \rho$, then the difference between each entry in the Markov parameters of $H(z)$ and $H_{MDA}(z)$ will be less than ρ. If the system is stable, then the Markov parameters will decay after a certain time and the error bound ρ may be larger than the entries in those smaller Markov parameters, so that they are not really approximated as they should.

2. Another observation that one could make is that by comparing the step responses of the given and the approximating system, we noted that they don't match very well after a certain time. The reason is that the information used are the Markov parameters which model the transient behaviour of the system in the first place. It would be interesting to have also time moment information so that also the steady state behaviour of the system could be matched.

5 Conclusion

We have tried to give a careful numerical implementation of the algorithm of Kung and Lin. The last step of the algorithm which was predicted by Kung and Lin to be numerically the most difficult was solved satisfactory by isolating the stable part of the approximant via a state space description. The algorithm however contains another step which turned out to be ill conditioned when the approximation error bound ρ was close to a Hankel singular value. This is not a matter of numerical implementation, but it is a structural drawback of the method which doesn't give a solution if ρ is exactly equal to such a singular value. When using the method of Glover to solve a minimal degree approximation problem, we should first find the McMillan degree of the approximation and then solve a minimal norm approximation problem. This is only possible because the Hankel singular values are available in that case. In the method of Kung and Lin, the McMillan degree follows from the solution method and is not a priory determined since the Hankel singular values are not computed explicitly.

6 References

1. ADAMJAN V.M., AROV D.Z., KREIN M.G. (1971) Analytic properties of Schmidt pairs for a Hankel operator and the generalized Schur-Takagi problem, *Math. USSR Sbornik,* **15** (1) 31-73.

2. ADAMJAN V.M., AROV D.Z., KREIN M.G. (1978) Infinite Hankel block matrices and related extension problems, *American Mathematical Translations, Series 2,* **111** ,113-156.

3. DAVISON E.J., GESING W., WANG S.H. (1978) An algorithm for obtaining the minimal realization of a linear time-invariant system and determining if a system is stabilizable-detectable, *IEEE Trans. on Autom. Control,* **AC-23** (6) 1048-1054.

4. DONGARRA J.J., BUNCH J.R., MOLER C.B., STEWART G.W. (1979) *Linpack users guide,* SIAM

5. GLOVER K. (1984) All optimal Hankel-norm approximations of linear multivariable systems and their L^∞-error bounds, *Int. Journ. Control,* **39** (6) 115-1193.

6. KUNG S.Y., KAILATH T. (1980) Fast projection methods for minimal design problems in linear system theory, *Automatica,* **16**, 399-403.

7. KUNG S.Y., KAILATH T., MORF M. (1977) Fast and stable algorithms for minimal design problems, *4th IFAC symposium on multivariable technology and systems,* Fredericton N.B., Canada.

8. KUNG S.Y., LIN D.W. (1981) Optimal Hankel-norm model reductions: multivariable systems, *IEEE Trans. on Autom. Control,* **AC-26** (4) 832-852.

9. LAUB A.J. (1979) A Schur method for solving algebraic Riccati equations, *IEEE Trans. Autom. Control,* **AC-24** (6) 913-921.

10. SMITH B.T., BOYLE J.M., DONGARRA J.J., GARBOW B.S., IKEBE Y., KLEMA V.C., MOLER C.B. (1976) *Matrix eigensystems routines - EISPACK guide,* Lecture Notes in Computer Science 6, Springer Verlag, New York.

11. VAN DOOREN P.M. (1981) The generalized eigenstructure problem in linear system theory, *IEEE Trans. Autom. Contr.,* **AC-26** (1) 111-129.

12. VAN HEERTUM P. (1986) Benaderingen volgens Adamjan, Arov, Krein en aanverwante problemen: Theorie en algoritmes (in Dutch), Dissertation, Computer Science Department, K.U. Leuven.

13. VARGA A., SIMA V. (1981) Numerically stable algorithm for transfer function matrix evaluation, *Int. Journ. Control,* **33** (6) 1123-1133.

14. VERHAEGEN M.H. (1985) *A new class of algorithms in linear system theory,* Ph.D. Thesis, K.U.Leuven.

15. WERKGROEP PROGRAMMATUUR (1985) An inventory of basic software for computer aided control system design, *WGS-Report 85-1,* T.H. Eindhoven, The Netherlands.

16. WILLIAMS T.W.C. (1985) A numerically stable alternative to the structure theorem, School of Computing, Kingston Polytechnic, England.

17. WILLIAMS T.W.C. (1986) Numerically reliable software for control: The SLICE library, *IEE Proceedings D : Control theory and applications,* **133** (2) 73-82.

18. WOLOVICH W.A. (1974) *Linear multivariable systems,* Springer Verlag, New York.

7 Appendix

On the next pages we give a listing of the programs that are not available in some library. You will find the main program for the method of Kung and Lin followed by all the subroutines in an alphabetical order.

- COEFF
- CONTR
- HOUSH
- HSHTRF
- PROD
- STABL
- STLSL
- TRSFCT
- UPTR
- UPTR0
- VERSCH

Other routines from libraries that are needed are :

- **BLAS :**
  ISAMAX, SASUM, SAXPY, SDOT, SNRM2, SSCAL, SSWAP. .LI .B
- **LINPACK :**
  SGECO, SGEDI, SGEFA, SGESL, SQRDC, STRDI, STRSL.
- **EISPACK :**
  BALANC, BALBAK, HQR, ORTHES, ORTRAN, QZIT, QZVAL.
- **SLICE :**
  EXCHQR, HHDML, MATM, ORDERS, PMXDL, PMXSS, QTRORT, QRSTEP, RILAC, SDATA, SSXDL, SSXMC, SSXMR.
- **SYCOT :**
  AMAALB, BILNTR, CROUT2, EXCHNG, F04AEF, QRSTAP, SHRSLV, SORT, SPLIT, SWAPP, SYSSLV.

```
C
C THIS MAIN PROGRAM COMPUTES THE OPTIMAL HANKEL-NORM APPROXIMANT
C (MIN. DEGREE PROBLEM) FOR A MULTIVARIABLE STABLE SYSTEM WITH
C AN ALGORITHM OF KUNG EN LIN
C
      INTEGER NN,MM,RR,N,M,R,NZ(6,6),NP(6,6),GRDD,GRDVTM(6,6)
      INTEGER I,J,K,P,MAX0,IVV,IPHI,IALFA,IWRK(24),IERR,PP,ORD,L,KK
      INTEGER JOB,IWRK2(20),MINORD,NNPP
      REAL A(20,20),B(20,6),C(6,20),EPS,EPS2,ERROR,GAIN(6,6)
      REAL VTM(6,6,0:20),D(0:20),QCOEFF(6,6,21),SCOEFF(6,6,21)
      REAL WRK1(20,20),WRK2(492),WRK3(492),WRK4(21,21),WRK5(20)
      REAL WRK6(20),RCOND,SOM1,SOM2
      REAL WRK7(21),WRK8(21),WRK9(21,21),WRK10(21),WRK11(0:20)
      REAL AA(120,120),BB(120,6),CC(6,120),DD(6,6)
      REAL WRK12(20,20),Q2(492,48)
      REAL V(252,24),WRK17(492,48),WRK18(492,48),Q1(492,48)
      REAL PHI(492,24),PSI(492,24),ALFA(504,24),BETA(504,24)
      REAL Z(504,12),WRK19(20,20)
      COMPLEX ZERO(6,6,20),POLE(6,6,20)
      COMPLEX WRK13(20),WRK14(20),WRK15(20),WRK16(20)
C
C N   : MCMILLAN DEGREE OF THE ORIGINAL SYSTEM (MAX. NN=20)
C M   : NUMBER OF INPUTS (MAX. MM  = 6)
C R   : NUMBER OF OUTPUTS (MAX. RR = 6)
C
      NN = 20
      MM = 6
      RR = 6
C
C IF NN, MM OF RR ARE CHANGED, THE DECLARATIONS BECOME :
C
C     INTEGER NN,MM,RR,N,M,R,NZ(RR,MM),NP(RR,MM),GRDD,GRDVTM(RR,MM)
C     INTEGER I,J,K,P,MAX0,IVV,IPHI,IALFA,IWRK(4*PP),IERR,PP,ORD,L,KK
C     INTEGER JOB,IWRK2(NN),MINORD
C     REAL A(NN,NN),B(NN,MM),C(RR,NN),EPS,EPS2,ERROR,GAIN(RR,MM)
C     REAL VTM(RR,MM,0:NN),D(0:NN),QCOEFF(PP,PP,NN+1),SCOEFF(PP,PP,NN+1)
C     REAL WRK1(NN,NN),WRK2(IPHI),WRK3(IPHI),WRK4(NN,NN),WRK5(NN)
C     REAL WRK6(NN),RCOND,SOM1,SOM2
C     REAL WRK7(NN+1),WRK8(NN+1),WRK9(NN+1,NN+1),WRK10(NN+1),WRK11(0:NN)
C     REAL AA(NNPP,NNPP),BB(NNPP,PP),CC(PP,NNPP),DD(PP,MM),
C    1     WRK12(NN,NN),Q2(IPHI,8*PP)
C     REAL V(IVV,4*PP),WRK17(IPHI,8*PP),WRK18(IPHI,8*PP),Q1(IPHI,8*PP)
C     REAL PHI(IPHI,4*PP),PSI(IPHI,4*PP),ALFA(IALFA,4*PP),BETA(IALFA,4*PP)
C     REAL Z(IALFA,2*PP),WRK19(NN,NN)
C     COMPLEX ZERO(RR,MM,NN),POLE(RR,MM,NN)
C     COMPLEX WRK13(NN),WRK14(NN),WRK15(NN),WRK16(NN)
C
C WITH :
C
```

```
      PP = MAX0(MM,RR)
      IVV = 2*PP*(NN+1)
      IPHI = 2*PP*(2*NN+1)
      IALFA = 4*PP*(NN+1)
      NNPP = NN*PP
C
C NOW IS  NN = 20
C          IVV = 252
C         IPHI = 492
C        IALFA = 504
C
C FORMAT OF THE INPUTFILE FORT.2 :
C
C  ON THE FIRST LINE : N,M,R (FORMAAT 3(I2,1X))
C  THEN A IS GIVEN ROW BY ROW (NOT NECESSARILY 1 ROW PER LINE IN THE
C  FILE). THEN B STARTS ON A NEW LINE AND SUBSEQUENTLY C.(FORMAT 20E14.7)
C  ALL NUMBERS ARE SEPARATED BY COMMAS.
C  ON THE NEXT LINE : 0 OF 1 (FORMAT I1)
C    0 : DISCRETE SYSTEM
C    1 : CONTINUOUS SYSTEM
C  ON THE LAST LINE THE ERROR IN HANKEL-NORM (FORMAT E9.2).
C  THE RESULTS ARE GIVEN IN FORT.11; INTERMEDIATE RESULTS ARE WRITTEN
C  IN FORT.3.
C
      WRITE (3,'("ALGORITHM OF KUNG")')
      READ(2,10)N,M,R
 10   FORMAT(3(I2,1X))
      DO 11 I = 1,N
 11     READ(2,15) (A(I,J),J=1,N)
      DO 12 I = 1,N
 12     READ(2,15) (B(I,J),J=1,M)
      DO 13 I = 1,R
 13     READ(2,15) (C(I,J),J=1,N)
      READ(2,'(I1)')JOB
      READ(2,'(E10.3)')ERROR
      WRITE(3,'(/,"ERROR =",E11.3)')ERROR
   15 FORMAT(20E14.7)
      WRITE(3,35) ((A(I,J),J=1,N),I=1,N)
   35 FORMAT(/,11H MATRIX A :,/,100(/,4E14.7))
      WRITE(3,45) ((B(I,J),J=1,M),I=1,N)
   45 FORMAT(/,11H MATRIX B :,/,50(/,4E14.7))
      WRITE(3,55) ((C(I,J),J=1,N),I=1,R)
   55 FORMAT(/,11H MATRIX C :,/,50(/,4E14.7))
      EPS = 1.0E-05
      WRITE(6,'("EPS2 (TOLERANCE FOR COMPUTATION OF MIN. REALISATION")')
      WRITE(6,'("AND STABLE PART) = ?")')
      READ (5,'(E9.2)')EPS2
      WRITE(3,'(/,"EPS2 =",E10.2)')EPS2
      IF (JOB .EQ. 1)
C
```

```
C MAKE DISCRETE IF NECESSARY
C
     1 CALL BILNTR(A,B,C,NN,NN,RR,N,M,R,.FALSE.,WRK1,WRK4,WRK9,
     2       WRK12,WRK19,WRK2,IWRK2,WRK3,WRK5)
C
      WRITE(10,10)N,M,R
      DO 60 I = 1,N
60      WRITE(10,15)(A(I,J),J=1,N)
      DO 61 I = 1,N
61      WRITE(10,15)(B(I,J),J=1,M)
      DO 62 I = 1,R
62      WRITE(10,15)(C(I,J),J=1,N)
C
C COMPUTATION OF THE TRANSFER FUNCTION
C
      WRITE(3,'(/,"ROUTINE  TRSFCT :",/)')
C
      CALL TRSFCT(A,B,C,NN,MM,RR,N,M,R,EPS,ZERO,POLE,GAIN,NZ,NP,WRK1,
     *            WRK2,WRK3,WRK4,WRK5,WRK6,WRK7,WRK8,WRK9,WRK10,WRK12)
C
      WRITE(3,'("ZEROS OF THE TRANSFER FUNCTION,ROW BY ROW")')
      WRITE(3,'(/)')
      DO 80 I = 1,R
DO 90 J = 1,M
          IF (NZ(I,J).EQ.0)THEN
            WRITE(3,'("NO ZEROS")')
                          ELSE
            WRITE(3,70)(ZERO(I,J,K),K=1,NZ(I,J))
          END IF
90      CONTINUE
80    CONTINUE
      WRITE(3,'(/)')
      WRITE(3,'("POLES OF THE TRANSFER FUNCTION,ROW BY ROW")')
      WRITE(3,'(/)')
      DO 100 I = 1,R
DO 110 J = 1,M
110       WRITE(3,70)(POLE(I,J,K),K=1,NP(I,J))
100   CONTINUE
      WRITE(3,'(/)')
      WRITE(3,'("GAINFACTORS OF THE TRANSFER FUNCTION, ROW BY ROW")')
      WRITE(3,'(/)')
      DO 120 I = 1,R
120     WRITE(3,70)(GAIN(I,J),J=1,M)
70    FORMAT (20(E9.2,1X))
C
C TRANSFER FUNCTION = (1/D(Z))*VTM(Z)
C
      WRITE(3,'(/,"ROUTINE COEFF :",/)')
C
      CALL COEFF(ZERO,POLE,GAIN,MM,RR,N,M,R,NZ,NP,GRDD,D,GRDVTM,VTM,
```

```
     *           WRK13,WRK14,WRK15,WRK16,WRK11)
C
      WRITE(3,'("D(Z) = (INCREASING DEGREE )",/)')
      WRITE(3,130)(D(K),K=0,GRDD)
      WRITE(3,'(/,"VTM(Z) = (ELEMENTS GIVEN ROW BY ROW )",/)')
      DO 140 I = 1,R
        DO 150 J = 1,M
150       WRITE(3,130)(VTM(I,J,K),K=0,NZ(I,J))
140   CONTINUE
130   FORMAT(20E12.4)
C
C LINEAR SYSTEM V(Z)*Z(Z) = 0
C
      WRITE(3,'(/,"ROUTINE STLSL :",/)')
      P = MAX0(M,R)
      DO 200 I = 1,(GRDD+1)*2*P
        DO 210 J = 1,4*P
210       V(I,J) = 0.0
200   CONTINUE
      DO 220 I = 0,GRDD
        DO 230 J = 1,P
          V(2*P*I+J,J) = D(I)
          V(P*(2*I+1)+J,P+J) = ERROR*D(I)
  V(2*P*I+J,2*P+J) = ERROR*D(GRDD-I)
  V(P*(2*I+1)+J,3*P+J) = D(GRDD-I)
230     CONTINUE
220   CONTINUE
      DO 240 I = 1,R
DO 250 J = 1,M
  DO 260 K = 0,GRDD - 1
    IF (P .EQ. M) THEN
      V(M-R+2*P*K+I,P+J) = -VTM(I,J,K)
      V(P*(2*K+3)+J,2*P+M-R+I) = -VTM(I,J,GRDD-K-1)
  ELSE
             V(2*P*K+I,P+R-M+J) = -VTM(I,J,K)
      V(P*(2*K+3)+R-M+J,2*P+I) = -VTM(I,J,GRDD-K-1)
           END IF
260       CONTINUE
250     CONTINUE
240   CONTINUE
C
      CALL STLSL(V,IVV,P,GRDD,WRK17,WRK18,Q1,Q2,PHI,PSI,ALFA,BETA,
     * IWRK,WRK2,WRK3,Z,IALFA)
C
      WRITE (3,'(/,"Z =",/)')
      DO 270 I = 1,4*P*(GRDD+1)
270     WRITE(3,'(20E14.7)')(Z(I,J),J=1,2*P)
      DO 280 I = 1,P
DO 290 J = 1,P
          WRK19(I,J) = Z(P+I,J)
```

```
  WRK1(I,J) = Z(I,J)
  WRK12(I,J) = Z(P+I,P+J)
290     CONTINUE
280   CONTINUE
C
C ALGEBRAIC RICCATI EQUATION
C
      WRITE(3,'(/,"ROUTINE RILAC :",/)')
C
      CALL RILAC(P,2*P,WRK19,NN,WRK1,WRK12,RCOND,WRK1,WRK4,NN+1,WRK9,
     *           IWRK,WRK5,WRK6,IERR)
C
      IF (IERR .NE. 0) THEN
WRITE (6,'("RILAC : IERR =",I2)') IERR
        WRITE (3,'("IERR =",I2)') IERR
STOP
      END IF
      WRITE (3,'("RCOND =",E10.2)') RCOND
      WRITE (3,'(/,"H =",/)')
      DO 300 I = 1,P
300     WRITE (3,'(10E15.7)')(WRK1(I,J),J=1,P)
C
C A LEFT MFD
C
      DO 355 I = 1,P
355     IWRK(I) = 0
      WRITE(3,'(/,/,"QCOEFF,SCOEFF (NUMERATOR- AND DENOMINATOR POLYNOMIAL")')
      WRITE(3,'("OF THE LEFT MFD BEFORE THE COMPUTATION OF THE CORRES-")')
      WRITE(3,'("PONDING REALISATION, ORDERED BY DECREASING POWERS :)",/)')
      DO 360 I = 1,P
        DO 370 J = 1,P
          DO 380 K = 1,GRDD+1
            KK = K - 1
            SOM1 = 0.0
            SOM2 = 0.0
    DO 390 L = 1,P
      SOM1 = SOM1 + WRK1(I,L)*Z(4*P*KK+2*P+J,P+L)
      SOM2 = SOM2 + WRK1(I,L)*Z(4*P*KK+3*P+J,P+L)
390         CONTINUE
    QCOEFF(I,J,K) = Z(4*P*KK+2*P+J,I) - SOM1
    SCOEFF(I,J,K) = Z(4*P*KK+3*P+J,I) - SOM2
380       CONTINUE
          DO 400 K = 1,GRDD+1
            IF (QCOEFF(I,J,K) .GT. EPS) THEN
      IF (GRDD-K+1 .GT. IWRK(I)) IWRK(I) = GRDD-K+1
      GOTO 370
            END IF
400       CONTINUE
370     CONTINUE
        DO 410 J = 1,P
```

```
  DO 420 K = 1,IWRK(I)+1
            QCOEFF(I,J,K) = QCOEFF(I,J,K+GRDD-IWRK(I))
            SCOEFF(I,J,K) = SCOEFF(I,J,K+GRDD-IWRK(I))
420       CONTINUE
410     CONTINUE
360   CONTINUE
      DO 430 I = 1,P
DO 440 J = 1,P
  WRITE(3,'("QCOEFF(",2(I2,","),"K) =",27(1X,E10.3))')
     1         I,J,(QCOEFF(I,J,K),K=1,IWRK(I)+1)
  WRITE(3,'("SCOEFF(",2(I2,","),"K) =",27(1X,E10.3))')
     1         I,J,(SCOEFF(I,J,K),K=1,IWRK(I)+1)
440     CONTINUE
430   CONTINUE
C
C A NON-MINIMAL REALISATION FROM THE LEFT MFD
C
      WRITE(3,'(/,"ROUTINE PMXSS :",/)')
C
      CALL PMXSS(M,R,IWRK,PP,QCOEFF,NN+1,SCOEFF,ORD,AA,NNPP,BB,CC,DD,
     1           RCOND,WRK1,WRK5,IWRK2,0,IERR)
C
      IF (IERR .NE. 0) WRITE(6,'("PMXSS : IERR, RCOND =",I2,1X,E10.2)')
     1                 IERR,RCOND
      WRITE(3,'("RCOND =",E10.2)')RCOND
      WRITE(3,'("ORDER OF THE NON-MINIMAL REALISATION :",I3)')ORD
C
C A MINIMAL REALISATION FROM THE NON-MINIMAL ONE
C
      WRITE(3,'(/,"ROUTINE SSXMR :")')
      CALL SSXMR(ORD,M,R,AA,NNPP,BB,PP,CC,MINORD,KK,IWRK,WRK1,WRK12,
     1           WRK5,WRK6,IWRK2,EPS2,0)
C
C RETURN TO THE ORIGINAL NUMBER OF IN- AND OUTPUTS
C
      IF (M .EQ. P) THEN
DO 310 I = 1,R
  DO 320 J = 1,MINORD
320     CC(I,J) = CC(I+M-R,J)
310     CONTINUE
    ELSE
        DO 330 J = 1,M
  DO 340 I = 1,MINORD
340     BB(I,J) = BB(I,J+R-M)
330     CONTINUE
      END IF
C
      WRITE(3,'(/,"THE REALISATION, NON STABLE",/)')
      WRITE(3,'("MC MILLAN DEGREE =",I3)')MINORD
      WRITE(3,'(/,"AA =",/)')
```

```
      DO 350 I = 1,MINORD
350     WRITE(3,'(20E9.2)')(AA(I,J),J=1,MINORD)
      WRITE(3,'(/,"BB =",/)')
      DO 450 I = 1,MINORD
450     WRITE(3,'(6E9.2)')(BB(I,J),J=1,M)
      WRITE(3,'(/,"CC =",/)')
      DO 460 I = 1,R
460     WRITE(3,'(20E9.2)')(CC(I,J),J=1,MINORD)
C
C STABLE PART
C
      WRITE(3,'(/,"ROUTINE STABL :",/)')
C
      CALL STABL(AA,BB,CC,NNPP,RR,MINORD,M,R,K,WRK5,WRK6,WRK1,WRK4,
     1           WRK9,WRK12,IWRK2,EPS2)
C
      WRITE(11,'(3(I2,1X))')K,M,R
      WRITE(3,'("THE REALISATION, STABLE",/)')
      WRITE(3,'("MC MILLAN DEGREE =",I2)')K
      WRITE(3,'(/,"AA =",/)')
      DO 500 I = 1,K
        WRITE(11,'(20E14.7)')(AA(I,J),J=1,K)
500     WRITE(3,'(20E9.2)')(AA(I,J),J=1,K)
      WRITE(3,'(/,"BB =",/)')
      DO 510 I = 1,K
        WRITE(11,'(6E14.7)')(BB(I,J),J=1,M)
510     WRITE(3,'(6E9.2)')(BB(I,J),J=1,M)
      WRITE(3,'(/,"CC =",/)')
      DO 520 I = 1,R
        WRITE(11,'(20E14.7)')(CC(I,J),J=1,K)
520     WRITE(3,'(20E9.2)')(CC(I,J),J=1,K)
      IF (JOB .EQ. 1) THEN
C
C RETURN TO CONTINUOUS IF NECESSARY
C
        CALL BILNTR(AA,BB,CC,NNPP,NNPP,RR,K,M,R,.FALSE.,WRK1,WRK4,WRK9,
     1              WRK12,WRK19,WRK2,IWRK2,WRK3,WRK5)
C
        WRITE(3,'(/,"CONTINUOUS EQUIVALENT",/)')
        DO 550 I = 1,K
550       WRITE(3,15)(AA(I,J),J=1,K)
        DO 540 I = 1,K
540       WRITE(3,15)(BB(I,J),J=1,M)
        DO 530 I = 1,R
530       WRITE(3,15)(CC(I,J),J=1,K)
      END IF
      STOP
      END
```

```
      SUBROUTINE COEFF(ZERO,POLE,GAIN,MM,RR,N,M,R,NZ,NP,GRDD,D,GRDVTM,
     *                 VTM,ZEROD,VER,WRK1,WRK2,WRK3)
C
C WRITE THE TRANSFERFUNCTION MATRIX, GIVEN BY ITS ZEROS, POLES AND
C ITS GAINFACTORS, AS
C                          1
C                        ---- * VTM(Z)
C                        D(Z)
C WITH D(Z) A POLYNOMIAL
C AND VTM(Z) A POLYNOMIAL MATRIX
C
      INTEGER MM,RR,N,M,R,NZ(RR,M),NP(RR,M),GRDD,GRDVTM(RR,M)
      REAL GAIN(RR,M),D(0:*),VTM(RR,MM,0:*),WRK3(0:*)
      COMPLEX ZERO(RR,MM,*),POLE(RR,MM,*),ZEROD(*),VER(*)
      COMPLEX WRK1(*),WRK2(*)
C
C INPUT
C
C  ZERO,POLE,GAIN : ZEROS, POLES AND GAINFACTORS, ROW BY ROW
C     RR,MM : MAX. NUMBER OF ROWS,RESP. COLUMNS OF THE TRANSFER
C             FUNCTION MATRIX (SEE DECLARATIONS)
C     N,M,R : NUMBER OF STATES, INPUTS AND OUTPUTS
C     NZ,NP : MATRICES CONTAIN, EL. BY EL., THE NUMBER OF ZEROS AND POLES
C
C OUTPUT
C
C IN D EN VTM THE COEFFICIENTS ARE ORDERED WITH INCREASING
C POWERS OF Z
C THE DEGREES OF THE POLYNOMIALS ARE GIVEN BY GRDD AND GRDVTM
C ZEROD : ZEROS OF D
C VER,WRK1,WRK2,WRK3 : WORKSPACE
C
C SUBROUTINES REQUIRED : VERSCH,PROD
C
      INTEGER I,J,K,GRDVER
C
C LOOK FOR THE LEAST COMMON MULTIPLE OF THE DENOMINATORS
C
      GRDD = 0
      DO 20 I = 1,R
DO 30 J = 1,M
  DO 10 K = 1,NP(I,J)
10          WRK1(K) = POLE(I,J,K)
  CALL VERSCH(NP(I,J),WRK1,GRDD,ZEROD,GRDVER,VER,WRK2)
  DO 40 K = 1,GRDVER
40          ZEROD(GRDD+K) = VER(K)
  GRDD = GRDD + GRDVER
30      CONTINUE
20    CONTINUE
C
```

```
C ADAPT THE ZEROS OF THE NUMERATORS
C
      DO 50 I = 1,R
DO 60 J = 1,M
  DO 70 K =1,NP(I,J)
70          WRK1(K) = POLE(I,J,K)
          CALL VERSCH(GRDD,ZEROD,NP(I,J),WRK1,GRDVER,VER,WRK2)
  DO 80 K = 1,GRDVER
80          ZERO(I,J,NZ(I,J)+K) = VER(K)
  NZ(I,J) = NZ(I,J) + GRDVER
60      CONTINUE
50    CONTINUE
C
C COMPUTE THE COEFFICIENTS
C
      CALL PROD(GRDD,ZEROD,D)
      DO 90 I = 1,R
DO 100 J = 1,M
  DO 110 K = 1,NZ(I,J)
110         WRK1(K) = ZERO(I,J,K)
  CALL PROD(NZ(I,J),WRK1,WRK3)
  DO 120 K = 0,NZ(I,J)
120         VTM(I,J,K) = WRK3(K)*GAIN(I,J)
100     CONTINUE
90    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CONTR(A,B,C,NN,N,EPS,RANK,AB,U,HLPVEC)
C
C COMPUTE THE CONTROLLABLE PART OF A SISO-SYSTEM OF ORDER N
C GIVEN BY A REALISATION A,B,C
C
      INTEGER NN,N,RANK
      REAL A(NN,N),B(N),C(N),EPS,AB(NN+1,N+1),U(N),HLPVEC(N)
C
C INPUT
C
C   A,B,C,N
C        NN : DECLARED ROWDIMENSION OF A IN THE CALLING PROGRAM
C       EPS : RELATIVE ACCURACY OF THE DATA
C
C OUTPUT
C
C   THE CONTROLLABLE PART IS GIVEN BY THE FIRST RANK ROWS AND
C   COLUMNS OF A AND THE FIRST RANK ELEMENTS OF B AND C
C   AB,U,HLPVEC : WORKSPACE
C
C SUBROUTINES REQUIRED : HOUSH, HSHTRF
C
      INTEGER I,J,K,L
      REAL ABS,MAX
C
C BUILD THE MATRIX AB := [A|B]
C
      DO 4 I = 1,N
AB (I,N+1) = B(I)
DO 6 J = 1,N
  AB(I,J) = A(I,J)
6       CONTINUE
4     CONTINUE
C
      RANK = 0
C
C TEST IF B = 0
C
      MAX = ABS(AB(1,N+1))
      DO 10 I = 2,N
IF (ABS(AB(I,N+1)) .GT. MAX) MAX = ABS(AB(I,N+1))
10    CONTINUE
      IF (MAX .LT. EPS) RETURN
C
      RANK = RANK + 1
C
      IF (N .EQ. 1) RETURN
C
C LOOP OVER ALL COLUMNS
C
```

```
      DO 14 L = N+1,3,-1
C
C FIND THE HOUSEHOLDER TRANSFORMATION Q THAT MAKES COLUMN L OF AB ZERO
C
        DO 16 K = 1,L-1
16        HLPVEC(K) = AB(K,L)
CALL HOUSH (HLPVEC,1,L-1,.FALSE.,U)
C
C A := Q*A ; B := Q*B
C
DO 20 J = 1,L
  DO 30 K = 1,L-1
30          HLPVEC(K) = AB(K,J)
  CALL HSHTRF(HLPVEC,1,L-1,U)
  DO 40 K = 1,L-1
40          AB(K,J) = HLPVEC(K)
20      CONTINUE
C
C A := A*TRANSP(Q) = TRANSP(Q*TRANSP(A))
C
        DO 50 J = 1,N
  DO 60 K = 1,L-1
60          HLPVEC(K) = AB(J,K)
  CALL HSHTRF(HLPVEC,1,L-1,U)
  DO 70 K = 1,L-1
70          AB(J,K) = HLPVEC(K)
50      CONTINUE
C
C  T     T  T
C C  := C *Q
C
CALL HSHTRF(C,1,L-1,U)
C
C TEST PREVIOUS COLUMN
C
        MAX = ABS(AB(1,L-1))
DO 80 I = 2,L-2
  IF (ABS(AB(I,L-1)) .GT. MAX) MAX = ABS(AB(I,L-1))
80      CONTINUE
IF (MAX .LT. EPS) GOTO 990
C
RANK = RANK + 1
C
14    CONTINUE
C
C SET A, B AND C = THE CONTROLLABLE PART
C
990   CONTINUE
      DO 90 I = 1,RANK
B(I) = AB(N-RANK+I,N+1)
```

```
C(I) = C(N-RANK+I)
DO 100 J = 1,RANK
100       A(I,J) = AB(N-RANK+I,N-RANK+J)
90    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE HOUSH(B,I,J,FIRST,U)
C
C FINDS A HOUSEHOLDER TRANSFORMATION TO REDUCE A VECTOR TO ITS 1ST OR ITS
C LAST COMPONENT.
C MORE SPECIFICALY THE VECTOR U IS FOUND IN THE HOUSEHOLDER TRANSFORMATION
C
C                     T
C                   U*U
C         I - 2 * -----
C                    T
C                   U *U
C
      INTEGER I,J
      LOGICAL FIRST
      REAL B(*),U(*)
C
C INPUT
C
C         B : VECTOR TO BE REDUCED (UNCHANGED ON RETURN)
C I,J,FIRST : ONLY THE COMPONENTS B(I),...,B(J) ARE CHANGED.
C       IF FIRST = .TRUE. THEN B(I+1),...,B(J) ARE MADE ZERO.
C       IF FIRST = .FALSE. THEN B(I),...,B(J-1) ARE MADE ZERO.
C
C OUTPUT
C
C         U : VECTOR DEFINING THE TRANSFORMATION
C
      INTEGER K
      REAL NORM2,NORM,SQRT,SIGN
C
C COMPUTE U
C
      NORM2 = 0.0
      DO 10 K = I,J
U(K) = B(K)
NORM2 = NORM2 + B(K)**2
 10   CONTINUE
      NORM = SQRT(NORM2)
      IF (FIRST .EQ. .TRUE.) THEN
        K=I
                             ELSE
        K=J
      END IF
      IF (B(K) .EQ. 0.0) THEN
U(K) = U(K) + NORM
                   ELSE
        U(K) = U(K) + SIGN(NORM,B(K))
      END IF
      RETURN
```

```
END
```

```
      SUBROUTINE HSHTRF(B,I,J,U)
C
C APPLIES A HOUSEHOLDER TRANSFORMATION TO A PART OF A VECTOR B
C
      INTEGER I,J
      REAL B(*),U(*)
C
C INPUT
C
C    B : VECTOR TO BE TRANSFORMED
C  I,J : FIRST,RESP. LAST INDEX OF THE PART OF B TO BE TRANSFORMED
C    U : VECTOR DEFINING THE TRANSFORMATION TO BE APPLIED
C        (SEE ROUTINE HOUSH)
C
C OUTPUT
C
C    B : CONTAINS GETRANSFORMED VECTOR
C
      INTEGER K
      REAL UTU,FACTOR
      UTU = 0.0
      FACTOR = 0.0
      DO 10 K = I,J
UTU = UTU + U(K)**2
FACTOR = FACTOR+U(K)*B(K)
 10   CONTINUE
      FACTOR = FACTOR*2.0/UTU
      DO 20 K = I,J
B(K) = B(K) - FACTOR*U(K)
 20   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PROD(GRD,NUL,A)
C
C COMPUTES THE COEFFICIENTS OF A POLYNOMIAL FROM ITS ZEROS
C
      INTEGER GRD
      REAL A(0:GRD)
      COMPLEX NUL(GRD)
C
C INPUT
C
C GRD : NUMBER OF ZEROS
C NUL : ROW OF ZEROS
C
C OUTPUT
C
C   A : COEFFICIENTS FOR INCREASING EXPONENTS
C
      INTEGER GRDA,I,J
      REAL RL,IM,NORM2,REAL,AIMAG
      COMPLEX NUL(GRD)
      GRDA = 0.0
      A(0) = 1.0
      I = 1
10    IF (I .GT. GRD) RETURN
RL = REAL(NUL(I))
IM = AIMAG(NUL(I))
IF (IM .EQ. 0.0) THEN
  A(GRDA+1) = 1.0
  DO 20 J = GRDA,1,-1
20          A(J) = A(J-1) - NUL(I)*A(J)
  A(0) = -NUL(I)*A(0)
  GRDA = GRDA + 1
  I = I+1
 ELSE
  NORM2 = RL**2 + IM**2
  A(GRDA + 2) = 1.0
  A(GRDA + 1) = A(GRDA - 1) - 2*RL*A(GRDA)
  DO 30 J = GRDA,2,-1
30     A(J) = A(J-2) - 2*RL*A(J-1) + NORM2*A(J)
          A(1) = -2*RL*A(0) + NORM2*A(1)
  A(0) = NORM2*A(0)
  GRDA = GRDA + 2
  I = I + 2
       END IF
GOTO 10
       END
```

```
      SUBROUTINE STABL(AA,BB,CC,NN,RR,ORD,M,R,K,WRK1,WRK2,WRK3,WRK4,
     1                 WRK5,WRK6,IWRK,EPS)
C
C ISOLATES THE STABLE PART OF A LINEAR SYSTEM GIVEN BY ITS STATE SPACE
C DESCRIPTION. THIS ROUTINE IS BASED ON A PART OF THE ROUTINE GLOVER
C FROM THE SYCOT LIBRARY. THIS PART IS ADAPTED TO BE CONSISTENT WITH
C THE AVAILABLE ROUTINES AND MEMORY SPACE. IT IS CONVERTED TO THE CASE
C OF DISCRETE TIME SYSTEMS.
C
      INTEGER IWRK(ORD),I,J,K,L,II,JJ,LL,NN,RR,ORD,M,R,IERR
      REAL AA(NN,ORD),BB(NN,M),CC(RR,ORD),EPS
      REAL WRK1(ORD),WRK2(ORD),WRK3(NN,ORD),WRK4(NN,ORD),WRK5(NN,ORD)
C
C INPUT
C
C AA,BB,CC : GIVEN SYSTEM
C       NN : DECLARED FIRST DIMENSION OF AA AND BB
C       RR : DECLARED FIRST DIMENSION OF CC
C      ORD : ORDER OF AA
C      M,R : NUMBER OF INPUTS AND OUTPUTS OF THE SYSTEM
C      EPS : TOLERANCE ON THE DATA
C
C OUTPUT
C
C        K : ORDER OF THE STABLE PART
C AA,BB,CC : CONTAIN ON RETURN THE STABLE PART (ORDER(AA) = KXK,
C      ORDER(BB)=KXM, ORDER(CC)=RXK)
C
C WORKING SPACE
C
C WRK1,WRK2,WRK3,WRK4,WRK5,WRK6,IWRK
C
C SUBROUTINES REQUIRED : QTRORT (SLICE)
C  SORT, SHRSLV (SYCOT)
C
      REAL WRK6(NN,ORD)
      CALL QTRORT(ORD,AA,NN,WRK3,WRK1,ORD,0,IERR)
      IF (IERR .EQ. 0) GOTO 200
      WRITE(6,175) IERR
  175 FORMAT(49H ERROR SUBROUTINE STABL : REDUCTION OF AA TO REAL,
     X       25H UPPER SCHUR FORM FAILED.,I2)
      STOP
  200 CALL SORT(AA,NN,WRK3,NN,ORD,WRK1,WRK2,K,IWRK,EPS)
C
C UPDATE BB AND CC  ( TRANSP(WRK3)*BB AND CC*WRK3 )
C
      DO 240 I=1,R
        DO 220 J=1,ORD
          WRK2(J)=0.0
          DO 210 L=1,ORD
```

```
            WRK2(J)=WRK2(J)+CC(I,L)*WRK3(L,J)
  210     CONTINUE
  220   CONTINUE
        DO 230 J=1,ORD
          CC(I,J)=WRK2(J)
  230   CONTINUE
  240 CONTINUE
      DO 280 J=1,M
        DO 260 I=1,ORD
          WRK2(I)=0.0
          DO 250 L=1,ORD
            WRK2(I)=WRK2(I)+WRK3(L,I)*BB(L,J)
  250     CONTINUE
  260   CONTINUE
        DO 270 I=1,ORD
          BB(I,J)=WRK2(I)
  270   CONTINUE
  280 CONTINUE
      IF (K.EQ.ORD) GOTO 400
C
C  TRANSFORM AA TO BLOCK DIAGONAL FORM.
C
      DO 310 I=1,K
        DO 290 J=1,K
          WRK5(I,J)=AA(I,J)
  290   CONTINUE
        DO 300 J=1,ORD-K
          JJ=J+K
          WRK6(I,J)=-AA(I,JJ)
          AA(I,JJ)=0.0
  300   CONTINUE
  310 CONTINUE
      DO 330 I=1,ORD-K
        DO 320 J=1,ORD-K
          II=I+K
          JJ=J+K
          WRK4(I,J)=-AA(II,JJ)
  320   CONTINUE
  330 CONTINUE
      CALL SHRSLV(WRK5,WRK4,WRK6,NN,K,ORD-K)
C
C  UPDATE BB,CC,T WITH THE TRANSFORMATIONMATRIX (I WRK6)
C                                               (0 I   )
C
      DO 360 I=1,K
        DO 350 J=1,M
          DO 340 L=1,ORD-K
            LL=L+K
            BB(I,J)=BB(I,J)-WRK6(I,L)*BB(LL,J)
  340     CONTINUE
```

```
350   CONTINUE
360 CONTINUE
    DO 390 I=1,R
      DO 380 J=1,ORD - K
        JJ=J+K
        DO 370 L=1,K
          CC(I,JJ)=CC(I,JJ)+CC(I,L)*WRK6(L,J)
370     CONTINUE
380   CONTINUE
390 CONTINUE
    GOTO 400
395 WRITE(6,399)
399 FORMAT(50H ERROR SUBROUTINE STABL : SIGMA(K+1) IS NOT SINGLE)
    STOP
400 RETURN
    END
```

```
      SUBROUTINE STLSL(V,IVV,P,GRDD,WRK1,WRK2,Q1,Q2,PHI,PSI,ALFA,BETA,
     *        IPVT,HLPVEC,U,Z,IZ)
      INTEGER IVV,P,GRDD,IPVT(4*P),IZ
      REAL V(IVV,4*P),WRK1(2*P*(2*GRDD+1),8*P),WRK2(2*P*(2*GRDD+1),8*P)
      REAL Q1(2*P*(2*GRDD+1),8*P),Q2(2*P*(2*GRDD+1),8*P)
      REAL PHI(2*P*(2*GRDD+1),4*P)
      REAL PSI(2*P*(2*GRDD+1),4*P),ALFA(4*P*(GRDD+1),4*P)
      REAL BETA(4*P*(GRDD+1),4*P),HLPVEC(2*P*(2*GRDD+1))
      REAL U(2*P*(2*GRDD+1)),Z(IZ,2*P)
C
C SOLVES A POLYNOMIAL MATRIX EQUATION V(Z)*Z(Z) = 0
C BY THE PROJECTION METHOD OF KUNG AND KAILATH (AUTOMATICA 16(1980) 399-403)
C SOME DETAILS ARE FOUND IN J. DECORTE ET AL REPORT TW81, 1986
C COMPUTER SCIENCE DEPT. K.U. LEUVEN
C
C INPUT
C  V : V(Z) IN THE FORM
C
C         V0
C         V1
C         .
C         .
C         .
C         VGRDD
C
C WITH VI V(Z)= SUM VI * Z ** I
C
C IVV  : DECLARED FIRST DIMENSION OF V IN THE CALLING PROGRAM
C   P  : V(Z) HAS DIMENSION (2P X 4P) AND Z(Z) HAS DIMENSION (4P X 2P)
C GRDD : DEGREE OF V(Z) AND Z(Z)
C   IZ : DECLARED FIRST DIMENSION OF Z IN THE CALLING PROGRAM
C
C OUTPUT
C
C   Z  : THE SOLUTION IN THE SAME FORM AS V
C
C WORKING SPACE
C
C  WRK1,WRK2,Q1,Q2,PHI,PSI,ALFA,BETA,
C
C SUBROUTINES REQUIRED :
C UPTR0, UPTR
C STRSL, STRDI, SGECO, SGESL (LINPACK)
C
      INTEGER IV,I,J,K,L,IWRK,INFO
      REAL SOM1,SOM2,DET(2),RCOND
C
C COMPUTE PHI(0) = PSI(0) AND ALFA(0) = BETA(0)
C
      IV = 2*P*(GRDD+1)
```

```
      IWRK = 2*P*(2*GRDD+1)
      DO 3 I = 1,IWRK
        DO 4 J = 1,8*P
          WRK1(I,J) = 0.0
          WRK2(I,J) = 0.0
4       CONTINUE
3     CONTINUE
      CALL UPTRO(V,IVV,IV,4*P,Q1,IWRK,HLPVEC,U)
      CALL STRDI(Q1,IWRK,4*P,DET,111,INFO)
      IF (INFO .NE. 0) THEN
WRITE(6,'("INVERSE OF A SINGULAR TRIANGULAR MATRIX ")')
        WRITE(6,'("WHEN COMPUTING ALFA(0)")')
        WRITE(6,'("DIAGONAL ELEMENT NR.",I3," = 0")')INFO
        STOP
      END IF
      WRITE(3,'("COMPUTATION OF ALFA(0), DET =",E10.2,"* 10**",E9.2)')
     1      DET(1),DET(2)
      DO 10 I = 1,IV
DO 20 J = 1,4*P
  SOM1 = 0.0
  DO 30 K = 1,4*P
30          SOM1 = SOM1 + V(I,K)*Q1(K,J)
  PHI(I,J) = SOM1
  PSI(I,J) = SOM1
20      CONTINUE
10    CONTINUE
      DO 40 I = 1,4*P
DO 50 J =  1,4*P
  ALFA(I,J) = Q1(I,J)
  BETA(I,J) = Q1(I,J)
50      CONTINUE
40    CONTINUE
C
C LOOP FOR THE RECURSION
C
      DO 15 I = 1,GRDD-1
C
C COMPUTE THE CONSTANT MATRICES FROM THE RECURENCE RELATIONS
C
CALL UPTR(PSI,PHI,.FALSE.,I,P,GRDD,Q1,HLPVEC,U)
CALL STRDI(Q1,IWRK,8*P,DET,111,INFO)
        IF (INFO .NE. 0) THEN
  WRITE(6,'("INVERSE OF A SINGULAR TRIANGULAR MATRIX ")')
          WRITE(6,'("WHEN COMPUTING ALFA(",I2,")")')I
          WRITE(6,'("DIAGONAL ELEMENT NR.",I3," = 0")')INFO
          STOP
        END IF
WRITE(3,
     1  '("COMPUTATION OF ALFA(",I2,"), DET =",E10.2,"* 10**",E9.2)')
     2  I,DET(1),DET(2)
```

```
CALL UPTR(PHI,PSI,.TRUE.,I,P,GRDD,Q2,HLPVEC,U)
CALL STRDI(Q2,IWRK,8*P,DET,111,INFO)
        IF (INFO .NE. 0) THEN
  WRITE(6,'("INVERSE OF A SINGULAR TRIANGULAR MATRIX ")')
          WRITE(6,'("WHEN COMPUTING BETA(",I2,")")')I
          WRITE(6,'("DIAGONAL ELEMENT NR.",I3," = 0")')INFO
          STOP
        END IF
        WRITE(3,
     1  '("COMPUTATION OF BETA(",I2,"), DET =",E10.2,"* 10**",E9.2)')
     2  I,DET(1),DET(2)
C
C COMPUTE THE NEW PHI AND PSI
C
DO 25 J = 1,2*P*(GRDD+I)
  DO 5 K = 1,4*P
    SOM1 = 0.0
    SOM2 = 0.0
    DO 45 L = 1,4*P
      SOM1 = SOM1 + PSI(J,L)*Q1(L,4*P+K)
      SOM2 = SOM2+PHI(J,L)*Q2(L,4*P+K)
45          CONTINUE
    WRK1(J,K) = WRK1(J,K) + SOM1
    WRK2(2*P+J,K) = WRK2(2*P+J,K) + SOM2
    SOM1 = 0.0
    SOM2 = 0.0
    DO 55 L = 1,K
      SOM1 = SOM1 + PHI(J,L)*Q1(4*P+L,4*P+K)
               SOM2 = SOM2 + PSI(J,L)*Q2(4*P+L,4*P+K)
55          CONTINUE
    WRK1(2*P+J,K) = WRK1(2*P+J,K) + SOM1
    WRK2(J,K) = WRK2(J,K) + SOM2
5         CONTINUE
25      CONTINUE
DO 60 J = 1,2*P*(GRDD+I+1)
  DO 70 K = 1,4*P
            PHI(J,K) = WRK1(J,K)
    WRK1(J,K) = 0.0
    PSI(J,K) = WRK2(J,K)
    WRK2(J,K) = 0.0
70        CONTINUE
60      CONTINUE
C
C COMPUTE THE NEW ALFA AND BETA
C
DO 80 J = 1,4*P*I
  DO 90 K = 1,4*P
    SOM1 = 0.0
    SOM2 = 0.0
    DO 100 L = 1,4*P
```

```
      SOM1 = SOM1 + BETA(J,L)*Q1(L,4*P+K)
      SOM2 = SOM2 + ALFA(J,L)*Q2(L,4*P+K)
100         CONTINUE
    WRK1(J,K) = WRK1(J,K) + SOM1
    WRK2(4*P+J,K) = WRK2(4*P+J,K) + SOM2
    SOM1 = 0.0
    SOM2 = 0.0
    DO 110 L = 1,K
      SOM1 = SOM1 + ALFA(J,L)*Q1(4*P+L,4*P+K)
              SOM2 = SOM2 + BETA(J,L)*Q2(4*P+L,4*P+K)
110         CONTINUE
    WRK1(4*P+J,K) = WRK1(4*P+J,K) + SOM1
    WRK2(J,K) = WRK2(J,K) + SOM2
90        CONTINUE
80      CONTINUE
DO 120 J = 1,4*P*(I+1)
  DO 130 K = 1,4*P
            ALFA(J,K) = WRK1(J,K)
    WRK1(J,K) = 0.0
    BETA(J,K) = WRK2(J,K)
    WRK2(J,K) = 0.0
130       CONTINUE
120     CONTINUE
C
C END LOOP FOR THE RECURSION
C
15    CONTINUE
C
C COMPUTATION OF ALFA(GRDD) IN DESIRED FORM
C
      CALL UPTR(PSI,PHI,.FALSE.,I,P,GRDD,Q1,HLPVEC,U)
      DO 29 I = 1,2*P
DO 31 J = 1,2*P
          WRK2(I,J) = ALFA(2*P*(2*GRDD-1)+I,J)
  WRK2(I,2*P+J) = ALFA(2*P*(2*GRDD-1)+I,2*P+J)
  WRK2(2*P+I,J) = Q1(4*P+I,4*P+J)
  WRK2(2*P+I,2*P+J) = Q1(4*P+I,6*P+J)
31      CONTINUE
29    CONTINUE
      CALL SGECO(WRK2,IWRK,4*P,IPVT,RCOND,HLPVEC)
      IF (1.0+RCOND .EQ. RCOND) THEN
WRITE(6,'("SINGULAR SYSTEM WHEN COMPUTING ")')
        WRITE(6,'("Z, RCOND =",E10.2)')RCOND
        STOP
      END IF
      WRITE(3,'("COMPUTATION OF Z, FACTORISATION OF THE COMPLETE ")')
      WRITE(3,'("MATRIX, RCOND =",E10.2)')RCOND
      DO 33 I = 1,2*P
C
C COMPUTATION OF THE I-TH COLUMN OF THE SOLUTION Z
```

```
C
DO 32 J = 1,4*P
32        HLPVEC(J) = 0.0
        HLPVEC(I) = 1.0
CALL SGESL(WRK2,IWRK,2*P+I,IPVT,HLPVEC,0)
DO 34 J = 1,4*P
  SOM1 = 0.0
  DO 35 K = 1,2*P+I
35          SOM1 = SOM1 - Q1(J,4*P+K)*HLPVEC(K)
  U(J) = SOM1
34      CONTINUE
CALL STRSL(Q1,IWRK,4*P,U,01,INFO)
IF (INFO .NE. 0) THEN
  WRITE(6,'("SINGULAR TRIANGULAR SYSTEM WHEN COMPUTING ")')
          WRITE(6,'("Z(*,",I2,"), DIAGONAL ELEMENT NR.",I3," = 0")')
     *          I,INFO
          STOP
END IF
WRITE(3,'("SOLVE THE TRIANGULAR SYSTEM FOR THE")')
WRITE(3,'("COMPUTATION OF Z, DIAGONAL ELEMENTS =")')
        WRITE(3,'(24E8.1)')(Q1(J,J),J=1,4*P)
        DO 36 J = 1,4*P*GRDD
  SOM1 = 0.0
  DO 37 K = 1,4*P
37          SOM1 = SOM1 + BETA(J,K)*U(K)
  Z(J,I) = SOM1
36      CONTINUE
        DO 38 J = 1,2*P*(2*GRDD-1)
  SOM1 = 0.0
  DO 39 K = 1,2*P+I
39          SOM1 = SOM1 + ALFA(J,K)*HLPVEC(K)
  Z(4*P+J,I) = Z(4*P+J,I) + SOM1
38      CONTINUE
33    CONTINUE
      DO 340 I = 1,2*P
DO 350 J = 1,2*P
350   Z(2*P*(2*GRDD+1)+I,J) = 0.0
        Z(2*P*(2*GRDD+1)+I,I) =1.0
340   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE TRSFCT(A,B,C,NN,MM,RR,N,M,R,EPS,ZERO,POLE,GAIN,NZ,NP,
     *                  AA,BB,CC,V,U,HLPVEC,WR,WI,W,BETA,Z)
C
C COMPUTES THE ZEROS,POLES AND GAINFACTORS OF A (NOT NECESSARILY
C MINIMAL) REALISATION OF A LINEAR SYSTEM
C
      INTEGER NN,MM,RR,N,M,R,NZ(RR,M),NP(RR,M)
      REAL A(NN,N),B(NN,M),C(RR,N),EPS,GAIN(RR,M)
      REAL AA(NN,N),BB(N),CC(N),BETA(N+1),Z(NN,N)
      REAL WR(N+1),WI(N+1),V(NN+1,N+1),W(NN+1,N+1),HLPVEC(N),U(N)
      COMPLEX ZERO(RR,MM,N-1),POLE(RR,MM,N)
C
C INPUT
C
C A,B,C : THE GIVEN REALISATION
C     N : ORDER OF THE SYSTEM
C   M,R : NUMBER OF INPUTS, RESP.OUTPUTS
C    NN : DECLARED 1ST DIMENSION OF A AND B IN THE CALLING PROGRAM
C    MM : DECLARED 2ND DIMENSION OF ZERO AND POLE IN THE CALLING
C         PROGRAM
C    RR : DECLARED 1ST DIMENSION OF ZERO,POLE AND GAIN IN THE
C         CALLING PROGRAM
C   EPS : RELATIVE ACCURACY OF THE DATA
C
C OUTPUT
C
C ZERO,POLE,GAIN : CONTAIN RESP. THE ZEROS,POLES AND GAIN FACTORS
C                  OF EACH ELEMENT IN THE TRANSFER FUNCTION
C          NZ,NP : CONTAIN THE NUMBER OF ZEROS,RESP. POLES OF EACH
C                  ELEMENT IN THE TRANSFER FUNCTION
C AA,BB,CC,,WR,WI,HLPVEC,U,V,W,BETA,Z : WORKING SPACE
C
C REQUIRED SUBROUTINES : HOUSH,HSHTRF,CONTR
C                        HQR,QZIT,QZVAL (EISPACK)
C
C METHOD USED : VARGA AND SIMA, NUMERICALLY STABLE ALGORITHM
C FOR TRANSFER FUNCTION MATRIX EVALUATION, INTERNATIONAL JOURNAL CONTROL,
C 1981,VOL.33,NO.6,P,1123 - 1133
C
      INTEGER I,J,JJ,KK,K,L,RANK,RANK2,P,IERR,NNZ
      REAL HULP,PROD,RL,IM,Z0,CABS
      COMPLEX CMPLX
      REAL REAL,AIMAG
      DO 10 I = 1,R
DO 20 J = 1,M
C
C COMPUTATION OF THE ELEMENT IN THE I-TH ROW AND THE J-TH COLUMN OF
C THE TRANSFER FUNCTION
C
  DO 24 JJ = 1,N
```

```
    BB(JJ) = B(JJ,J)
            CC(JJ) = C(I,JJ)
    DO 26 KK = 1,N
26       AA(JJ,KK) = A(JJ,KK)
24        CONTINUE
C
C COMPUTE THE CONTROLLABLE PART
C
          CALL CONTR(AA,BB,CC,NN,N,EPS,RANK,V,U,HLPVEC)
C
C COMPUTE THE OBSERVABLE PART
C
C TRANSPOSE OF THE CONTROLLABLE PART OF AA
C
  DO 30 K = 1,RANK
    DO 40 L = K,RANK
      HULP = AA(K,L)
      AA(K,L) = AA(L,K)
      AA(L,K) = HULP
40          CONTINUE
30        CONTINUE
C
C COMPUTE THE CONTROLLABLE PART OF THE DUAL SYSTEM
C
  CALL CONTR(AA,CC,BB,NN,RANK,EPS,RANK2,V,U,HLPVEC)
C
C TRANSPOSE OF THE CONTROLLABLE AND OBSERVABLE PART
C
          DO 50 K = 1,RANK2
            DO 60 L = K,RANK2
              HULP = AA(K,L)
              AA(K,L) = AA(L,K)
              AA(L,K) = HULP
60          CONTINUE
50        CONTINUE
C
C SAVE AA FOR FURTHER COMPUTATIONS
C
          DO 15 JJ = 1,N
            DO 16 KK = 1,N
16            Z(JJ,KK) = AA(JJ,KK)
15        CONTINUE
C
C COMPUTE THE POLES
C
  CALL HQR(NN,RANK2,1,RANK2,AA,WR,WI,IERR)
  IF (IERR.NE.0) THEN
    WRITE(6,'("NOT ALL THE POLES ARE FOUND")')
    STOP
          END IF
```

```
  NP(I,J) = RANK2
          Z0 = 0.0
  DO 45 K = 1,RANK2
    POLE(I,J,K) = CMPLX(WR(K),WI(K))
            IF (CABS(POLE(I,J,K)) .GT. Z0) Z0 = CABS(POLE(I,J,K))
45        CONTINUE
C
C COMPUTE THE ZEROS
C
C FORM THE MATRICES OF THE GENERALIZED EIGENVALUE PROBLEM
C
  DO 51 K = 1,RANK2
    DO 52 L = 1,RANK2
      V(K,L) = Z(K,L)
      W(K,L) = 0.0
52          CONTINUE
    V(K,RANK2+1) = BB(K)
    W(K,K) = 1.0
    W(K,RANK2+1) = 0.0
51        CONTINUE
  DO 53 K = 1,RANK2
    V(RANK2+1,K) = CC(K)
    W(RANK2+1,K) = 0.0
53        CONTINUE
          V(RANK2+1,RANK2+1) = 0.0
          W(RANK2+1,RANK2+1) = 0.0
C
C QZ-ALGORITHM
C
          CALL QZIT(NN+1,RANK2+1,V,W,0.0,.FALSE.,V,IERR)
          IF (IERR .NE. 0)THEN
            WRITE(6,'("NOT ALL THE ZEROS ARE FOUND")')
            STOP
          END IF
  CALL QZVAL(NN+1,RANK2+1,V,W,WR,WI,BETA,.FALSE.,V)
C
C COMPUTE THE GENERALIZED EIGENVALUES
C
  NNZ = 0
  DO 48 K = 1,RANK2+1
    IF (BETA(K).GE.EPS) THEN
              NNZ = NNZ+1
      ZERO (I,J,NNZ) = CMPLX(WR(K)/BETA(K),WI(K)/BETA(K))
              IF (CABS(ZERO(I,J,NNZ)) .GT. Z0) Z0 = CABS(ZERO(I,J,NNZ))
            END IF
  48      CONTINUE
          Z0 = Z0 + 5.0
          NZ(I,J) = NNZ
C
C COMPUTE THE CONSTANT FACTOR
```

```
C
C FORM (Z0*I - Z)
C
          DO 54 K = 1,RANK2
    DO 58 L = 1,RANK2
      Z(K,L) = -Z(K,L)
58          CONTINUE
54        CONTINUE
  DO 70 K = 1,RANK2
70          Z(K,K) = Z0 + Z(K,K)
  IF (RANK2 .GT. 1) THEN
C
C MAKE Z UPPER TRIANGULAR
C
    DO 80 K = 1,RANK2-1
      DO 90 L = 1,RANK2
90              HLPVEC(L) = Z(L,K)
      CALL HOUSH(HLPVEC,K,K+1,.TRUE.,U)
      DO 100 P = K,RANK2
DO 110 L = K,K+1
110               HLPVEC(L) = Z(L,P)
CALL HSHTRF(HLPVEC,K,K+1,U)
DO 120 L = K,K+1
120               Z(L,P) = HLPVEC(L)
100           CONTINUE
      CALL HSHTRF(BB,K,K+1,U)
80          CONTINUE
  END IF
C
C COMPUTE THE GAIN FACTOR GAIN(I,J)
C
  PROD = BB(RANK2)*CC(RANK2)/Z(RANK2,RANK2)
  K = 1
130       IF (K .GT. RANK2) GOTO 140
    RL = REAL(POLE(I,J,K))
    IM = AIMAG(POLE(I,J,K))
    IF (IM .EQ. 0.0) THEN
      PROD = PROD*(Z0 - RL)
      K = K+1
     ELSE
               PROD = PROD*(Z0**2-2.0*Z0*RL+RL**2+IM**2)
               K=K+2
            END IF
          GOTO 130
140       K = 1
160       IF (K .GT. NNZ) GOTO 150
    RL = REAL(ZERO(I,J,K))
          IM = AIMAG(ZERO(I,J,K))
    IF (IM .EQ. 0.0) THEN
      PROD = PROD/(Z0-RL)
```

```
     K = K+1
    ELSE
             PROD = PROD/(Z0**2 - 2.0*Z0*RL+RL**2+IM**2)
     K=K+2
           END IF
 GOTO 160
150      GAIN(I,J) = PROD
20     CONTINUE
10   CONTINUE
     RETURN
     END
```