LOUGHBOROUGH
UNIVERSITY OF TECHNOLOGY
LIBRARY

AUTHOR/FILING TITLE

DRUMMOND, J P

ACCESSION/COPY NO.

002465/01

VOL. NO. | CLASS MARK

ARCHIVES
COPY

FOR REFERENCE ONLY

COMPUTATIONAL AND ALGEBRAIC ASPECTS OF TWO-DIMENSIONAL, LINEAR, MULTIVARIABLE CONTROL SYSTEMS

BY

J. P. DRUMMOND B.Sc, M.Sc (Loughborough)

A Doctoral Thesis submitted in partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy at Loughborough University of Technology.

June 1983

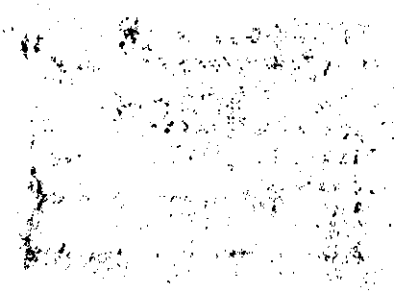
Supervisor: Prof. C. Storey.

© by Paul Drummond.

Loughborough University
of Technology Library
Date 0d 83
Class
Acc. No. 002465/01

ACKNOWLEDGEMENTS

I would obviously like to thank Prof C. Storey for his help and guidance throughout my research. Also many thanks are due to the staff and postgraduates in the Mathematics department at Loughborough for their help and friendship over the years. I would also like to thank Logica VTS for the use of their word processor which certainly made typing a lot easier for me, and the staff of the print room at Logica Ltd for preparing the copies of this thesis. Finally I would like to thank my parents without whose help none of this would have been possible.



ABSTRACT

There are at present a large number of theoretical and algorithmic results relating to one variable polynomial matrices arising from one-dimensional multivariable systems. In recent years many of the theoretical results have been extended to two variable polynomial matrices arising from two-dimensional multivariable systems, such as delay-differential or partial differential systems. However there has been no major attempt to extend the algorithmic results associated with single variable polynomial matrices to two variable or multivariable polynomial matrices.

This thesis investigates further some of the extensions of the algebra of one-dimensional multivariable systems to two-dimensional multivariable systems. The main area of interest is the equivalence of a two variable polynomial matrix with its Smith form over the ring $\mathbb{R}[s,z]$.

The thesis then provides algorithmic extensions to two variable polynomial system matrices. The algorithms developed are for the equivalence of a two variable polynomial matrix with its Smith form, the equivalence of a two variable rational matrix with its Smith-McMillan form, and the minimal realization of a two variable rational transfer function matrix as a state-space system matrix.

LIST OF SYMBOLS

$\mathbb{R}$	The field of real numbers.
$\mathbb{R}[s]$	The ring of polynomials in s, with coefficients over $\mathbb{R}$.
$\mathbb{R}(s)$	The field of rational functions in s, with coefficients over $\mathbb{R}$.
$\mathbb{R}[s,z]$	The ring of polynomials in s and z, with coefficients over $\mathbb{R}$.
$\mathbb{R}(z)[s]$	The ring of polynomials in s, with coefficients over $\mathbb{R}(z)$.
$\mathbb{R}[z][s]$	The ring of polynomials in s, with coefficients over $\mathbb{R}[z]$.
$\mathbb{R}^{mxn}[s,z]$	The m x n matrix with elements polynomials over $\mathbb{R}[s,z]$.

CONTENTS

APPENDIX E

PROGRAM LISTINGS: THE SMITH MCMILLAN FORM ALGORITHM

APPENDIX F

PROGRAM LISTINGS: THE REALIZATION ALGORITHM

CHAPTER 1

INTRODUCTION

1.1 Statement of the problem and basic definitions

There are at present a large number of theoretical and algorithmic results relating to system matrices associated with ordinary differential systems. In recent years many of these theoretical results have been extended for the theory of system matrices associated with delay-differential or partial differential systems. However there has been no major attempt to extend the algorithmic results associated with single variable polynomial system matrices to two variable or multivariable polynomial system matrices. This thesis will cover the algorithmic extensions to two variable polynomial system matrices whilst also attempting to design the algorithms so that they can be more easily extended to three or more variable polynomial system matrices, if required for future developments.

It will also be necessary to investigate some of the algebraic extensions, particularly those relating to the equivalence of two matrices over $R[s,z]$. This will be done so that conditions can be found for the equivalence of a

matrix with its Smith form over R[s,z], which would be useful in the design of the algorithms.

At this stage it is appropriate to give a few preliminary definitions.

Firstly full definitions of the Smith form over R[s], R[s,z] and R(z)[s] are given.

Definition 1.1: Smith form over R[s]

The Smith form of a p x q polynomial matrix K(s) over R[s] is defined to be the p x q matrix

$$S(s) = \begin{cases} [\, E(s) \,|\, 0 \,] & ; \; p<q \\ E(s) & ; \; p=q \\ \begin{bmatrix} E(s) \\ \hline 0 \end{bmatrix} & ; \; p>q \end{cases}$$

Here

$$E(s) = \text{diag} [\, e_i(s) \,]$$

where the ith invariant polynomial

$$e_i(s) = d_i(s) \,/\, d_{i-1}(s) \qquad i=1,2,\ldots,\min(p,q);$$

and the determinantal divisor $d_i(s)$ is the greatest common divisor of all the ith order minors of K(s), and d_0 is defined to be 1. All of the non-zero determinantal divisors

and invariant polynomials will be monic as polynomials in $\mathbb{R}[s]$, that is the leading coefficient is 1.

Also the invariant polynomials $e_i(s)$ have the divisibility property, that is:

$$e_1(s) \mid e_2(s) \mid \ldots \mid e_r(s)$$

where $r = \text{rank}(K(s))$.

Definition 1.2: Smith form over $\mathbb{R}[s,z]$ and $\mathbb{R}(z)[s]$

The Smith form of a p x q polynomial matrix $K(s,z)$ over $\mathbb{R}[s,z]$ is defined to be the p x q matrix:

$$S(s,z) = \begin{cases} [\, E(s,z) \,\vdots\, 0 \,] & ; \; p<q \\ E(s,z) & ; \; p=q \\ \begin{bmatrix} E(s,z) \\ \hline 0 \end{bmatrix} & ; \; p>q \end{cases}$$

Here

$$E(s,z) = \text{diag} [\, e_i(s,z) \,]$$

where the diagonal elements of $E(s,z)$ are the invariant polynomials over $\mathbb{R}[s,z]$ of $K(s,z)$, given by

$$e_i(s,z) = d_i(s,z) \,/\, d_{i-1}(s,z) \qquad i=1,2,\ldots, \min(p,q).$$

where d_0 is defined to be 1 and the determinantal divisor $d_i(s,z)$ is the greatest common divisor of all the ith order

minors of K(s,z). All the non-zero determinantal divisors and invariant polynomials will be taken to be monic over $\mathbb{R}[s,z]$.

Also the invariant polynomials $e_i(s,z)$ have the divisibility property, that is:

$$e_1(s,z) \mid e_2(s,z) \mid \ldots \mid e_r(s,z)$$

where r = rank(K(s,z)).

The Smith form, $S^S(s,z)$, over $\mathbb{R}(z)[s]$ of the p x q matrix K(s,z) has the same form as S(s,z) above, but in this case the invariant polynomials $e^S{}_i(s,z)$ are defined in terms of determinantal divisors $d^S{}_i(s,z)$ which are monic as polynomials in $\mathbb{R}(z)[s]$.

Similarly the Smith form $S^Z(s,z)$ over $\mathbb{R}(s)[z]$ can be defined.

Next the definitions of equivalence of two matrices over the rings $\mathbb{R}[s]$, $\mathbb{R}[s,z]$, and $\mathbb{R}(z)[s]$ are given.

Definition 1.3: Equivalence of two matrices over $\mathbb{R}[s]$

Two polynomial matrices $K_1(s)$ and $K_2(s)$ are equivalent over $\mathbb{R}[s]$ if and only if there exist two polynomial matrices M(s) and N(s) which are unimodular over $\mathbb{R}[s]$, and such that

$$K_1(s) = M(s)\, K_2(s)\, N(s).$$

M(s) and N(s) are unimodular over $\mathbb{R}[s]$ if the condition

$$\det(M(s)),\ \det(N(s)) \in \mathbb{R} \neq 0$$

is true, that is M(s) and N(s) are non-singular for all s.

Definition 1.4: Equivalence of two matrices over $\mathbb{R}[s,z]$ or $\mathbb{R}(z)[s]$

Two polynomial matrices $K_1(s,z)$ and $K_2(s,z)$ are equivalent over $\mathbb{R}[s,z]$ if and only if there exist two polynomial matrices M(s,z) and N(s,z) which are unimodular over $\mathbb{R}[s,z]$, such that

$$K_1(s,z) = M(s,z)\ K_2(s,z)\ N(s,z).$$

M(s,z) and N(s,z) are unimodular over $\mathbb{R}[s,z]$ if they are non-singular for all values of the pair (s,z).

For equivalence over $\mathbb{R}(z)[s]$, it is required that M(s,z) and N(s,z) are unimodular over $\mathbb{R}(z)[s]$, that is

$$\det(M(s,z)),\ \det(N(s,z)) \in \mathbb{R}(z) \neq 0.$$

These definitions are the basic ones to extend the theory of system matrices over $\mathbb{R}[s]$ to system matrices over $\mathbb{R}[s,z]$. Other definitions which will prove useful in the algebraic investigation and the algorithmic development will also be stated here.

Definition 1.5: Degree of a polynomial over $\mathbb{R}[s]$

A polynomial p(s) in $\mathbb{R}[s]$ can be expressed as the sum

$$p(s) = \sum_{i=0}^{r} a_i s^i \qquad a_i \in \mathbb{R}.$$

where the integer r is the degree of p(s). Also p(s) is monic over $\mathbb{R}[s]$ if $a_r=1$.

Definition 1.6: Degree of a polynomial over $\mathbb{R}[s,z]$

A polynomial p(s,z) in $\mathbb{R}[s,z]$ can be expressed as the sum

$$p(s,z) = \sum_{i=0}^{q} \sum_{j=0}^{r} a_{ij}\, s^i z^j \qquad a_{ij} \in \mathbb{R}.$$

If (m,n) is one of the pairs (i,j) such that m+n is the maximum value of i+j for which $a_{ij}\neq 0$, then m+n is the degree of p(s,z) over $\mathbb{R}[s,z]$. Also p(s,z) is monic over $\mathbb{R}[s,z]$ if $a_{mn}=1$. If there are more than one such pair , then for uniqueness the one which has largest degree in s is chosen as the leading term.

Definition 1.7: Degree of a polynomial over R(z)[s]

A polynomial p(s,z) in R(z)[s] can be expressed as the sum

$$p(s,z) = \sum_{i=0}^{r} a_i(z)\, s^i \qquad a_i(z) \in R(z).$$

where the integer r is the degree of p(s,z) over R(z)[s]. p(s,z) is monic as a polynomial in R(z)[s] if $a_r(z)=1$.

In one of the algorithms it is required to find the polynomial of least degree over R(z)[s] in a matrix where the elements are not necessarily monic, to use as a pivot. If two polynomials p(s,z) and q(s,z) have the same degree over R(z)[s], that is

$$p(s,z) = \sum_{i=0}^{r} a_i(z)\, s^i \qquad a_i(z) \in R(z),$$

$$q(s,z) = \sum_{i=0}^{r} b_i(z)\, s^i \qquad b_i(z) \in R(z).$$

Then an extra criterion for a better pivot is to choose p(s,z) if the degree of $a_r(z)$ is less than the degree of $b_r(z)$, since certainly q(s,z) could not divide p(s,z) without involving rational terms in z as it has leading term of higher degree in z than the leading term of p(s,z).

Definition 1.8: Existence of a division algorithm over $\mathbb{R}[s]$

Given any two polynomials $p_1(s)$ and $p_2(s)$ in $\mathbb{R}[s]$, with $\deg(p_1(s)) > \deg(p_2(s))$, there exist a unique pair of polynomials $q(s)$ and $r(s)$ in $\mathbb{R}[s]$, such that

$$p_1(s) = q(s)\, p_2(s) + r(s),$$

and $\deg(r(s)) < \deg(p_2(s))$. $q(s)$ is the quotient polynomial and $r(s)$ is the remainder polynomial.

Definition 1.9: Existence of a division algorithm over $\mathbb{R}(z)[s]$

Given any two polynomials $p_1(s,z)$ and $p_2(s,z)$ in $\mathbb{R}(z)[s]$, with $\deg(p_1(s,z)) > \deg(p_2(s,z))$, there exist a unique pair of polynomials $q(s,z)$ and $r(s,z)$ in $\mathbb{R}(z)[s]$, such that

$$p_1(s,z) = q(s,z)\, p_2(s,z) + r(s,z),$$

and $\deg(r(s,z)) < \deg(p_2(s,z))$. $q(s,z)$ is the quotient polynomial and $r(s,z)$ is the remainder polynomial.

However, a division algorithm over $\mathbb{R}[s,z]$ does not exist, as the following example will show.

Example 1.1:

Consider the polynomials $p_1(s,z) = s^2$, and $p_2(s,z) = z$ in $\mathbb{R}[s,z]$. Here $\deg(p_1(s,z)) > \deg(p_2(s,z))$. Let $q(s,z)$ and $r(s,z)$ be polynomials in $\mathbb{R}[s,z]$ and consider the equation

$$s^2 = q(s,z)\ z + r(s,z). \qquad (1.1)$$

As it would be required that $\deg(r(s,z)) < \deg (z) = 1$, then $r(s,z) \in \mathbb{R}$. Therefore the right hand side of (1.1) must have a term in z, whereas the left hand side does not. Therefore $q(s,z)$ and $r(s,z)$ cannot be in $\mathbb{R}[s,z]$ and so there does not exist a division algorithm over $\mathbb{R}[s,z]$.

If $q(s,z)$ and $r(s,z)$ were polynomials in $\mathbb{R}(z)[s]$, then $q(s,z) = s^2z^{-1}$, $r(s,z) = 0$ would solve equation (1.1).

One final result, which will be widely used throughout the thesis, will also be given here.

Theorem 1.1: Hilberts Nullstellensatz

(Van der Waerden 1964)

Polynomials $f_1,....,f_n \in R[z_1,....,z_q]$ have no common zeros if and only if the relationship

$$g_1 f_1 + ... + g_n f_n = 1$$

is valid in $R[z_1,....,z_q]$.

Where $g_1,....,g_n \in R[z_1,....,z_q]$, and are non-unique.

This will be mainly used in the following context. A necessary and sufficient condition that the polynomials $x(s,z)$ and $y(s,z)$ have no common zeros over $R[s,z]$ is that there exist polynomials $a(s,z)$ and $b(s,z)$ over $R[s,z]$ such that

$$a(s,z)\ x(s,z) + b(s,z)\ y(s,z) = 1.$$

1.2 Historical background of one variable polynomial system matrices

One variable polynomial system matrices arise from linear constant differential systems of the form

$$\dot{x}(t) = A\,x(t) + B\,u(t) \tag{2.1}$$

$$y(t) = C\,x(t) + D\,u(t) \tag{2.2}$$

where A,B,C,D are constant matrices.
On taking Laplace transforms, assuming zero initial conditions, these equations become:

$$(sI - A)\,\bar{x} = B\,\bar{u} \tag{2.3}$$

$$\bar{y} = C\,\bar{x} + D\,\bar{u} \tag{2.4}$$

which, when combined give the input-output mapping

$$\bar{y} = (\,C\,(sI - A)^{-1}\,B + D\,)\,\bar{u} \tag{2.5}$$

where

$$G(s) = C\,(sI - A)^{-1}\,B + D \tag{2.6}$$

is the transfer function matrix.

This system can be represented by the state-space

matrix

$$P(s) = \left[\begin{array}{c|c} sI - A & B \\ \hline -C & D \end{array}\right] \qquad (2.7)$$

If the output equation (2.2) contained differential terms of the control u(t), then we would have $D \equiv D(s)$ in (2.4) and (2.7).

If the output equation (2.2) was independant of the control u(t), then we would have $D \equiv 0$ in (2.4) and (2.7).

More generally linear constant differential systems may be represented by the polynomial system matrix

$$P(s) = \left[\begin{array}{c|c} T(s) & U(s) \\ \hline -V(s) & W(s) \end{array}\right] \qquad (2.8)$$

where, if the corresponding output equation is independant of the control, $W(s) \equiv 0$.

The polynomial system matrix (2.8) gives rise to the transfer function matrix

$$G(s) = V(s)\, T^{-1}(s)\, U(s) + W(s) \qquad (2.9)$$

It can be seen that the state-space form is a special case of the polynomial matrix form. Therefore the polynomial matrix form will be the one mainly considered here.

It can be shown (see, for example, Rosenbrock and Storey 1970) that any polynomial matrix in $\mathbb{R}[s]$ is always equivalent to its Smith form over $\mathbb{R}[s]$. In fact the proof of this result is constructive and is analogous to the technique of Gaussian elimination for transforming a matrix to diagonal form. As the method is constructive and could form the basis of an algorithm it will be given in full here.

Consider the m x n polynomial matrix P(s),

$$P(s) = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{bmatrix} \qquad (2.10)$$

where the polynomials $p_{ij} = p_{ij}(s)$ are elements in $\mathbb{R}[s]$.

Firstly a non-zero polynomial of least degree is brought to position p_{11} by appropriate row and column operations.

Then for each of the elements $p_{12},\ldots,p_{1n}$ of the pivotal row, in turn, the division algorithm is applied to form

$$p_{1j} = p_{11}\, g_{1j} + r_{1j} \qquad j = 2,\ldots,n$$

with r_{1j} zero, or having degree less than the degree of p_{11}. Then g_{1j} times column 1 is subtracted from column j.

The same process is applied to the elements $p_{21},\ldots,p_{m1}$ of the pivotal column, in turn, so that

$$p_{i1} = p_{11}\, g_{i1} + r_{i1} \qquad i = 2,\ldots,m$$

with r_{i1} zero, or having degree less than the degree of p_{11}. Then g_{i1} times row 1 is subtracted from row i.

This procedure will leave elements r_{1j} in the pivotal row and r_{i1} in the pivotal column. Now either all the r_{1j}, r_{i1} are zero, or an element of lower degree can be brought to position p_{11}. This procedure is then repeated for the new p_{11} and continued until all the r_{1j}, r_{i1} are zero.

If any of the columns $2,\ldots,n$ now contains an element which is not divisible by the leading element p_{11}, then this column is added to the first column. Then the degree of the leading element p_{11} can be reduced until the process will finally terminate with the matrix Q(s) which is equivalent over R[s] to P(s),

$$Q(s) = \begin{bmatrix} p_1 & 0 & \cdots & 0 \\ 0 & q_{22} & \cdots & q_{2n} \\ 0 & q_{32} & \cdots & q_{3n} \\ \vdots & \vdots & & \vdots \\ 0 & q_{m2} & \cdots & q_{mn} \end{bmatrix} \qquad (2.11)$$

where p_1 can be made to be monic over R[s] and is a factor of all the remaining elements of Q.

If the above procedure is now repeated for the sub-matrix Q'(s) of Q(s),

$$Q'(s) = \begin{bmatrix} q_{22} & q_{23} & \cdots & q_{2n} \\ q_{32} & q_{33} & \cdots & q_{3n} \\ \vdots & \vdots & & \vdots \\ q_{m2} & q_{m3} & \cdots & q_{mn} \end{bmatrix} \qquad (2.12)$$

and so on, then the Smith form of P(s) will be finally obtained.

The algorithm may now be given formally in a structured form.

Algorithm 2.1: Smith form of a one variable polynomial matrix

(1) Let K be the initial polynomial matrix of dimension m x n.

(2) r=1.

(3) Consider the sub-matrix of K which has its top left hand corner at position (r,r). If all the elements are zero goto 16.

(4) Find the position of the non-zero element with least degree. Move it to position (r,r) by row and column interchange.

(5) i=r+1, j=r+1, pivot=element K(r,r).

(6) Consider column j. Divide element K(r,j) by pivot, giving the quotient q. Subtract q times column r from column j.

(7) j=j+1. If $j \leq n$ goto 6.

(8) Consider row i. Divide element K(i,r) by pivot, giving the quotient q. Subtract q times row r from row i.

(9) i=i+1. If $i \leq m$ goto 8.

(10) If not all elements in the pivotal row and column are zero goto 4.

(11) k=r.

(12) k=k+1. If 1>n goto 15.

(13) If pivot divides all elements in column k goto 12.

(14) Add column k to column r. Goto 4.

(15) r=r+1. If r<min(m,n) goto 3.

(16) Stop.

A number of different algorithms have been developed along the same lines as algorithm 2.1, with modifications. Pace and Barnett (1974a) have produced the most efficient of these algorithms, using a new version of the Euclidean algorithm by Blankinship (1963) which calculates the greatest common divisor of two polynomials, and a set of multipliers for each of the rows and columns.

However, the Blankinship algorithm uses the division algorithm to find the greatest common divisor. The approach taken in this thesis is to overcome the lack of a division algorithm over $\mathbb{R}[s,z]$ by extending algorithm 2.1 rather than the method of Pace and Barnett.

Another canonical form to be considered later is the Smith-McMillan form of a rational polynomial matrix.

Definition 2.1: Smith-McMillan form of a matrix over $\mathbb{R}(s)$

Consider a p x q rational matrix K(s), where the elements of K(s) are

$$n_{ij}(s) \,/\, d_{ij}(s) \qquad \begin{matrix} i=1,\ldots,p \\ j=1,\ldots,q \end{matrix}$$

and $n_{ij}(s), d_{ij}(s) \in \mathbb{R}[s]$.

The Smith-McMillan form of K(s) is defined to be the matrix

$$M(S) = \begin{cases} [\ E(s) \mid 0\ ] & ;\ p<q \\ E(s) & ;\ p=q \\ \begin{bmatrix} E(s) \\ \hline 0 \end{bmatrix} & ;\ p>q \end{cases}$$

where $E(s) = \text{diag}\ [\ e_i(s) / g_i(s)\ ]$,

and $e_i(s), g_i(s)$ are relatively prime polynomials in $\mathbb{R}[s]$ with the division property along the diagonal, that is

$$e_1(s) \mid e_2(s) \mid \ldots\ \mid e_r(s),$$

and $g_r(s) \mid g_{r-1}(s) \mid \ldots\ \mid g_1(s)$,

where $r = \text{rank}(\ K(s)\ )$.

Algorithms to find the Smith-McMillan form of a rational matrix K(s) are based on the Smith form of a related polynomial matrix (see, for example, Rosenbrock 1970). If the least common denominator d(s) of all the elements of K(s) is found, then K(s) can be expressed

$$K(s) = N(s) / d(s). \qquad (2.13)$$

If N(s) is now transformed into its Smith form, S(s), then K(s) is equivalent over $\mathbb{R}[s]$ to the matrix M(s),

$$M(s) = S(s) / d(s). \qquad (2.14)$$

If any common factors between the numerator and denominator of the elements of the leading diagonal of M(s) are cancelled, then M(s) will be of the form of definition 2.1, the Smith-McMillan form of K(s).

Finally, in this section, it is worth considering the realization problem in $\mathbb{R}[s]$, that is given a transfer function matrix G(s), construct a state-space system matrix corresponding to G(s), of the form

$$\left[\begin{array}{c|c} sI - A & B \\ \hline -C & D(s) \end{array}\right] \qquad (2.15)$$

such that

$$G(s) = C\,(sI - A)^{-1}\,B + D(s). \qquad (2.16)$$

If G(s) is proper, that is G(s) tends to the zero matrix as s tends to infinity, then $D(s) \equiv 0$.
If the matrix A is of least dimensions, then the realization is said to be minimal (see, for example, Barnett 1971). Also it can be shown that a minimal realization over $\mathbb{R}[s]$ is always controllable and observable (Barnett 1971).

Once again a number of computer algorithms have been written, on the whole based on the method of Rosenbrock (1970). One such algorithm is that of Munro and McLeod

(1971). This method involves the construction of an observable state-space realization, and then removing any input-decoupling zeros to give a minimal state-space realization. This method is shown to be more efficient than its predecessors, with a reduction of the order of 100:1 in computation time.

However a more recent algorithm, Pace and Barnett (1974b) is shown to be the most efficient of all. The strategy of this method is slightly different to that of Munro and McLeod. A minimal polynomial realization is firstly constructed and then this is transformed into a minimal state-space realization. It is this difference which makes the method more efficient.

1.3 Historical background of two variable polynomial system matrices

Two variable polynomial system matrices can arise from a number of different systems. One of the main ones is the linear delay-differential system

$$\dot{x}(t) = A(d)\, x(t) + B(d)\, u(t) \tag{3.1}$$

$$y(t) = C(d)\, x(t) + D(d)\, u(t) \tag{3.2}$$

where d is the delay operator

$$d\, x(t) = x(t-h)$$

for some fixed delay h.

Another system which also gives rise to a two variable polynomial system matrix is the partial differential system

$$x_t(t) = A(d)\, x(t) + B(d)\, u(t) \tag{3.3}$$

$$y(t) = C(d)\, x(t) + D(d)\, u(t) \tag{3.4}$$

where d is now the partial differential operator with respect to the extra space variable T

$$d\, x(t) = x_T(t).$$

If the Laplace transform is taken for either of these systems the resulting equations are

$$(sI - A(z)) \bar{x} = B(z) \bar{u} \tag{3.5}$$

$$\bar{y} = C(z) \bar{x} + D(z) \bar{u} \tag{3.6}$$

which, when combined, give the input-output mapping

$$\bar{y} = (C(z) (sI - A(z))^{-1} B(z) + D(z)) \bar{u} \tag{3.7}$$

where

$$G(s,z) = C(z) (sI - A(z))^{-1} B(z) + D(z) \tag{3.8}$$

is the transfer function matrix.

This system can be represented by the state-space matrix over $\mathbb{R}[s,z]$,

$$P(s,z) = \left[\begin{array}{c|c} sI - A(z) & B(z) \\ \hline -C(z) & D(z) \end{array}\right] \tag{3.9}$$

If the output equation, (3.2) or (3.4), contained differential terms of the control u(t), then $D \equiv D(s,z)$.

If the output equation, (3.2) or (3.4), is independant of the control u(t), then $D \equiv 0$.

More generally, these systems can be represented by the polynomial system matrix over $\mathbb{R}[s,z]$,

$$P(s,z) = \left[\begin{array}{c|c} T(s,z) & U(s,z) \\ \hline -V(s,z) & W(s,z) \end{array}\right] \qquad (3.10)$$

where, if the corresponding output equation is independant of the control u(t), then $W(s,z) \equiv 0$.

The polynomial system matrix (3.10) gives rise to the transfer function matrix

$$G(s,z) = V(s,z)\ T^{-1}(s,z)\ U(s,z) + W(s,z) \qquad (3.11)$$

It can be seen that if there is more than one delay, say, the resulting system matrix will be over the ring $\mathbb{R}[s,z_1,z_2,\ldots,z_r]$, where $z_1,\ldots,z_r$ are the Laplace variables of the r independant delays. Therefore when considering the extension of results for $\mathbb{R}[s]$ to results for $\mathbb{R}[s,z]$, it would be useful to consider also the further extension to $\mathbb{R}[s,z_1,\ldots,z_r]$.

However, as will now be shown, the extension of results for $\mathbb{R}[s]$ to results for $\mathbb{R}[s,z]$ is not straightforward. Indeed the extension is actually to the ring $\mathbb{R}(z)[s]$, as both $\mathbb{R}[s]$ and $\mathbb{R}(z)[s]$ are principal ideal domains, whereas the ring $\mathbb{R}[s,z]$ is not.

Frost (1979) found a major difference between matrices over $\mathbb{R}[s]$ and matrices over $\mathbb{R}[s,z]$. For a matrix over $\mathbb{R}[s]$

it is certainly the case that if the determinantal divisor $d_i(s)$ is removed from all the ith order minors, then the remaining polynomials cannot be simultaneously zero for any value of s. This result does not extend for matrices over $\mathbb{R}[s,z]$, and so prompts the definition of zeros of a matrix over $\mathbb{R}[s,z]$.

Definition 3.1: Zeros of a matrix over $\mathbb{R}[s,z]$ (Frost 1979)

Given a matrix over $\mathbb{R}[s,z]$ it is possible that on removal of the determinantal divisor $d_i(s,z)$ from all the ith order minors of the matrix, the remaining polynomials can all be simultaneously zero for one or more values of the pair (s,z). Such a value of (s,z) will be defined as an ith order zero of the matrix over $\mathbb{R}[s,z]$.

Example 3.1: (Frost 1979)

Consider the matrix

$$K(s,z) = \begin{bmatrix} s+z & 0 & z \\ 0 & s+z & 0 \\ 0 & 0 & s \end{bmatrix}$$

which has determinantal divisors

$$d_1(s,z) = 1,\ d_2(s,z) = s+z,\ d_3(s,z) = s(s+z)^2.$$

If the second order determinantal divisor $d_2(s,z)$ is removed

from all the second order minors the following non-zero polynomials remain:

$$(s+z),\ z,\ s,\ s.$$

These are all simultaneously zero for the pair (0,0). Therefore (0,0) is a second order zero of K(s,z).

This is a very important property and shows the need to extend the property of zeros in $\mathbb{R}[s]$ to factors and zeros in $\mathbb{R}[s,z]$. Confusion must be avoided, as zeros in $\mathbb{R}[s]$ are actually factors in $\mathbb{R}[s]$.

As shown in section 1.1 there exists the concept of equivalence of matrices over $\mathbb{R}(z)[s]$ or $\mathbb{R}(s)[z]$. For this reason Morf et al (1977) have suggested a method to transform a matrix K(s,z) to its Smith form over $\mathbb{R}[z][s]$. This could be done by firstly transforming K(s,z) to its Smith form over $\mathbb{R}(z)[s]$, that is

$$M(s,z)\ K(s,z)\ N(s,z) = S^S(s,z) \qquad (3.12)$$

where $\det(M(s,z))$, $\det(N(s,z)) \in \mathbb{R}(z)$, and the matrices M(s,z), N(s,z), $S^S(s,z)$ are over the ring $\mathbb{R}(z)[s]$. If M(s,z) and N(s,z) are now renormalized by multiplication by diagonal matrices over $\mathbb{R}[z]$, such that M(s,z), N(s,z) are now matrices over $\mathbb{R}[s,z]$ and $\det(M(s,z))$, $\det(N(s,z))$

are in $\mathbb{R}[z]$, then the resulting Smith form $S^{s}{}'(s,z)$ would be the Smith form of $K(s,z)$ over $\mathbb{R}[z][s]$. This results in the following theorem.

Theorem 3.1: (Morf et al 1977)

Given any polynomial matrix $K(s,z)$, there exist two polynomial matrices $M(s,z)$ and $N(s,z)$ with

$$\det(M(s,z)),\ \det(N(s,z)) \in \mathbb{R}[z]$$

such that the Smith form $S^{s}(s,z)$, of $K(s,z)$ over $\mathbb{R}[z][s]$ can be obtained by

$$M(s,z)\ K(s,z)\ N(s,z) = S^{s}(s,z).$$

The same result would apply over $\mathbb{R}[s][z]$. However the Smith forms over $\mathbb{R}[z][s]$ and $\mathbb{R}[s][z]$ may be quite different and neither may be the Smith form over $\mathbb{R}[s,z]$.

Example 3.2

Consider the matrix

$$K(s,z) = \begin{bmatrix} s & -(1+z) \\ -z^2 & s \end{bmatrix}$$

This has Smith form over $\mathbb{R}[z][s]$

$$S^s(s,z) = \begin{bmatrix} z+1 & 0 \\ 0 & (z+1)(s^2-z^2(z+1)) \end{bmatrix} .$$

and Smith form over $\mathbb{R}[s][z]$

$$S^z(s,z) = \begin{bmatrix} s & 0 \\ 0 & s(s^2-z^2(z+1)) \end{bmatrix}$$

and the Smith form over $\mathbb{R}[s,z]$ is

$$S(s,z) = \begin{bmatrix} 1 & 0 \\ 0 & s^2-z^2(z+1) \end{bmatrix}$$

which shows that all three Smith forms are different.

Although the method of Morf et al would give an equivalence over $\mathbb{R}[z][s]$, if possible it would be more desirable to obtain equivalence over $\mathbb{R}[s,z]$. As it is known that equivalence over $\mathbb{R}[s,z]$ is not always possible, it would be useful to find conditions on the matrix which are necessary and sufficient for equivalence over $\mathbb{R}[s,z]$ to its Smith form.

Frost (1979) found that a transformation of equivalence over $\mathbb{R}[s,z]$ will preserve the zeros of a matrix, whereas a

transformation of equivalence over $\mathbb{R}(z)[s]$ need not do so. As the Smith form of any matrix has no zeros, then the invariance of zeros for equivalence over $\mathbb{R}[s,z]$ gives the following result.

Theorem 3.2: (Frost 1979)

A necessary condition for the equivalence of a polynomial matrix $K(s,z)$ with its Smith form $S(s,z)$ over $\mathbb{R}[s,z]$ is that $K(s,z)$ should have no zeros.

Frost and Storey (1978) initially thought that the property of zeros of a matrix over $\mathbb{R}[s,z]$ is also a sufficient condition for equivalence to its Smith form over $\mathbb{R}[s,z]$. However Lee and Zak (1981) are able to obtain a necessary and sufficient condition for equivalence to its Smith form over $\mathbb{R}[s,z]$ of a certain class of matrix. From this they are able to produce a counter example to Frost and Storeys result. To discuss their result it is first necessary to define a cyclic vector, and a cyclic (or non-derogatory) matrix.

Definition 3.2: A cyclic (or non-derogatory) matrix over $\mathbb{R}[z]$

An n x n matrix $A(z)$ is said to be cyclic if there exists a vector $b(z)$ such that the matrix

$$[b, Ab, \ldots, A^{n-1}b]$$

has full rank for all z. Such a vector b(z) for which this holds is called a cyclic vector (Lee and Zak 1981).

Theorem 3.3: (Lee and Zak 1981)

Necessary and sufficient conditions for the existence of a cyclic vector $b(z) \in R^n[z]$, for a given matrix $A(z) \in R^{n \times n}[z]$ are the following:

(i) The Smith form of the matrix [sI - A(z)] is

$$S_A(s,z) = \left[\begin{array}{c|c} I & 0 \\ \hline 0 & \det(\, sI - A(z)\,) \end{array}\right]$$

that is, the degree in s of det(sI - A(Z)) is equal to the degree in s of the minimal polynomial of A(z).

(ii) The matrices $S_A(s,z)$ and [sI - A(z)] are equivalent over R[s,z].

As a consequence to this theorem, Lee and Zak provided the following counter example to the result of Frost and Storey (1978).

Example 3.3: (Lee and Zak 1981)

Consider the matrix

$$P(s,z) = \begin{bmatrix} s & -(1+z) \\ -z^2 & s \end{bmatrix}$$

which has no zeros and is of the form $[sI - A(z)]$ where

$$A(z) = \begin{bmatrix} 0 & 1+z \\ -z^2 & 0 \end{bmatrix}$$

Now

$$\det[b,Ab] = \det \begin{bmatrix} b_1 & (1+z)b_2 \\ b_2 & z^2 b_1 \end{bmatrix}$$

$$= z^2 b_1^2 - (1+z)b_2^2 \qquad \text{where } b = \begin{bmatrix} b_1(z) \\ b_2(z) \end{bmatrix}$$

But $z^2 b_1^2 - (1+z)b_2^2 \notin \mathbb{R} \neq 0$ for any polynomials $b_1, b_2 \in \mathbb{R}[z]$ as it is not sign definite for values of z.

Therefore by theorem 3.3, $P(s,z)$ is not equivalent to its Smith form $S(s,z)$,

$$S(s,z) = \begin{bmatrix} 1 & 0 \\ 0 & s^2 - z^2(1+z) \end{bmatrix}$$

over $\mathbb{R}[s,z]$ even though $P(s,z)$ has no zeros.

Frost (1979) tried to construct an equivalence transformation to transform a matrix $K(s,z)$, which has no zeros, to its Smith form over $\mathbb{R}[s,z]$.

Firstly $K(s,z)$ can be transformed by equivalence over $\mathbb{R}[s,z]$ to the form

$$e_1(s,z)\ K'(s,z) \tag{3.13}$$

where $e_1(s,z)$ is the first invariant polynomial of $K(s,z)$, and $K'(s,z)$ is such that the (1,1)th and (1,2)th elements have no common zeros. It can be seen, by theorem 1.1, that two polynomials $x(s,z), y(s,z) \in \mathbb{R}[s,z]$ have no common zeros if and only if there exist polynomials $a(s,z), b(s,z) \in \mathbb{R}[s,z]$ such that

$$a(s,z)\ x(s,z) + b(s,z)\ y(s,z) = 1 \tag{3.14}$$

Now using this result a transformation of equivalence over $\mathbb{R}[s,z]$ can be constructed which brings $K'(s,z)$ to the form

$$\left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & K_1(s,z) \end{array}\right] \tag{3.15}$$

Continuing this process on $K_1(s,z)$ will construct a transformation of equivalence over $\mathbb{R}[s,z]$ which will transform $K(s,z)$ into its Smith form.

However, as example 3.3 has shown, the property of zeros is not a sufficient condition for equivalence over $\mathbb{R}[s,z]$. Therefore the point at which the method fails is the assertion that it is always possible to transform $K(s,z)$ into $e_1(s,z)K'(s,z)$ where $K'(s,z)$ has the (1,1)th and (1,2)th elements having no common zeros.

The final extension to be covered in this section, is an algorithm for the greatest common divisor extraction from two multivariable polynomials due to Bose (1976). This is extremely useful because it finds the greatest common divisor of polynomials in any number of variables. This method makes use of bigradients or subresultants (see, for example, Barnett 1971).

Consider two polynomials $f(p_1,....,p_k)$ and $g(p_1,....,p_k)$ written as

$$f(p_1,....,p_k) = a_0 p_1^n + a_1 p_1^{n-1} + ... + a_n \qquad (3.16)$$

$$g(p_1,....,p_k) = b_0 p_1^m + b_1 p_1^{m-1} + ... + b_m \qquad (3.17)$$

where the a_i, b_j are polynomials in the variables

$(p_2,....,p_k)$. The method works for primitive polynomials. Therefore it is necessary to remove the content of each polynomial (that is the greatest common divisor of the a_i in the case of f), by recursively using the method on the (k-1) variables $(p_2,....,p_k)$ of firstly the polynomials $a_0,....,a_n$ for the content of f, and then $b_0,....,b_m$ for the content of g.

This gives the result

$$\gcd(f,g)=[\gcd(\ \mathrm{cont}(f),\mathrm{cont}(g)\ )][\gcd(\ \mathrm{pp}(f),\mathrm{pp}(g)\ )] \quad (3.18)$$

where cont(f) is the content of f, and pp(f) is the primitive part of f.

1.4 Synopsis of the thesis

The previous sections of this chapter have covered some of the relevant work relating to polynomial system matrices. However it is noticeable that there has been far less extension of the algorithms developed for one variable system matrices to two variable system matrices, than the underlying algebra.

Chapter 2 attempts to extend some of the algebra to the ring $\mathbb{R}[s,z]$. This is done by considering the 2x2 polynomial matrix over $\mathbb{R}[s,z]$,

$$\begin{bmatrix} x(s,z) & y(s,z) \\ u(s,z) & w(s,z) \end{bmatrix} \qquad (4.1)$$

The equivalence of this matrix to its known Smith form is analysed in an attempt to find conditions under which the equivalence exists, whilst at the same time investigating the construction of the actual equivalence transformation. However it is not possible to seperate the conditions for the equivalence and the construction. Therefore there is further investigation into the result of Lee and Zak given in theorem 3.3.

The chapter then deals with the development of a computer algorithm for the transformation of a matrix to its

Smith form over R[s,z], based on algorithm 2.1. There were quite a few problems in constructing this algorithm, and these will be fully discussed.

Because of these problems it was also worth trying to compare this algorithm with one which finds the Smith form by explicitly calculating the determinantal divisors of the matrix. It is then possible to compare computing times of the two methods, to show whether it is more efficient to calculate the determinantal divisors or to use the equivalence over R[s,z].

To conclude chapter 2, an algorithm is developed which will calculate the Smith-McMillan form of a rational polynomial matrix, using the previously developed Smith form algorithm.

A number of examples will be given to show how the algorithms perform on matrices chosen to illustrate various difficulties.

Because it is not possible to find conditions for equivalence over R[s,z] of a general matrix in chapter 2, chapter 3 considers the concept of extended equivalence of Pugh and Shelton (1978). This is again applied to the general 2x2 matrix, to investigate if the new approach will prove more useful in finding the required conditions. However this turns out to be just an alternative route to

the same results as chapter 2, and the new approach gives no other insights into the problem.

As it has not been possible to find necessary and sufficient conditions for the equivalence of a general polynomial matrix to its Smith form over $\mathbb{R}[s,z]$, chapter 4 considers the problem for a particular form of matrix, the Roesser matrix. The Roesser matrix has the form

$$\left[\begin{array}{c|c|c} sI_n - a_1 & -a_2 & b_1 \\ \hline -a_3 & zI_p - a_4 & b_2 \\ \hline -c_1 & -c_2 & d \end{array}\right] \qquad (4.2)$$

and arises from a number of different systems. The matrix (4.2) could be a special form of state-space matrix over $\mathbb{R}[s,z]$ arising from delay-differential or partial differential systems. Such matrices arise in the study of two dimensional image processing systems (see, for example, Kung et al 1977) and indeed arise naturally from the approach suggested by Givone and Roesser (1972,1973) or Fornasini and Marchesini (1975) for two-dimensional filters.

Mathematical induction is used to produce a sufficient condition for equivalence of a certain class of Roesser matrix of the form (4.2) to its Smith form over $\mathbb{R}[s,z]$. Although the method is tedious it is the only one known at

the present time to find the sufficient condition.

To conclude chapter 4 a number of Roesser matrices are used as examples for testing the Smith form program.

Chapter 5 moves on to the topic of the realization of a two variable rational transfer function matrix to a state-space system matrix over $\mathbb{R}[s,z]$. The algorithm is based on that developed by Pace and Barnett (1974b) for the single variable realization. However, as Frost (1979) has shown, a realization over $\mathbb{R}[s,z]$ may not always be both controllable and observable. This is again due to the property of zeros, in this case the fact that it may not be possible to remove both input-decoupling and output-decoupling zeros. Both the theoretical background and algorithmic development of the realization will be covered with examples to test the program.

Chapter 6 concludes the thesis, examining how successful were the attempts to extend the algebra and the algorithms to two variable system matrices. There are also suggestions of areas for further research which may help to resolve some of the outstanding problems.

All the work in the last five chapters is original unless otherwise stated.

CHAPTER 2

FURTHER EXTENSIONS OF THE THEORY, AND ALGORITHMS RELATING TO TWO VARIABLE POLYNOMIAL MATRICES

2.1 Introduction

From section 1.3 of the introduction it is clear that the theoretical extension of the results for 1-D systems to 2-D systems has not been completed. This is certainly true for results concerning equivalence of a general two variable polynomial matrix with its Smith form. Theorem 3.2 of chapter 1 gives a necessary condition, but a corresponding sufficiency condition has not been found.

This chapter attemts to find a sufficiency condition by examining a 2x2 general two variable matrix. If such a condition was found the result could be extended to a m x n two variable matrix by mathematical induction. Following this the result of Lee and Zak (1981), theorem 3.3 of chapter 1, is extended for a more general matrix.

Finally in this chapter a number of algorithms are developed for computing the Smith form and Smith-McMillan form of two variable polynomial or rational matrices.

2.2 Algebraic investigations into equivalence of a 2x2 two variable polynomial matrix with its Smith form over R[s,z]

To attempt to find the required sufficiency condition for equivalence of a two variable polynomial matrix with its Smith form over R[s,z] it would be useful firstly to consider a 2x2 matrix without zeros.

Consider the matrix A(s,z) , without zeros,

$$A(s,z) = \begin{bmatrix} x(s,z) & y(s,z) \\ u(s,z) & w(s,z) \end{bmatrix} \qquad (2.1)$$

It can be assumed that x(s,z),y(s,z),u(s,z), and w(s,z) have no common factor and (as A(s,z) has no zeros) also no common zeros. Therefore the Smith form of A(s,z) is the matrix S(s,z),

$$S(s,z) = \begin{bmatrix} 1 & 0 \\ 0 & x(s,z)\,w(s,z) - y(s,z)\,u(s,z) \end{bmatrix} \qquad (2.2)$$

To find the sufficiency condition it is assumed that A(s,z) is equivalent to S(s,z) over R[s,z]. That is there exist unimodular matrices M(s,z), N(s,z) over R[s,z] such that

$$M(s,z)\ A(s,z)\ N(s,z) = S(s,z) \tag{2.3}$$

or

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x & y \\ u & w \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & xw-uy \end{bmatrix} \tag{2.4}$$

where $x = x(s,z)$ etc, for ease of notation.
By solving the matrix equation, 2.4, it should be possible to find sufficiency conditions for $A(s,z)$ for equivalence over $\mathbb{R}[s,z]$ to exist. Also it should be possible to give the actual construction for the equivalence transformation.

In section 3 of the introduction it was shown that Frost was able to construct a transformation of equivalence if adjacent elements of the matrix had no common zeros.

For example if $x(s,z)$ and $u(s,z)$ have no common zero, then there exist, by theorem 1.1 of chapter 1, polynomials $a(s,z)$ and $b(s,z)$ over $\mathbb{R}[s,z]$ such that,

$$a(s,z)\ x(s,z) + b(s,z)\ u(s,z) = 1, \tag{2.5}$$

which gives the transformation of equivalence over $\mathbb{R}[s,z]$,

$$\begin{bmatrix} a & b \\ -u & x \end{bmatrix} \begin{bmatrix} x & y \\ u & w \end{bmatrix} \begin{bmatrix} 1 & -(ay+bw) \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & xw-uy \end{bmatrix} \quad (2.6)$$

Therefore for this investigation it is also assumed that any adjacent pair of polynomials of A(s,z) have common zeros. That is the pairs

$$(x,y),\ (x,u),\ (u,w),\ (y,w) \quad (2.7)$$

have common zeros.

Expanding (2.4) gives the four equations

$$e(ax+bu) + g(ay+bw) = 1 \quad (2.8)$$

$$f(ax+bu) + h(ay+bw) = 0 \quad (2.9)$$

$$e(cx+du) + g(cy+dw) = 0 \quad (2.10)$$

$$f(cx+du) + h(cy+dw) = xw-uy \quad (2.11)$$

Also there are the unimodularity conditions on the equivalence matrices M(s,z) and N(s,z), which give

$$ad - bc = k_1 \in R \neq 0 \quad (2.12)$$

$$eh - fg = k_2 \in R \neq 0 \quad (2.13)$$

It can be seen that equation (2.8) holds the key to the transformation, because once there is a 1 in the top left

hand corner of the matrix, the remaining transformation to the Smith form is straightforward.

If (2.8) holds, that is

$$e(ax+bu) + g(ay+bw) = 1$$

then the equivalence transformation is,

$$\begin{bmatrix} a & b \\ -(eu+gw) & (ex+gy) \end{bmatrix} \begin{bmatrix} x & y \\ u & w \end{bmatrix} \begin{bmatrix} e & -(ay+bw) \\ g & (ax+bu) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & xw-uy \end{bmatrix} \tag{2.14}$$

where the determinant of both the equivalence matrices is 1.

A few methods are now proposed which attempt to find the conditions under which (2.8) holds.

2.2.1 Method 1

The problem is to solve the equation

$$e(ax+bu) + g(ay+bw) = 1 \tag{2.1.1}$$

for polynomials $a,b,e,g \in R[s,z]$, for the given polynomials $x,u,y,w \in R[s,z]$.

By theorem 1.1 of chapter 1, (2.1.1) can be solved if and only if the polynomials

$$(ax+bu)\ ,\ (ay+bw) \qquad (2.1.2)$$

have no common zeros.

To investigate this, consider the set of zeros $\{ (s_1,z_1) \}$ of the polynomial (ax+bu), and examine the polynomial (ay+bw) at these points to find conditions such that (ay+bw) is not zero.

Let the set of pairs $\{ (s_1,z_1) \}$ be the zeros of (ax+bu), that is

$$a_1\ x_1 + b_1\ u_1 = 0 \qquad (2.1.3)$$

where $a_1 = a(s,z) \Big|_{\{ (s_1,z_1) \}}$ etc.

The set $\{ (s_1,z_1) \}$ is non-empty, as from (2.7) x and u have common zeros.

Evaluating (ay+bw) at this set, it is assumed that there is a subset $\{ (s_0,z_0) \}$ of $\{ (s_1,z_1) \}$ for which

$$a_0\ y_0 + b_0\ w_0 = 0 \qquad (2.1.4)$$

If necessary conditions can now be found for this subset to be non-empty, then these will be conditions for the polynomials (ax+bu) and (ay+bw) to have common zeros. From

this it should be possible to find sufficient conditions for which the polynomials (ax+bu) and (ay+bw) have no common zeros and so solve equation (2.1.1).

Firstly consider the case when $x_0 = 0$. From equation (2.1.3) we have

$$b_0 u_0 = 0 \qquad (2.1.5)$$

which implies that $b_0 = 0$ and/or $u_0 = 0$.
If $b_0 = 0$, then (2.1.4) gives

$$a_0 y_0 = 0 \qquad (2.1.6)$$

which implies that $y_0 = 0$, as $a_0 \neq 0$ because a and b have no common zeros. This gives the result.

Lemma 2.1.1

A sufficient condition for the polynomials (ax+bu) and (ay+bw) to have common zeros is that the polynomials b,x,y have common zeros.

Now consider $u_0 = 0$ in equation (2.1.5). To satisfy (2.1.4) we require

$$a_0 y_0 + b_0 w_0 = 0 \qquad (2.1.7)$$

at the common zeros of x and u.
To further investigate this, let $y_0 = 0$. This implies $b_0 = 0$ as x,y,u,w have no common zero. If $y_0 \neq 0$, let $w_0 = 0$ from which we have the condition

$$a_0 = 0 \text{ at the common zeros of } x,u,w. \qquad (2.1.8)$$

Finally if $y_0 \neq 0$ and $w_0 \neq 0$ this gives the condition

$$a_0 y_0 + b_0 w_0 = 0 \qquad (2.1.9)$$

at the common zeros of x and u which are not zeros of y or w. These can be combined to give the result.

Lemma 2.1.2

Sufficient conditions for the polynomials (ax+bu) and (ay+bw) to have common zeros are the following:

(i) The polynomials b,x,u,y have common zeros.

(ii) The polynomials a,x,u,w have common zeros.

(iii) The polynomial (ay+bw) is zero at the set of points which are common zeros of x and u, but not zeros of y or w.

To complete this approach it is necessary to consider the case when $x_0 \neq 0$.

In this set if $u_0 = 0$ then from (2.1.3) $a_0 = 0$. Also from (2.1.4) we have

$$b_0 w_0 = 0 \qquad (2.1.10)$$

which implies $w_0 = 0$, as a and b have no common zero. This gives the result

Lemma 2.1.3

A sufficient condition for the polynomials (ax+bu) and (ay+bw) to have common zeros is that $a_0 = 0$ at the common zeros of u and w which are not zeros of x.

Again if we now consider $x_0 \neq 0$ and $u_0 \neq 0$ then (2.1.3) gives

$$a_0 x_0 = - b_0 u_0 \quad \text{with } a_0 \neq 0,\ b_0 \neq 0. \qquad (2.1.11)$$

(2.1.4) gives

$$a_0 y_0 + b_0 w_0 = 0 \qquad (2.1.12)$$

now multiplying by x_0 ($\neq 0$) gives

$$a_0 x_0 y_0 + b_0 x_0 w_0 = 0$$

which gives, from (2.1.11)

$$- b_0 u_0 y_0 + b_0 x_0 w_0 = 0$$

or

$$b_0 (x_0 w_0 - u_0 y_0) = 0 \qquad (2.1.13)$$

and hence the result

Lemma 2.1.4

A sufficient condition for the polynomials (ax+bu) and (ay+bw) to have common zeros is that

$$x_0 w_0 - u_0 y_0 = 0$$

for some values of the set { (s_0, z_0) } which are not zeros of x or u.

Combining lemmas 2.1.1, 2.1.2, 2.1.3, and 2.1.4 will give the following theorem.

Thoerem 2.1.1

Sufficient conditions for the polynomials (ax+bu) and (ay+bw) to have common zeros are the following:

(i) The polynomials a,u, and w have common zeros.

(ii) The polynomials b,x, and y have common zeros.

(iii) The polynomial (ay+bw) is zero at the set of points which are common zeros of x and u, but not zeros of y or w.

(iv) The polynomials (ax+bu) and (xw-uy) have common zeros which are not zeros of x or u.

A necessary condition for the polynomials (ax+bu) and (ay+bw) to have common zeros is that one of the above four conditions must hold.

Directly from theorem 2.1.1, the actual result required can be found.

Theorem 2.1.2

A sufficient condition for two polynomials (ax+bu) and (ay+bw) to have no common zeros is that all of the following conditions must hold:

(i) The polynomials a,u, and w have no common zero.

(ii) The polynomials b,x, and y have no common zero.

(iii) The polynomial (ay+bw) must be non-zero at the set of points which are common zeros of x and u, but not zeros of y or w.

(iv) The polynomials (ax+bu) and (xw-uy) have no common zeros which are not zeros of x or u.

It can be seen from the above theorem that conditions (i) and (ii) are the constructions for the polynomials a and b. However from conditions (iii) and (iv) it is not possible to find explicit conditions on x, y, u, and w for the equivalence transformation to exist, or further construction for the polynomials a and b of the transforming matrices.

Therefore it is necessary to consider an alternative approach.

2.2.2 Method 2

Again the problem is to solve the equation

$$e(ax + bu) + g(ay + bw) = 1 \qquad (2.2.1)$$

for polynomials $a,b,e,g \in \mathbb{R}[s,z]$. This equation is solvable if and only if the polynomials

$$(ax + bu) \text{ and } (ay + bw) \qquad (2.2.2)$$

have no common zeros for general polynomials a and b which themselves have no common zero.

To investigate this consider the following proposition.

Proposition 2.2.1

For the polynomials (ax+bu) and (ay+bw) to have no common zero then

either

(i) They are never equal, and so cannot have any common value.

or

(ii) If they are equal, at those points where they are equal they are not zero.

For the polynomials (ax+bu) and (ay+bw) never to be equal it is necessary that

$$a(x - y) + b(u - w) \neq 0 \qquad (2.2.3)$$

for all values of (s,z), which immediately gives the result:

Lemma 2.2.1

A sufficient condition for the polynomials (ax+bu) and (ay+bw) to have no common zero is that the polynomials (x-y) and (u-w) have no common zero and then the polynomials a and b are chosen such that

$$a(x - y) + b(u - w) = 1.$$

Now if the polynomials (x-y) and (u-w) have common zeros then an analysis of the form used in method 1 is required. Unfortunately this gives results which will not explicitly determine whether a transformation of equivalence exists. This analysis also gives rise to conditions (i) and (ii) of theorem 2.1.2, showing consistency in the methods.

To illustrate this method consider the following example.

Example 2.2.1

For the matrix

$$\begin{bmatrix} s & -(1+z) \\ -z & s \end{bmatrix} \tag{2.2.4}$$

all adjacent pairs of polynomials have common zeros, but the matrix has no zeros.

Here

$$(x - y) = s+z+1 \tag{2.2.5}$$

$$(u - w) = -(s+z) \tag{2.2.6}$$

and it can easily be seen that (x-y) and (u-w) have no common zero. Choosing a = 1, b = 1 gives

$$1.(s+z+1) + 1.(-(s+z)) = 1 \tag{2.2.7}$$

which gives the equivalence transformation

$$\begin{bmatrix} 1 & 1 \\ s+z & s+z+1 \end{bmatrix} \begin{bmatrix} s & -(z+1) \\ -z & s \end{bmatrix} \begin{bmatrix} 1 & -(s-z-1) \\ -1 & s-z \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & s^2-z(z+1) \end{bmatrix} \tag{2.2.8}$$

2.2.3 Method 3

This method considers equation (2.8) in a different way, that is

$$(ae)x + (ag)y + (be)u + (bg)w = 1 \qquad (2.3.1)$$

As.x,y,u, and w have no common zero then there exist polynomials $k,l,m,n \in \mathbb{R}[s,z]$ such that

$$kx + ly + mu + nw = 1. \qquad (2.3.2)$$

However this is true for many k,l,m,n. The problem now is to find k,l,m,n which are factorizable such that

$$\begin{aligned} ae &= k \\ ag &= l \\ be &= m \\ bg &= n \end{aligned} \qquad (2.3.3)$$

However this problem is not definitive and would require searching through all possible k,l,m,n until one set is found which is factorizable.

To illustrate this method consider the example.

Example 2.3.1

Consider the matrix

$$\begin{bmatrix} s & -(1+z) \\ -z & s \end{bmatrix} \tag{2.3.4}$$

which has no zeros. Choosing

$$k = 1,\ l = -1,\ m = 1,\ n = -1 \tag{2.3.5}$$

gives

$$1.(s) - 1.(-(1+z)) + 1.(-z) - 1.(s) = 1 \tag{2.3.6}$$

and k,l,m,n are factorizable giving

$$a = 1,\ b = 1,\ e = 1,\ g = -1 \tag{2.3.7}$$

which gives the equivalence transformation

$$\begin{bmatrix} 1 & 1 \\ s+z & s+z+1 \end{bmatrix} \begin{bmatrix} s & -(z+1) \\ -z & s \end{bmatrix} \begin{bmatrix} 1 & -(s-z-1) \\ -1 & s-z \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & s^2-z(z+1) \end{bmatrix} \tag{2.3.8}$$

2.2.4 Use of the resultant

Resultants are often used to determine whether two specific polynomials have no common zeros. The method of Bose (1976) to find the greatest common divisor of two multivariable polynomials uses this property. It is to be considered here whether resultants can be used to determine if the general polynomials

$$(ax + bu) \text{ and } (ay + bw) \tag{2.4.1}$$

have no common zeros. Here the polynomials x,u,y,w are specified, and a and b are assumed to be of the form

$$a = \sum_{i=0}^{m} a_i(z)\, s^i \tag{2.4.2}$$

$$b = \sum_{j=0}^{n} b_j(z)\, s^j \tag{2.4.3}$$

however m and n are unknown.

To use the resultant for these polynomials, values of m and n have to be assumed. The resultant will then produce conditions on the $a_i(z)$ and $b_j(z)$ such that the polynomials (ax+bu) and (ay+bw) have no common zeros. If it is not possible to find conditions on the $a_i(z)$ and $b_j(z)$, then alternative values of m and n have to be assumed. Therefore this method is not conclusive, as all possible pairs (m,n) may have to be considered.

Example 2.4.1

Consider the matrix

$$A(s,z) = \begin{bmatrix} s & -(1+z) \\ -z & s \end{bmatrix}$$

It is required to find polynomials a and b such that the polynomials

$$(as - bz) \text{ and } (bs - a(1+z))$$

have no common zeros.

Let a and b be of the form

$$a = \sum_{i=0}^{m} a_i(z)\, s^i$$
$$b = \sum_{j=0}^{n} b_j(z)\, s^j$$

Let $m = n = 0$.

Now resultant $((a_0(z)s - b_0(z)z) , (b_0(z)s - a_0(z)(1+z)))$

$$= \det \begin{bmatrix} a_0(z) & -b_0(z)z \\ b_0(z) & -a_0(z)(1+z) \end{bmatrix}$$

$$= -a_0^2(z)(1+z) + b_0^2(z)z$$

$\not\equiv R \neq 0$ if $a_0(z) = 1$, $b_0(z) = 1$.

That is the polynomials (s-z) and (s-z-1) have no common zeros.

2.2.5 Some observations

To conclude this section some relevant observations are made.

The only fully known statement about the problem is that the matrix K(s,z) has no zeros, that is there exist polynomials $k,l,m,n \in \mathbb{R}[s,z]$, such that

$$kx + ly + mu + nw = 1. \tag{2.5.1}$$

Refering back to theorem 2.1.2 we see that condition (i) requires that the polynomials a,u, and w have no common zeros, and condition (ii) requires that the polynomials b,x, and y have no common zeros. If we choose a and b such that

$$a = kx + ly \tag{2.5.2}$$

$$b = mu + nw \tag{2.5.3}$$

then by equation (2.5.1)

$$a + mu + nw = 1 \tag{2.5.4}$$

$$b + kx + ly = 1 \tag{2.5.5}$$

which gives the result that a,u,w have no common zeros and b,x,y have no common zeros. We also have the extra result from (2.5.1), (2.5.2), (2.5.3) that

$$a + b = 1 \qquad (2.5.6)$$

Therefore if $a = kx + my$, $b = 1 - a$ it is necessary only to consider conditions (iii) and (iv) of theorem 2.1.2 on the polynomials

$$(a(x - u) + u) \text{ and } (a(y - w) + w) \qquad (2.5.7)$$

Use of these observations may assist in constructing a transformation of equivalence for specific examples. However for the general case these still do not give explicit conditions.

In conclusion it is obvious that an alternative necessary and sufficient condition to determine whether two polynomials have common zeros is required. This would be used in conjunction with Hilberts Nullstellensatz to construct the transformation of equivalence.

2.3 Extensions of the results of Lee and Zak (1981)

As shown in section 3 of chapter 1, Lee and Zak were interested in matrices of the form

$$[\ sI - A(z)\ ] \tag{3.1}$$

which has Smith form

$$\left[\begin{array}{c|c} I & 0 \\ \hline 0 & \det(\ sI - A(z)\ ) \end{array}\right] \tag{3.2}$$

The extension initially proposed here is for the general 2x2 matrix

$$P(s,z) = \begin{bmatrix} t & u \\ v & w \end{bmatrix} \tag{3.3}$$

where $t = t(s,z) \in \mathbb{R}[s,z]$ etc., and $P(s,z)$ has Smith form

$$S(s,z) = \begin{bmatrix} 1 & 0 \\ 0 & tw-uv \end{bmatrix} \tag{3.4}$$

It is possible to extend the definition of a matrix being cyclic over $\mathbb{R}[z]$, to being cyclic over $\mathbb{R}[s,z]$.

Definition 3.1

The matrix $P(s,z) \in \mathbb{R}^{n\times n}[s,z]$ is cyclic over $\mathbb{R}[s,z]$ if and only if there exists a vector $b \in \mathbb{R}^{n}[s,z]$ such that

$$\det([b,Ab,....,A^{n-1}b]) \in \mathbb{R} \neq 0 \qquad (3.5)$$

The vector b is said to be a cyclic vector over $\mathbb{R}[s,z]$.

Therefore if the matrix P(s,z) of (3.3) is cyclic then there exists

$$b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \in \mathbb{R}^2[s,z] \qquad (3.6)$$

such that

$$\det([b,Ab]) = \det \begin{bmatrix} b_1 & tb_1 + ub_2 \\ b_2 & vb_1 + wb_2 \end{bmatrix} = 1 \qquad (3.7)$$

which gives the equation

$$b_1^2 v + b_1 b_2 (w-t) - b_2^2 u = 1 \qquad (3.8)$$

From this it is possible to construct the equivalence matrices which will transform P(s,z) into its Smith form,

$$\begin{bmatrix} -b_2 & b_1 \\ -(b_1v+b_2w) & (ub_2+b_1t) \end{bmatrix} \begin{bmatrix} t & u \\ v & w \end{bmatrix} \begin{bmatrix} b_1 & ub_2-b_1w \\ b_2 & b_1v-b_2t \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & tw-uv \end{bmatrix} \tag{3.9}$$

with

$$\det \begin{bmatrix} -b_2 & b_1 \\ -(b_1v+b_2w) & (ub_2+b_1t) \end{bmatrix} = 1 \tag{3.10}$$

$$\det \begin{bmatrix} b_1 & ub_2-b_1w \\ b_2 & b_1v-b_2t \end{bmatrix} = 1 \tag{3.11}$$

and

$$\begin{bmatrix} -b_2 & b_1 \\ -(b_1v+b_2w) & (ub_2+b_1t) \end{bmatrix} \begin{bmatrix} b_1 & ub_2-b_1w \\ b_2 & b_1v-b_2t \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & t+w \end{bmatrix} \tag{3.12}$$

In fact immediately from the matrix [b,Ab] it is possible to find the similarity matrices to transform P(s,z) into its companion form,

$$\begin{bmatrix} b_1 & tb_1+ub_2 \\ b_2 & vb_1+wb_2 \end{bmatrix} \begin{bmatrix} t & u \\ v & w \end{bmatrix} \begin{bmatrix} ub_1+wb_2 & -(tb_1+ub_2) \\ -b_2 & b_1 \end{bmatrix} = \begin{bmatrix} 0 & -(tw-uv) \\ 1 & t+w \end{bmatrix} \tag{3.13}$$

where [b,Ab] and $[b,Ab]^{-1}$ are the transforming matrices.

However there seems to be some confusion whether cyclic and non-derogatory are equivalent properties over $\mathbb{R}[s,z]$. A derogatory matrix is one for which the minimal polynomial is of lower degree than the characteristic polynomial.

Example 3.1

Consider a general 2x2 two variable polynomial matrix

$$A(s,z) = \begin{bmatrix} x(s,z) & y(s,z) \\ u(s,z) & w(s,z) \end{bmatrix}$$

If A(s,z) is derogatory then it has a linear minimal polynomial

$$\lambda - a = 0.$$

Therefore

$$\begin{bmatrix} x(s,z) & y(s,z) \\ u(s,z) & w(s,z) \end{bmatrix} - a(s,z) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

For this to hold we have

$$u(s,z) = y(s,z) = 0,$$

$$x(s,z) = a(s,z),$$

$$w(s,z) = a(s,z).$$

Therefore the only 2x2 derogatory matrices are of the form

$$\begin{bmatrix} a(s,z) & 0 \\ 0 & a(s,z) \end{bmatrix}$$

which is already in Smith form. Moreover the only 2x2 derogatory two variable matrix which has Smith form

$$\begin{bmatrix} 1 & 0 \\ 0 & p(s,z) \end{bmatrix}$$

is the identity matrix.

Therefore it must be the case that non-derogatory matrices over R[s,z] are not necessarily similar to their companion forms. This can be shown by the following example:

Example 3.2

Consider the matrix

$$A(s,z) = \begin{bmatrix} s & -(z+1) \\ -z^2 & s \end{bmatrix}$$

From the previous result, this matrix is non-derogatory. It has companion form

$$C(s,z) = \begin{bmatrix} 0 & 1 \\ -(s^2-z^2(z+1)) & 2s \end{bmatrix}$$

For A(s,z) to be similar to C(s,z), we must have

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} s & -(z+1) \\ -z^2 & s \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -(s^2-z^2(z+1)) & 2s \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

where a = a(s,z) etc. This gives

$$as - bz^2 = c$$
$$bs - a(z+1) = d$$
$$cs - dz^2 = 2cs - a(s^2-z^2(z+1))$$
$$ds - c(z+1) = 2ds - b(s^2-z^2(z+1))$$

and for unimodularity of the transforming matrix

$$ad - bc \in \mathbb{R} \neq 0.$$

Now

$$ad - bc = a(bs - a(z+1)) - b(as - bz^2)$$
$$= b^2z^2 - a^2(z+1)$$

$\not\in \mathbb{R} \neq 0$ because it is not sign definite.

Therefore A(s,z) is not similar to its companion form.

Obviously there is a difference between cyclic and non-derogatory matrices over $\mathbb{R}[s,z]$. However it is still possible to extend the result of Lee and Zak to $\mathbb{R}[s,z]$ using definition 3.1.

Theorem 3.1

Consider the cyclic matrix P(s,z) over $\mathbb{R}[s,z]$,

$$P(s,z) = [p_{ij}] \qquad i,j = 1,2,\ldots,n$$

where $p_{ij} \in \mathbb{R}[s,z]$, which has Smith form S(s,z) over $\mathbb{R}[s,z]$,

$$S(s,z) = \left[\begin{array}{c|c} I_{n-1} & 0 \\ \hline 0 & \det(P) \end{array}\right] \qquad (3.14)$$

Then P(s,z) is equivalent over $\mathbb{R}[s,z]$ to S(s,z).

proof (similar to that for the result of Lee and Zak 1981)

If P(s,z) is cyclic, then there exists a matrix H(s,z),

$$H(s,z) = [b, Ab, \ldots, A^{n-1}b]$$

where b is the cyclic vector. Now P(s,z) is similar to its

companion form C(s,z) with similarity matrix H(s,z),

$$H^{-1}(s,z)\ P(s,z)\ H(s,z) = C(s,z).$$

It can be shown that C(s,z) is equivalent over $\mathbb{R}[s,z]$ to S(s,z), the Smith form of P(s,z). Therefore if P(s,z) is cyclic over $\mathbb{R}[s,z]$ then it is equivalent to its Smith form over $\mathbb{R}[s,z]$.

2.4 Development of an algorithm to produce the Smith form of a two variable polynomial matrix

As section 2 of chapter 1 shows, there are a number of methods for finding the Smith form of a one variable polynomial matrix. This section extends algorithm 2.1 of chapter 1 to transform a two variable polynomial matrix into its Smith form.

As there does not exist a division algorithm over $R[s,z]$ it is necessary to design the Smith form algorithm to change from equivalence over $R[s,z]$ to equivalence over $R(z)[s]$ or $R[z][s]$ if required. The main problem area for the algorithm is the actual Gaussian elimination. It was noticed that choice of pivot was crucial, so that if a "better" pivot is available at any time it should be used.

Consider the m x n polynomial matrix $P(s,z)$,

$$P(s,z) = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{bmatrix} \qquad (4.1)$$

where $p_{ij} = p_{ij}(s,z) \in R[s,z]$.

Element p_{11} is the pivot. Firstly element p_{12} is divided by element p_{11} as far as possible, that is until no further division can be carried out without introducing

rational terms in one of the variables. This will produce a quotient and remainder such that

$$p_{12} = qp_{11} + r. \tag{4.2}$$

Here r may not be of lower degree than p_{11}. Then q times column 1 is subtracted from column 2. At this stage it is checked if there is an element of lower degree than the pivot. If there is, then it is moved to position (1,1) becoming the new pivot and elimination is restarted. Otherwise the next element on the pivotal row is considered and the same procedure followed.

This is carried out for the pivotal row and column resulting in one of two situations. Either all the elements of the pivotal row and column are zero, and so this stage of the elimination has been successful. Or there are some elements on the pivotal row or column which are not zero, and the present pivot cannot further divide any of these elements over R[s,z], and a better pivot is not available. If this is the case then the elimination must continue over R(z)[s] or R[z][s].

If equivalence over R[s,z] is possible then this technique is quite straightforward to implement. However for equivalence over R(z)[s] a few problems arise.

For a number of reasons it is necessary to change the definition of monic over $\mathbb{R}(z)[s]$. To make the pivot monic under the usual definition would require division of a column of the matrix by a polynomial in $\mathbb{R}(z)$. The division here would have to be complete, that is without a remainder. If the algorithm used a notation for a polynomial in $\mathbb{R}(z)[s]$ of the form

$$p(s,z) = n(s,z) \;/\; d(z) \tag{4.3}$$

where $p(s,z) \in \mathbb{R}(z)[s]$, $n(s,z) \in \mathbb{R}[s,z]$, $d(z) \in \mathbb{R}[z]$, then this would cause no problem. However the algorithm uses a series expansion as its definition, that is for $p(s,z) \in \mathbb{R}(z)[s]$

$$p(s,z) = \sum_{i=0}^{r} \sum_{j=-l}^{m} a_{ij}\, s^i z^j \tag{4.4}$$

For this definition, dividing by a polynomial in $\mathbb{R}(z)$ may result in an infinite series expansion of the polynomial, such that at any stage of the division there would always be a remainder.

It was thought that if the pivot was made monic only after the elimination was complete, with pivotal row and column having all remaining elements zero, this problem could be overcome. However this still affects the

equivalence matrices.

Another effect of dividing through by a polynomial in $\mathbb{R}(z)$ with the notation (4.4) is that polynomial factors in $\mathbb{R}[z]$ may be lost when the matrix is renormalized using the method of Morf et al (1977) as described in section 3 of chapter 1.

Therefore it was decided to change the definition of monic over $\mathbb{R}(z)[s]$ to be that the coefficient of the leading term in s, is purely rational in z with leading term 1, that is by dividing through by a monomial in z.

It is worth expanding on the idea of Morf et al for renormalizing the equivalence over $\mathbb{R}(z)[s]$.

Consider the equivalence of a matrix A(s,z) with its Smith form $S_S(s,z)$ over $\mathbb{R}(z)[s]$,

$$M(s,z)\ A(s,z)\ N(s,z) = S_S(s,z) \qquad (4.5)$$

where M(s,z), N(s,z), $S_S(s,z)$ are matrices over $\mathbb{R}(z)[s]$ and M(s,z), N(s,z) are unimodular over $\mathbb{R}(z)[s]$. Now if M(s,z) and N(s,z) are renormalized by diagonal matrices over $\mathbb{R}[z]$ so that

$$M'(s,z)\ A(s,z)\ N'(s,z) = S_S'(s,z) \qquad (4.6)$$

and $M'(s,z), N'(s,z), S_S'(s,z)$ are now matrices over $\mathbb{R}[z][s]$

with M'(s,z) and N'(s,z) unimodular over R[z][s]. Now $S_S(s,z)$ is correct to within removed polynomial factors in z, and $S_S'(s,z)$ is correct to within added polynomial factors in z of the actual Smith form, S(s,z), over R[s,z].

It can be seen that the new definition of monic over R(z)[s] helps the renormalization, because the diagonal matrices over R[z] need only have monomials in z as their elements.

If the same procedure is repeated for equivalence over R(s)[z], and then renormalized to equivalence over R[s][z], then the resulting Smith form $S_Z'(s,z)$ will be correct to within added polynomial factors in s of the actual Smith form, S(s,z), over R[s,z].

Therefore it can be seen that the greatest common divisor of the corresponding elements of the Smith forms $S_S'(s,z)$ and $S_Z'(s,z)$ will give the actual Smith form over R[s,z].

So the strategy of the algorithm would be to use the Gaussian elimination initially over R[s,z]. If it is not possible to complete the equivalence over R[s,z], then elimination will be continued over R(z)[s] to produce the Smith form $S_S(s,z)$. If required the equivalence will be attempted over R[z,s] producing either the Smith form S(s,z) over R[z,s], or the Smith form $S_Z(s,z)$ over R(s)[z]. If $S_S(s,z)$ and $S_Z(s,z)$ are found then these can be renormalized

to $S_s'(s,z)$ and $S_z'(s,z)$ and the greatest common divisor of the elements taken to give the correct Smith form over $\mathbb{R}[s,z]$.

A further extension of this would be to consider equivalence over $\mathbb{R}[z][s]$ instead of equivalence over $\mathbb{R}(z)[s]$, when equivalence over $\mathbb{R}[s,z]$ is not possible. This would remove the need to renormalize the matrices $M(s,z)$ and $N(s,z)$. Equivalence over $\mathbb{R}[z][s]$ can be achieved by multiplying rows and columns by suitable polynomials in $\mathbb{R}[z]$ to ensure that division by the pivot is always possible, that is

$$k(z)\ p_{12}(s,z) = q(s,z)\ p_{11}(s,z) + r(s,z) \qquad (4.7)$$

where $p_{11}(s,z)$, $p_{12}(s,z)$, $q(s,z)$, $r(s,z)$ are in $\mathbb{R}[z][s]$, and the degree in s of $r(s,z)$ is less than the degree in s of $p_{11}(s,z)$. This will ensure that elimination will terminate with all the elements of the pivotal row and column being zero.

Use of equivalence over $\mathbb{R}[z][s]$ also removes the problems of the definition of monic and of handling rational terms in one of the variables, which makes the implementation of the algorithm easier.

2.5 Production of a computer program to implement the algorithm

The choice of computing language to implement this algorithm is very important. It has to be able to handle the required representation of a polynomial matrix, that is an array of arrays of real numbers. The language should be capable of using recursion and conditional loops, as these will form an important part of the algorithm. Therefore ALGOL 68 was chosen as the most suitable language, as it is extremely flexible and adaptable to a particular user's requirements.

In this section the various techniques and problems associated with writing the program will be outlined. The design of the algorithm and the production of a computer program have been carried out hand in hand as care must be taken to design an algorithm which will be relatively easy to program and will run efficiently.

2.5.1 Representation of a two variable polynomial matrix

The obvious way of representing a two variable polynomial is by an array of coefficients. A polynomial p(s,z) is represented by

$$p(s,z) = \begin{array}{c} \\ 0 \\ \vdots \\ i \\ \vdots \\ q \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & \cdots & j & \cdots & r \end{array} \\ \begin{bmatrix} x & \cdots & x & \cdots & x \\ \vdots & & \vdots & & \vdots \\ x & \cdots & x & \cdots & x \\ \vdots & & \vdots & & \vdots \\ x & \cdots & x & \cdots & x \end{bmatrix} \end{array} \qquad (5.1.1)$$

where element (i,j) is the coefficient of $s^i z^j$ of p(s,z) and

q = maximum power of s

r = maximum power of z.

Algol 68 allows for any integer indexing of arrays, so that in this case the constant term $s^0 z^0$ is easily seen. Also if it is required to have negative powers of one (or both) variables then this can be easily implemented. As storage is an important criterion in efficient programming, use will be made of flexible arrays where the size of the array can increase or decrease as required. Obviously this is a useful property as in the elimination degrees of polynomials will be decreasing, and hence it would be sensible to decrease the array representation as well. To ensure that this is

carried out, after operations on the polynomial, a small procedure will remove rows and columns of zeros until there is at least one non zero coefficient in the qth row and rth column. This is operator % (see appendix). This compares favourably with other languages which require the maximum dimensions of an array to be declared before use, which is very expensive in stack size.

A polynomial matrix would be normally represented by an array of polynomials, or more specifically as an array of arrays of coefficients. However arrays of arrays cannot be defined in Algol 68. It would be possible to use four dimensional arrays where element (i,j,k,l) would be the coefficient of $s^k z^l$ of the (i,j)th element of the polynomial matrix. However this again would be inefficient in the use of the stack. To overcome this problem, a slight change is made in the definition of polynomials using the "structure" mode in Algol 68, namely that a polynomial is now a structure of an array of coefficients. As Algol 68 allows the user to define his own modes, the following mode declaration is used for a polynomial:

```
'mode' 'poly' = 'struct' ([0:0'flex',0:0'flex'] 'real' p),
```

and simply for an m x n polynomial matrix K,

```
[1:m,1:n] 'poly' K.
```

2.5.2 Arithmetical operators

All the normal arithmetic operators +, - (both monadic and diadic), *, / have to be defined for two variable polynomials. The operators +, -, * are all defined fairly easily by operations on the coefficients of the polynomials. They can be defined using the 'op' operator mode of Algol 68 as they use only one or two parameters.

However the division operator causes difficulties as a division algorithm does not exist over R[s,z]. The operator / carries out normal long division either over R[s,z], R(z)[s] or R[z][s] depending on whether a global logical flag "rat" is set as true or false. It is useful to look into the division operator in greater depth.

The division operator finds the leading term of both polynomials and applies long division between them. Over R[s,z] the long division will continue until either the remainder is of lower degree than the divisor, or it is not possible to continue the division without introducing rational terms in one of the variables. It should be noted that it may be necessary to re-dimension the array of coefficients of either the remainder or the quotient. This arises because although the degree over R[s,z] of the remainder will be reduced, the degree in one of the variables may increase. Over R(z)[s] it may be necessary to re-dimension the quotient or remainder due to either

increasing degree in z or the need for more rational terms in z. Over $\mathbb{R}(z)[s]$ the division will terminate when the degree in s of the remainder is less than the degree in s of the divisor. However there is another case when division must stop. Consider dividing the polynomial $p_1 = s$ by $p_2 = z+1$. After one step of the division

$$s = (z^{-1}s)(z+1) - z^{-1}s \qquad (5.2.1)$$

that is $q = z^{-1}s$, $r = -z^{-1}s$. Now the degree in s of r is greater than the degree in s of p_2. But it can be seen that this division will never terminate as r will always have greater degree than p_2. Therefore when it can be detected that the division would never terminate, an extra criterion for halting the division would be when the sum of the powers of the leading term of r is less than the sum of the powers of the leading term of p_2. In (5.2.1) division would now stop as the sum of the powers of the leading term of $r = -1+1 = 0$, and the sum of the powers of the leading term of $p_2 = 1$.

Division over $\mathbb{R}[z][s]$ follows the same lines as division over $\mathbb{R}[s,z]$ except for the different definitions of degree and leading term.

2.5.3 Input-ouput

To run the program the data must be input in the following format:

1) Two integers for the size of the matrix.
2) For each of the elements of the matrix, row by row,
   Two integers for the degrees in s and z of the element.
   The array of real coefficients of that element.

Given the data in this form the program will construct the required polynomial matrix ready for the elimination.

The form of the data output has been designed to give the user all the relevant information while trying to keep output volume to a minimum. When a polynomial matrix is output, the position of each element will be given followed by its array of coefficients. Any zero elements will not be printed to save on output volume.

Example 5.3.1

The following output

```
[2,3]
1.0  0.0  3.0  0.0
0.0  1.0  0.0  2.0
```

indicates that the (2,3)th element of the matrix is the polynomial

$$2sz^3 + sz + 3z^2 + 1.$$

The following information will be printed:

(i) The initial polynomial matrix.

(ii) For each transformation, whether over $\mathbb{R}[s,z]$, $\mathbb{R}[z][s]$, $\mathbb{R}[s][z]$, $\mathbb{R}(z)[s]$, or $\mathbb{R}(s)[z]$:

Both the equivalence matrices and their determinants.

The Smith form over the particular ring.

The actual matrix product of the equivalence matrices and the initial matrix.

(iii) Also for the rings $\mathbb{R}(z)[s]$ and $\mathbb{R}(s)[z]$:

The renormalized Smith form.

(iv) If the equivalence was not completed over $\mathbb{R}[s,z]$:

The calculated Smith form over $\mathbb{R}[s,z]$.

The determinants are printed as a check for unimodularity over the various rings. The actual matrix product is printed as a check that there has been no errors in the calculation of the equivalence matrices.

2.5.4 The greatest common divisor procedure

The greatest common divisor method of Bose (1976) as described in section 3 of chapter 1 is ideal for two variable polynomials. Firstly it is necessary to extract the content from each polynomial, leaving the primitive part. Then

$$\gcd[f,g] = \{\gcd[\text{cont}(f),\text{cont}(g)]\}\{\gcd[\text{pp}(f),\text{pp}(g)]\} \quad (5.4.1)$$

where cont(f) is the content of f, and pp(f) is the primitive part of f, f and g being the two polynomials under consideration.

This would involve recursion to first of all extract the contents of f and g, and then find their greatest common divisor.

If

$$f(s,z) = a_0(z)s^n + ... + a_n(z) \quad (5.4.2)$$

$$g(s,z) = b_0(z)s^m + ... + b_m(z) \quad (5.4.3)$$

then

$$\text{cont}(f) = \gcd[a_0,....,a_n] \quad (5.4.4)$$

$$\text{cont}(g) = \gcd[b_0,....,b_m] \quad (5.4.5)$$

The method has therefore to be programmed such that it can find the greatest common divisor of polynomials in one or two variables. It will also be necessary to consider polynomials of the form

$$p(s,z) = c_0(s)z^n + ... + c_n(s) \qquad (5.4.6)$$

Therefore when the procedure is called it has to decide whether it is dealing with

(i) polynomials in two variables.

(ii) polynomials in s only.

(iii)polynomials in z only.

This is important to ensure that further recursion is not carried out, and the bigradient matrix is correctly constructed.

In the method it is also necessary to evaluate the determinants of two variable polynomial matrices. Normally numerical techniques to evaluate determinants are based on equivalence or similarity of the original matrix to a triangular form. An example of this is of course Gaussian elimination.

Obviously this is not appropriate for two variable polynomial matrices. Therefore it is necessary to evaluate the determinants from a basic definition, that is by expansion along the first row of the matrix, and using

recursion to evaluate the determinants of the corresponding lower order minors.

It was found however that the method was very expensive in terms of storage and computing time. To overcome this the method of Blankinship (1963) was used for the greatest common divisors of single variable polynomials when required, this being called as a default from the main recursive greatest common divisor routine of Bose (1976).

2.6 An algorithm developed from the definition of the Smith form

Because it is not always possible to find the transformation of equivalence between a two variable polynomial matrix and its Smith form over $\mathbb{R}[s,z]$ it seems desirable to investigate the development of an algorithm which produces the Smith form directly from the determinantal divisors of the original matrix.

Obviously this technique would be very time consuming especially as the dimensions of the matrix increase. However with the following ideas the method can be made quite efficient.

To calculate the Smith form in this way it would seem necessary to calculate all the determinants of the minors of a given order, and then find their greatest common divisor. From practice it is seen that, especially with the lower order minors, the greatest common divisor is often unity. It seems that it is only necessary to calculate the determinants one by one until the greatest common divisor is unity or all the minors have been considered.

However it was noticed that the calculation of the determinants by expansion along the first row and recursion for lower order determinants is itself very time consuming. It was felt that the best way to overcome this was to make use of any previously calculated determinants of lower

order. That is using expansion along the first row and a "table lookup" technique to find the value of the determinant of the corresponding lower order minor. This removes the need to use recursion and so keeps the running time down. However using this method requires all the determinants of minors of a given order to be evaluated, but tests show that there is still a definite saving in running time.

The "table lookup" technique requires a matrix, MINN, of real numbers which correspond to the determinants, and two matrices, RN and CN, the rows of which are the r-tuples of the selected rows and columns of the rth order minors. The ith row of RN and the jth row of CN correspond to the rth order minor which has determinant stored in MINN(i,j). From this it is possible to form RN1, CN1, MINN1 which correspond to the (r+1)th order minors of the matrix.

A comparison of the computing times of this method and the ones using the Gaussian elimination will be given for various examples in the results section.

2.7 The Smith-McMillan form algorithm

This section considers the production of the Smith-McMillan form of a rational matrix. It is first necessary to define the Smith-McMillan form over $\mathbb{R}[s,z]$.

Definition 7.1

Consider a p x q rational matrix K(s,z) where the elements of K(s,z) are

$$n_{ij}(s,z) \;/\; d_{ij}(s,z) \qquad \begin{matrix} i = 1,\ldots,p \\ j = 1,\ldots,q \end{matrix} \qquad (7.1)$$

and $n_{ij}(s,z),\ d_{ij}(s,z) \in \mathbb{R}[s,z]$.

The Smith-McMillan form of K(s,z) is defined to be the matrix

$$M(s,z) = \begin{cases} [\ E(s,z) \;\vdots\; 0\ ] & ;\ p<q \\ E(s,z) & ;\ p=q \\ \begin{bmatrix} E(s,z) \\ \hline 0 \end{bmatrix} & ;\ p>q \end{cases} \qquad (7.2)$$

where $E(s,z) = \mathrm{diag}[\ e_i(s,z) \;/\; g_i(s,z)\ ]$,

and $e_i(s,z)$, $g_i(s,z)$ are relatively prime polynomials in $\mathbb{R}[s,z]$ with the divisibility property

$$e_1(s,z) \mid e_2(s,z) \mid \ldots \mid e_r(s,z)$$

$$g_r(s,z) \mid g_{r-1}(s,z) \mid \ldots \mid g_1(s,z)$$

where r = rank(K(s,z)), and the $e_i(s,z)$ and $g_i(s,z)$ are related as in equations 7.4 and 7.5.

The algorithm will follow along the same lines as those outlined for single variable rational matrices described in section 2 of the introduction.

Given a rational matrix K(s,z), firstly find the least common denominator of the elements of K(s,z), say d(s,z) in $\mathbb{R}[s,z]$. Then form the matrix N(s,z) such that

$$K(s,z) = N(s,z) \;/\; d(s,z) \qquad (7.3)$$

where N(s,z) is a matrix over $\mathbb{R}[s,z]$. Now calculate the Smith form, S(s,z), of N(s,z) using the algorithm of section 4 (that is either by equivalence over $\mathbb{R}[s,z]$ or by the joint equivalence over $\mathbb{R}[z][s]$ and $\mathbb{R}[s][z]$). The Smith-McMillan form of K(s,z) is now the matrix

$$M(s,z) = S(s,z) \;/\; d(s,z) \qquad (7.4)$$

If the polynomial d(s,z) is now used such that

$$S_{ii}(s,z) \;/\; d(s,z) = e_i(s,z) \;/\; g_i(s,z) \qquad (7.5)$$

where $e_i(s,z)$ and $g_i(s,z)$ are relatively prime, then M(s,z) is of the required form in definition 7.1.

Because the Smith-McMillan form algorithm is an extension of the Smith form algorithm it is quite straightforward to implement as a computer program. However because of the large number of times the greatest common divisor routine is used it is very time consuming.

2.8 Results

As a test of the algorithms a number of examples which have arisen in the work of Frost (1979) and Lee and Zak (1981) are used. These are used because of the relevance to the algebraic difficulties outlined earlier in this chapter.

Results will be given over the rings R(z)[s] and R[z][s] when required, comparing the two algorithms. A table will also be given of the computing times for the two main algorithms and the algorithm based on the determinantal divisors.

Example 8.1 (Frost 1979)

Consider the 4x5 matrix K(s,z) which has no zeros:

$$k(s,z) = \begin{bmatrix} s+1 & z(sz+1) & 0 & z(s+1) & z^2 \\ s & sz+1 & -(s+1)(s+z) & sz & z^2 \\ 0 & s(s+1) & s+1 & s(s+1) & sz^2(s+1) \\ s+1 & z(s+1)+1 & -(s+1)(s+z) & z(s+1) & z^2 \end{bmatrix}$$

This was found to be equivalent over $\mathbb{R}[s,z]$ to its Smith form

$$S(s,z) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & s+1 & 0 & 0 \\ 0 & 0 & 0 & s(s+1)(s+z) & 0 \end{bmatrix}$$

with equivalence matrices

$$M(s,z) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ s^2(s+1) & -s(s+1)^2 & 1 & 0 \\ s^2(s+1)(s+z)-1 & s(s+z)(s+1)^2 & s+z & 1 \end{bmatrix}$$

$$N(s,z) = \begin{bmatrix} -z & -z(s+1)-1 & -(s+1)(s+z) & s(s+1)(s+z)-z & z^3 \\ 1 & s+1 & (s+z)(s+1)^2 & -s(s+z)(s+1)^2 & -z^2 \\ 0 & 0 & 1 & -s & 0 \\ 0 & 0 & -(s+z)(s+1)^2 & s(s+z)(s+1)^2+1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Example 8.2

Consider the 2x2 matrix K(s,z) which has no zeros:

$$K(s,z) = \begin{bmatrix} s & s+z+1 \\ sz & z \end{bmatrix}$$

It was not possible to directly find the Smith form over $\mathbb{R}[s,z]$. Below are the various Smith forms formed by the different equivalences:

over $\mathbb{R}(z)[s]$

The Smith form was

$$\begin{bmatrix} 1 & 0 \\ 0 & s(s+z) \end{bmatrix}$$

with equivalence matrices

$$\begin{bmatrix} 1 & -z^{-1}(s+z) \\ -z & s+z+1 \end{bmatrix}, \quad \begin{bmatrix} 0 & z^{-1} \\ 1 & sz^{-1}(s+z-1) \end{bmatrix}$$

over R[z][s]

The Smith form was

$$\begin{bmatrix} z & 0 \\ 0 & sz(s+z) \end{bmatrix}$$

with equivalence matrices

$$\begin{bmatrix} 0 & 1 \\ z & -(s+z+1) \end{bmatrix}, \quad \begin{bmatrix} 0 & -1 \\ 1 & s \end{bmatrix}$$

over R(s)[z]

The Smith form was

$$\begin{bmatrix} 1 & 0 \\ 0 & z(z+s) \end{bmatrix}$$

with equivalence matrices

$$\begin{bmatrix} 1 & 0 \\ z(z+s-1) & 1 \end{bmatrix}, \quad \begin{bmatrix} -s^{-1}(z+s) & s^{-1}(z+s+1) \\ 1 & -1 \end{bmatrix}$$

over $\mathbb{R}[s][z]$

The Smith form was

$$\begin{bmatrix} s & 0 \\ 0 & zs(z+s) \end{bmatrix}$$

with equivalence matrices

$$\begin{bmatrix} 1 & 0 \\ -z & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & z+s+1 \\ 0 & -s \end{bmatrix}$$

These, when combined, give the calculated Smith form over $\mathbb{R}[s,z]$,

$$S(s,z) = \begin{bmatrix} 1 & 0 \\ 0 & sz(s+z) \end{bmatrix}$$

Example 8.3 (Lee and Zak 1981)

Consider the 2x2 matrix K(s,z) which has no zeros:

$$K(s,z) = \begin{bmatrix} s & -(z+1) \\ -z^2 & s \end{bmatrix}$$

It was not possible to find directly the Smith form over

$\mathbb{R}[s,z]$. Below are the various Smith forms found by the different equivalences.

over $\mathbb{R}(z)[s]$

The Smith form was

$$\begin{bmatrix} z^{-1}(z+1) & 0 \\ 0 & z^{-1}(z+1)(s^2-z^2(z+1)) \end{bmatrix}$$

with equivalence matrices

$$\begin{bmatrix} 1 & 0 \\ s & z+1 \end{bmatrix}, \quad \begin{bmatrix} 0 & z^{-1}(z+1) \\ -z^{-1} & sz^{-1} \end{bmatrix}$$

over $\mathbb{R}[z][s]$

The Smith form was

$$\begin{bmatrix} z+1 & 0 \\ 0 & (z+1)(s^2-z^2(z+1)) \end{bmatrix}$$

with equivalence matrices

$$\begin{bmatrix} 1 & 0 \\ -s & -(z+1) \end{bmatrix}, \quad \begin{bmatrix} 0 & -(z+1) \\ -1 & -s \end{bmatrix}$$

over $\mathbb{R}(s)[z]$

The Smith form was

$$\begin{bmatrix} 1 & 0 \\ 0 & z^2(z+1)-s^2 \end{bmatrix}$$

with equivalence matrices

$$\begin{bmatrix} 1 & 0 \\ s^{-1}(s^2-z^3) & 1 \end{bmatrix}, \quad \begin{bmatrix} -s^{-1}z & -(z+1) \\ -1 & -s \end{bmatrix}$$

over $\mathbb{R}[s][z]$

The Smith form was

$$\begin{bmatrix} s & 0 \\ 0 & s(z^2(z+1)-s^2) \end{bmatrix}$$

with equivalence matrices

$$\begin{bmatrix} 1 & 0 \\ z^2 & s \end{bmatrix}, \quad \begin{bmatrix} 1 & -(z+1) \\ 0 & -s \end{bmatrix}$$

These, when combined, give the calculated Smith form over $\mathbb{R}[s,z]$,

$$S(s,z) = \begin{bmatrix} 1 & 0 \\ 0 & s^2-z^2(z+1) \end{bmatrix}$$

Example 8.4 (Frost 1981)

Consider the 2x2 matrix K(s,z) which has no zeros:

$$K(s,z) = \begin{bmatrix} s & -(z+1) \\ -z & s \end{bmatrix}$$

It was not possible to find directly the Smith form over $\mathbb{R}[s,z]$. Below are the various Smith forms found by the different equivalences.

over $\mathbb{R}(z)[s]$

The Smith form was

$$\begin{bmatrix} -z^{-1}(z+1) & 0 \\ 0 & z^{-1}(z+1)(s^2-z(z+1)) \end{bmatrix}$$

with equivalence matrices

$$\begin{bmatrix} 1 & 0 \\ s & z+1 \end{bmatrix}, \begin{bmatrix} 0 & z^{-1}(z+1) \\ -z^{-1} & z^{-1}s \end{bmatrix}$$

over $\mathbb{R}[z][s]$

The Smith form was

$$\begin{bmatrix} z+1 & 0 \\ 0 & (z+1)(s^2-z(z+1)) \end{bmatrix}$$

with equivalence matrices

$$\begin{bmatrix} 1 & 0 \\ -s & -(z+1) \end{bmatrix}, \begin{bmatrix} 0 & -(z+1) \\ -1 & -s \end{bmatrix}$$

over $\mathbb{R}(s)[z]$

The Smith form was

$$\begin{bmatrix} 1 & 0 \\ 0 & z(z+1)-s \end{bmatrix}$$

with equivalence matrices

$$\begin{bmatrix} 1 & 0 \\ s^{-1}(s^2-z^2) & 1 \end{bmatrix}, \begin{bmatrix} -s^{-1}z & -(z+1) \\ -1 & -s \end{bmatrix}$$

over $\mathbb{R}[s][z]$

The Smith form was

$$\begin{bmatrix} s & 0 \\ 0 & s(z(z+1)-s^2) \end{bmatrix}$$

with equivalence matrices

$$\begin{bmatrix} 1 & 0 \\ z & s \end{bmatrix} , \begin{bmatrix} 1 & -(z+1) \\ 0 & -s \end{bmatrix}$$

These, when combined, give the calculated Smith form over $\mathbb{R}[s,z]$,

$$S(s,z) = \begin{bmatrix} 1 & 0 \\ 0 & s^2-z(z+1) \end{bmatrix}$$

note

By using the algebraic results shown earlier it is possible to find the transforming matrices over $\mathbb{R}[s,z]$, these are:

$$\begin{bmatrix} 1 & 1 \\ s+z & s+z+1 \end{bmatrix} , \begin{bmatrix} 1 & -(s-z-1) \\ -1 & s-z \end{bmatrix}$$

Example 8.5 (Frost 1979)

Consider the 3x3 matrix K(s,z) which has zeros:

$$K(s,z) = \begin{bmatrix} s & 0 & 0 \\ 0 & sz+1 & 1 \\ 0 & 0 & z \end{bmatrix}$$

It was not possible to find directly the Smith form over $\mathbb{R}[s,z]$. Below are the various Smith forms found by the different equivalences.

over $\mathbb{R}(z)[s]$

The Smith form was

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & z^{-1}s(sz+1) \end{bmatrix}$$

with equivalence matrices

$$\begin{bmatrix} 0 & 1 & 0 \\ z^2 & -z & 1 \\ sz+1 & -s & z^{-1}s \end{bmatrix}, \quad \begin{bmatrix} 0 & -z^{-1} & z^{-1}(sz+1) \\ 0 & -z^{-1} & s \\ 1 & z^{-1}(sz+1) & -s(sz+1) \end{bmatrix}$$

over $\mathbb{R}[z][s]$

The Smith form was

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & z & 0 \\ 0 & 0 & sz(sz+1) \end{bmatrix}$$

with equivalence matrices

$$\begin{bmatrix} 0 & 1 & 0 \\ z^2 & -z & 1 \\ -z(sz+1) & sz & -z \end{bmatrix}, \quad \begin{bmatrix} 0 & -1 & -(sz+1) \\ 0 & -1 & -sz \\ 1 & sz+1 & sz(sz+1) \end{bmatrix}$$

over $\mathbb{R}(s)[z]$

The Smith form was

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s^{-1}z(sz+1) \end{bmatrix}$$

with equivalence matrices

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & -z & 1 \end{bmatrix}, \quad \begin{bmatrix} 0 & s^{-1} & 0 \\ 0 & 0 & -s^{-1} \\ 1 & 0 & s^{-1}(sz+1) \end{bmatrix}$$

over $\mathbb{R}[s][z]$

The Smith form was

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & sz(sz+1) \end{bmatrix}$$

with equivalence matrices

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ z(sz+1) & -sz & s \end{bmatrix}, \quad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & -1 \\ 1 & -(sz+1) & sz+1 \end{bmatrix}$$

These, when combined, give the calculated Smith form over $\mathbb{R}[s,z]$,

$$S(s,z) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & sz(sz+1) \end{bmatrix}$$

Table 8.1

This table compares the computing times for the different algorithms for the previous examples.

Algorithm 1 uses equivalence over $\mathbb{R}[s,z]$ if possible. If it is not then the matrix is transformed over the rings $\mathbb{R}(z)[s]$ and $\mathbb{R}(s)[z]$, the Smith forms are then renormalized, and the calculated Smith form over $\mathbb{R}[s,z]$ found.

Algorithm 2 uses equivalence over $\mathbb{R}[s,z]$ if possible. If it is not then the matrix is transformed over the rings $\mathbb{R}[z][s]$ and $\mathbb{R}[s][z]$ and the calculated Smith form over $\mathbb{R}[s,z]$ found.

Algorithm 3 finds the Smith form over $\mathbb{R}[s,z]$ directly from the determinantal divisors of the matrix.

	Ex 8.1	Ex 8.2	Ex 8.3	Ex 8.4	Ex 8.5
Algorithm 1	55	27	49	39	42
Algorithm 2	47	22	26	25	34
Algorithm 3	322	9	9	9	15

The values are the mill units (approx 1 sec) for running the algorithms on the ICL 1904s.

The results given in table 8.1 show that algorithm 2 is the most efficient over the given examples, and is consistently faster than algorithm 1. As the number of minors in a matrix increases factorially with the size of the matrix, algorithm 3 can be seen to be ineffective on all but small matrices. In particular example 8.1 shows that even for a 4x5 matrix algorithm 3 is 7 times slower than algorithm 2. Therefore it is shown that algorithm 2 is the best algorithm.

Example 8.6

Consider the 3x3 rational polynomial matrix

$$\begin{bmatrix} 1/(s+z) & 1 & 1/(z+1) \\ 0 & 2s/(s^2-z^2) & (z+3)/(s+z) \\ s+z & 0 & (s-z)/(z+1) \end{bmatrix}$$

This can be rewritten

$$\frac{1}{(z+1)(s^2-z^2)} \begin{bmatrix} (z+1)(s-z) & (z+1)(s^2-z^2) & (s^2-z^2) \\ 0 & 2(z+1) & (z+3)(z+1)(s-z) \\ (z+1)(s+z)(s^2-z^2) & 0 & (s+z)(s^2-z^2) \end{bmatrix}$$

It was not possible to transform the new numerator matrix directly to its Smith form over $\mathbb{R}[s,z]$. The combined equivalence over $\mathbb{R}[z][s]$ and $\mathbb{R}[s][z]$ was necessary. This produces the correct Smith-McMillan form:

$$\begin{bmatrix} 1/(z+1)(s^2-z^2) & 0 & 0 \\ 0 & 1/(s+z) & 0 \\ 0 & 0 & (s+z)[(s+z)((s-z)(z^2+4z+3)-2)+2] \end{bmatrix}$$

Example 8.7

Consider the 3x3 rational polynomial matrix:

$$\begin{bmatrix} 1/z(sz+1) & 0 & 1/sz(sz+1) \\ 0 & 1/sz & 1/sz(sz+1) \\ 0 & 0 & 1/s(sz+1) \end{bmatrix}$$

This can be rewritten:

$$1/sz(sz+1) \begin{bmatrix} s & 0 & 1 \\ 0 & sz+1 & 1 \\ 0 & 0 & z \end{bmatrix}$$

The new numerator matrix was directly transformed into its correct Smith form over $\mathbb{R}[s,z]$.

This produces the correct Smith-McMillan form:

$$\begin{bmatrix} 1/sz(sz+1) & 0 & 0 \\ 0 & 1/sz(sz+1) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

CHAPTER 3

THE CONCEPT OF EXTENDED EQUIVALENCE

3.1 Introduction

As it was not possible to bring the algebraic work of section 2.2, for the equivalence of a matrix with its Smith form over R[s,z], to full completion it was decided to attempt an alternative equivalence transformation, that of extended equivalence. This is an equivalence of the form discussed by Pugh and Shelton (1978) based on the work of Fuhrmann (1977). The background to this concept is given in section 3.2. The concept is then applied to a 2x2 matrix over R[s,z] to investigate if there are any advantages over the usual equivalence transformation.

3.2 Background to the concept of extended equivalence of matrices over $\mathbb{R}[s]$

In this section the results of Pugh and Shelton will be summarized.

Definition 2.1

Two polynomial matrices $P_1(s)$, $P_2(s)$ are said to be extended equivalent if there exist matrices $M(s)$, $N(s)$ such that

$$M(s)\, P_1(s) = P_2(s)\, N(s) \tag{2.1}$$

and

$$M(s),\ P_2(s) \text{ are relatively left prime} \tag{2.2}$$

$$P_1(s),\ N(s) \text{ are relatively right prime} \tag{2.3}$$

For this equivalence $P_1(s)$ and $P_2(s)$ need not be of the same size.

An important result of extended equivalence is:

Lemma 2.1 (Pugh and Shelton 1978)

The matrices

$$\left[\begin{array}{c|c} T(s) & U(s) \\ \hline -V(s) & W(s) \end{array}\right] \tag{2.4}$$

and

$$\left[\begin{array}{c|c|c} I_{rl-r} & 0 & 0 \\ \hline 0 & T(s) & U(s) \\ \hline 0 & -V(s) & W(s) \end{array}\right] \tag{2.5}$$

are extended equivalent. That is, trivial expansion (or deflation) is an operation of extended equivalence.

The final result to complete the background is:

Lemma 2.2

If two matrices of the same size are equivalent, then they are extended equivalent.

3.3 Implementation of extended equivalence for a 2x2 two variable polynomial matrix

Consider the 2x2 two variable polynomial matrix

$$A(s,z) = \begin{bmatrix} x & y \\ u & w \end{bmatrix} \tag{3.1}$$

where $x = x(s,z) \in \mathbb{R}[s,z]$ etc., which has Smith form

$$S(s,z) = \begin{bmatrix} 1 & 0 \\ 0 & p(s,z) \end{bmatrix} \tag{3.2}$$

where $p(s,z) = \det(A(s,z)) = xw - uy$.

It is now possible to illustrate lemma 2.2 for a 2x2 matrix. Assume that the matrix $A(s,z)$ is equivalent to $S(s,z)$ over $\mathbb{R}[s,z]$, that is there exist unimodular matrices $M(s,z)$, $N(s,z)$ over $\mathbb{R}[s,z]$ such that

$$M(s,z)\ A(s,z)\ N(s,z) = S(s,z) \tag{3.3}$$

or

$$M(s,z)\ A(s,z) = S(s,z)\ N_1(s,z) \tag{3.4}$$

where $N_1(s,z) = N^{-1}(s,z)$.

(3.4) can be written in full as

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x & y \\ u & w \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & xw-uy \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} \quad (3.5)$$

where

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}, \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

are unimodular over $\mathbb{R}[s,z]$, that is

$$ad - bc = k_1 \in \mathbb{R} \neq 0 \quad (3.6)$$

$$eh - fg = k_2 \in \mathbb{R} \neq 0 \quad (3.7)$$

Consider now the last row of (3.5),

$$[\, c \quad d \,] \begin{bmatrix} x & y \\ u & w \end{bmatrix} = (xw - uy) [\, g \quad h \,] \quad (3.8)$$

which demonstrates extended equivalence. It is now required to prove that:

(i) $[\, c \quad d \,]$, $(xw - uy)$ are relatively left prime (3.9)

(ii) $\begin{bmatrix} x & y \\ u & w \end{bmatrix}$, $[\, g \quad h \,]$ are relatively right prime (3.10)

Now $[\ c \quad d\ ]$, $(xw - uy)$ are relatively left prime if

$$(xw - uy)q_1 + [\ c \quad d\ ] \begin{bmatrix} q_{21} \\ q_{22} \end{bmatrix} = 1 \qquad (3.11)$$

for some $q_1, q_{21}, q_{22} \in \mathbb{R}[s,z]$, that is

$$cq_{21} + dq_{22} + (xw-uy)q_1 = 1 \qquad (3.12)$$

which is the condition that the polynomials c, d, $(xw-uy)$ have no common zeros. But from (3.6) c and d have no common zeros, so (3.12) is satisfied. Therefore

$[\ c \quad d\ ]$ and $(xw-uy)$

are relatively left prime.

Now

$$\begin{bmatrix} x & y \\ u & w \end{bmatrix}, \quad [\ g \quad h\ ]$$

are relatively right prime if

$$\begin{bmatrix} q_{31} & q_{32} \\ q_{33} & q_{34} \end{bmatrix} \begin{bmatrix} x & y \\ u & w \end{bmatrix} + \begin{bmatrix} q_{41} \\ q_{42} \end{bmatrix} [\ g \quad h\ ] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad (3.13)$$

for some $q_{31}, q_{32}, q_{33}, q_{34}, q_{41}, q_{42} \in \mathbb{R}[s,z]$, that is

$$q_{31}x + q_{32}u + q_{41}g = 1 \tag{3.14}$$

$$q_{31}y + q_{32}w + q_{41}h = 0 \tag{3.15}$$

$$q_{33}x + q_{34}u + q_{42}g = 0 \tag{3.16}$$

$$q_{33}y + q_{34}w + q_{42}h = 1. \tag{3.17}$$

But as $\begin{bmatrix} x & y \\ u & w \end{bmatrix}$ is equivalent to $\begin{bmatrix} 1 & 0 \\ 0 & xw-uy \end{bmatrix}$, from (3.5) we have

$$ax + bu - e = 0 \tag{3.18}$$

$$ay + bw - f = 0 \tag{3.19}$$

$$cx + du - g(xw-uy) = 0 \tag{3.20}$$

$$cy + dw - h(xw-uy) = 0. \tag{3.21}$$

Choose $q_{31} = ah / k_2$

$q_{32} = bh / k_2$

$q_{41} = -f / k_2$

then (3.14) is satisfied from (3.18) and (3.7), and (3.15) is satisfied from (3.19).

Choose $q_{33} = -ag / k_2$

$q_{34} = -bg / k_2$

$q_{42} = e / k_2$

then (3.17) is satisfied from (3.19) and (3.7), and (3.16)

is satisfied from (3.18).

Therefore $\begin{bmatrix} x & y \\ u & w \end{bmatrix}$ and $[\, g \quad h \,]$ are relatively right prime, and so extended equivalence is a necessary condition for unimodular equivalence.

Consider now the proposition that the matrix

$$\begin{bmatrix} x & y \\ u & w \end{bmatrix} \tag{3.22}$$

is extended equivalent to its Smith form,

$$\begin{bmatrix} 1 & 0 \\ 0 & xw-uy \end{bmatrix} \tag{3.23}$$

But, by trivial deflation the Smith form (3.23) is extended equivalent to the polynomial

$$xw - uy.$$

Therefore we have

$$[\, a \quad b \,] \begin{bmatrix} x & y \\ u & w \end{bmatrix} = (xw - uy)\, [\, e \quad f \,] \tag{3.24}$$

with the conditions of relative primeness,

(i) $[a \quad b]$ and $(xw-uy)$ are relatively left prime. (3.25)

(ii) $\begin{bmatrix} x & y \\ u & w \end{bmatrix}$ and $[e \quad f]$ are relatively right prime. (3.26)

Now, expanding (3.24) gives

$$ax + bu = e(xw - uy) \tag{3.27}$$

$$ay + bw = f(xw - uy) \tag{3.28}$$

(3.27) multiplied by y, minus (3.28) multiplied by x gives

$$b(uy - wx) = (xw-uy)(ey - fx)$$

which implies

$$b = fx - ey \tag{3.29}$$

$$a = ew - fu \tag{3.30}$$

To satisfy (3.25), that is $[a \quad b]$ and $(xw - uy)$ are relatively left prime we require,

$$(xw - uy)q_1 + [\, a \quad b \,] \begin{bmatrix} q_{21} \\ q_{22} \end{bmatrix} = 1 \tag{3.31}$$

for some $q_1, q_{21}, q_{22} \in R[s,z]$. This may be written

$$(xw - uy)q_1 + aq_{21} + bq_{22} = 1$$

which, from (3.29) and (3.30) gives

$$(xw-uy)q_1 + (ew-fu)q_{21} + (fx-ey)q_{22} = 1 \qquad (3.32)$$

that is (xw-uy), (ew-fu), and (fx-ey) must have no common zeros.

To satisfy (3.26), we require

$$\begin{bmatrix} q_{31} & q_{32} \\ q_{33} & q_{34} \end{bmatrix} \begin{bmatrix} x & y \\ u & w \end{bmatrix} + \begin{bmatrix} q_{41} \\ q_{42} \end{bmatrix} [e \quad f] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad (3.33)$$

for some $q_{31}, q_{32}, q_{33}, q_{34}, q_{41}, q_{42} \in R[s,z]$. This may be written as the set of equations

$$q_{31}x + q_{32}u + q_{41}e = 1 \qquad (3.34)$$

$$q_{31}y + q_{32}w + q_{41}f = 0 \qquad (3.35)$$

$$q_{33}x + q_{34}u + q_{42}e = 0 \qquad (3.36)$$

$$q_{33}y + q_{34}w + q_{42}f = 1. \qquad (3.37)$$

Then by choosing

$$q_{31} = (q_1 w + q_{22} f)$$

$$q_{32} = -(q_1 y + q_{21} f)$$

$$q_{33} = -(q_1 u + q_{22} e)$$

$$q_{34} = (q_1 x + q_{21} e)$$

$$q_{41} = (q_{21} w - q_{22} y)$$

$$q_{42} = (q_{22} x - q_{21} u)$$

(3.34) and (3.37) will be satisfied from (3.32), and (3.35) and (3.36) will be satisfied by cancellation.

Therefore the relative right primeness condition (3.26) is satisfied by the relative left primeness condition (3.25). So all that is required to be proved is that equation (3.32) holds for some $q_1, q_{21}, q_{22} \in \mathbb{R}[s,z]$.

The problem reduces to finding polynomials e, f in R[s,z] such that the polynomials

(xw-uy), (ew-fu), (fx-ey)

have no common zeros. Then a and b are such that

a = ew-fu, b= fx-ey

and (3.24), (3.25), and (3.26) are all satisfied.

We note that this condition seems to be weaker than that of section 2.2 which requires that the polynomials

(fx-ey), (ew-fu)

have no common zeros. Obviously if (fx-ey) and (ew-fu) have no common zeros then (fx-ey), (ew-fu), and (xw-uy) have no

common zeros.

It is now worth investigating the existence of the common zeros of

$$(xw-uy),\ (ew-fu),\ (fx-ey).$$

Immediately it can be seen that it is necessary that

$$x,\ y,\ u,\ w$$

have no common zeros, that is the matrix

$$\begin{bmatrix} x & y \\ u & w \end{bmatrix}$$

has no zeros.

Now consider the set of points $\{ (s_1,z_1) \}$ the zeros of $(xw-uy)$, that is

$$x_1w_1 - u_1y_1 = 0 \qquad (3.38)$$

where $x_1 = x(s_1,z_1)$ etc.

Firstly consider the case when $\underline{x_1 = 0}$. It is now required that either $e_1y_1 \neq 0$ or $e_1w_1 - f_1u_1 \neq 0$ for the polynomials $(xw-uy)$, $(ew-fu)$, $(fx-ey)$ to have no common zeros. From (3.38) either $u_1 = 0$ or $y_1 = 0$ or both.

consider $x_1=0$, $y_1=0$

We require

$$e_1w_1 - f_1u_1 \neq 0 \tag{3.39}$$

Now if $u_1 = 0$, then (3.39) implies $e_1 \neq 0$. If $w_1 = 0$ then (3.39) implies $f_1 \neq 0$.

consider $x_1=0$, $u_1=0$

We require

$$\text{either } e_1y_1 \neq 0, \text{ or } e_1w_1 \neq 0 \tag{3.40}$$

This immediately implies that $e_1 \neq 0$.

Combining these will give the following result:

Lemma 3.1

Necessary conditions for the polynomials

(xw-uy), (ew-fu), (fx-ey)

to have no common zeros are

(i) The polynomials e, x, u have no common zeros.

(ii) The polynomials f, x, y, w have no common zeros.

Now consider the case when $x_1 \neq 0$.

consider $x_1 \neq 0$, $w_1 = 0$

We require

$$\text{either } f_1 u_1 \neq 0, \text{ or } f_1 x_1 - e_1 y_1 \neq 0 \tag{3.41}$$

$x_1 \neq 0$, $w_1 = 0$, $u_1 = 0$

In this case it is required that

$$f_1 x_1 - e_1 y_1 \neq 0. \tag{3.42}$$

If $y_1 = 0$, then (3.42) implies that $f_1 \neq 0$. If $y_1 \neq 0$ then (3.42) implies that $f_1 x_1 \neq e_1 y_1$.

$x_1 \neq 0$, $w_1 = 0$, $y_1 = 0$

Here it is required that

$$\text{either } f_1 u_1 \neq 0, \text{ or } f_1 x_1 \neq 0 \tag{3.43}$$

which immediately implies that $f_1 \neq 0$.

consider $x_1 \neq 0$, $w_1 \neq 0$

From (3.38) this immediately implies $u_1 \neq 0$, $y_1 \neq 0$. Now consider

$$f_1x_1 - e_1y_1$$

Multiplying by u_1 gives

$$\begin{aligned} u_1f_1x_1 - u_1e_1y_1 &= u_1f_1x_1 - e_1x_1w_1 \quad \text{from (3.38)} \\ &= x_1(f_1u_1 - e_1w_1) \end{aligned}$$

which implies that if $f_1x_1 - e_1y_1 = 0$ then $f_1u_1 - e_1w_1 = 0$.

Combining all these results gives:

Theorem 3.1

Necessary and sufficient conditions for the polynomials

(xw-uy), (ew-fu), (fx-ey)

to have no common zeros are:

(i) The polynomials e, x, u have no common zeros.

(ii) The polynomials f, y, w have no common zeros.

(iii) The polynomials (fx-ey), (xw-uy) have no common zeros which are not zeros of x, y, u, or w.

It is now seen that these conditions are exactly the same as those of theorem 2.1.2 of chapter 2, and unfortunately no stronger results have been obtained by considering this alternative approach.

CHAPTER 4

ROESSER MATRICES

4.1 Introduction

The previous chapters have considered two dimensional system matrices of the form

$$\left[\begin{array}{c|c} T(s,z) & U(s,z) \\ \hline -V(s,z) & W(s,z) \end{array}\right] \tag{1.1}$$

However this chapter will consider a special form of two dimensional system matrices, the Roesser matrix,

$$\left[\begin{array}{c|c|c} sI_n - a_1 & -a_2 & b_1 \\ \hline -a_3 & zI_p - a_4 & b_2 \\ \hline -c_1 & -c_2 & d \end{array}\right] \tag{1.2}$$

where a_1, a_2, a_3, a_4, b_1, b_2, c_1, c_2, and d are respectively nxn, nxp, pxn, pxp, nxl, pxl, mxn, mxp, and mxl matrices over R. These matrices are particular forms of a state-space system matrix. Such matrices arise naturally in the study of two dimensional systems (see, for example, Kung et al 1977) particularaly from the approach suggested for such systems by Givone and Roesser (1973) or Fornasini

and Marchesini (1975).

Roesser's model, which seems to be the most general two dimensional state-space model, can arise from considering two dimensional filters or image processing. Here

$$x(i,j) = \begin{bmatrix} x^h(i,j) \\ x^v(i,j) \end{bmatrix} \tag{1.3}$$

where x is the local state, x^h, an n-vector, is the horizontal state, x^v, a p-vector, is the vertical state and

$$\begin{bmatrix} x^h(i+1,j) \\ x^v(i,j+1) \end{bmatrix} = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \begin{bmatrix} x^h(i,j) \\ x^v(i,j) \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u(i,j) \tag{1.4}$$

$$y(i,j) = \begin{bmatrix} C_1 & C_2 \end{bmatrix} \begin{bmatrix} x^h(i,j) \\ x^v(i,j) \end{bmatrix} + Du(i,j) \tag{1.5}$$

for $i,j \geq 0$, is the discrete time model of the system. By taking (z,w) transforms a matrix of the form (1.2) will be produced.

Matrices of the form (1.2) may also arise directly from delay-differntial systems (see, for example, Zakian and Williams 1973, or Frost 1979).

This chapter builds up results for equivalence of a

Roesser matrix to its Smith form over $\mathbb{R}[s,z]$. To accomplish this the 2x2 and 3x3 Roesser matrices and other related matrices will be studied. Then by using mathematical induction on the indices l, m, n, and p the results will be extended to the (n+p+m)x(n+p+l) Roesser matrix (1.2).

4.2 The 2x2 and 3x3 Roesser matrices - Equivalence over $\mathbb{R}[s,z]$

Lemma 2.1

The 2x2 matrix

$$P(s,z) = \begin{bmatrix} s-a_1 & -a_2 \\ -a_3 & z-a_4 \end{bmatrix}$$

is equivalent to its Smith form over $\mathbb{R}[s,z]$ if and only if it has no zeros.

proof

If $P(s,z)$ has no zeros, then at least one of a_2, a_3 must be non zero, say a_2. Then using a_2 as a pivot, simple row and column operations will transform $P(s,z)$ into its Smith form

$$\begin{bmatrix} 1 & 0 \\ 0 & (s - a_1)(z - a_4) - a_2a_3 \end{bmatrix}$$

Lemma 2.2

The 2x2 matrix

$$P(s,z) = \begin{bmatrix} s-a_1 & z-a_2 \\ -a_3 & -a_4 \end{bmatrix}$$

is equivalent to its Smith form over $\mathbb{R}[s,z]$ if and only if it has no zeros.

proof

Similar to that for lemma 2.1.

Lemma 2.3

The 2x3 matrix

$$P(s,z) = \begin{bmatrix} s-a_1 & z-a_2 & b_1 \\ -a_3 & -a_4 & b_2 \end{bmatrix}$$

is equivalent to its Smith form over $\mathbb{R}[s,z]$ if and only if it has no zeros.

proof

Similar to that for lemma 2.1.

Lemma 2.4

The 3x3 matrix

$$P(s,z) = \begin{bmatrix} s-a_1 & -a_2 & b_1 \\ -a_3 & z-a_4 & b_2 \\ -c_1 & -c_2 & d \end{bmatrix}$$

is equivalent to its Smith form over $\mathbb{R}[s,z]$ if and only if it has no zeros.

proof

Assume $P(s,z)$ has no zeros.

Consider $d \neq 0$. Then using d as a pivot,

$$P(s,z) \sim \begin{bmatrix} 1 & 0 & 0 \\ 0 & z-a_4' & -a_3' \\ 0 & -a_2' & s-a_1' \end{bmatrix} \qquad (2.1)$$

Therfore, by lemma 2.1, $P(s,z)$ is equivalent to its Smith form over $\mathbb{R}[s,z]$ if and only if it has no zeros.

Now consider $d = 0$, $b_1 \neq 0$. Then using b_1 as a pivot,

$$P(s,z) \sim \begin{bmatrix} 1 & 0 & 0 \\ 0 & s-a_1' & z-a_4' \\ 0 & -c_1' & -c_2' \end{bmatrix} \qquad (2.2)$$

Therefore, by lemma 2.2, P(s,z) is equivalent to its Smith form over $\mathbb{R}[s,z]$ if and only if it has no zeros.

Now consider $d = 0$, $b_1 = 0$, $b_2 \neq 0$, and so on, giving the same result. Finally if d, b_1, b_2, c_1, c_2 are all zero then,

$$P(s,z) = \begin{bmatrix} s-a_1 & -a_2 & 0 \\ -a_3 & z-a_4 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

and immediately, by lemma 2.1, P(s,z) is equivalent to its Smith form over $\mathbb{R}[s,z]$ if and only if it has no zeros.

This has shown that the general 3x3 Roesser matrix is equivalent to its Smith form over $\mathbb{R}[s,z]$ if and only if it has no zeros. This result will be the initial condition used for the method of mathematical induction in the next section.

4.3 The general (n+p+m)x(n+p+1) Roesser matrix

Consider the (n+p+m)x(n+p+1) matrix

$$P(s,z) = \left[\begin{array}{c|c|c} sI_n - a_1 & -a_2 & b_1 \\ \hline -a_3 & zI_p - a_4 & b_2 \\ \hline -c_1 & -c_2 & d \end{array}\right] \qquad (3.1)$$

Lemma 2.4 proves that for $n = p = m = l = 1$ the matrix $P(s,z)$ is equivalent to its Smith form over $\mathbb{R}[s,z]$ if and only if it has no zeros. Therefore it would seem reasonable to extend this result to that for general n, p, m, l by the use of mathematical induction, in turn, on the indices m, l, n, and p.

To achieve this it will be necessary firstly to prove certain intermediate results required in the main induction.

Lemma 3.1

The matrix

$$R(s,z) = \left[\begin{array}{c|c} P(s,z) & 0 \end{array}\right]$$

is equivalent to its Smith form $S_R(s,z)$ over $\mathbb{R}[s,z]$ if and only if the matrix $P(s,z)$ is equivalent to its Smith form $S_P(s,z)$ over $\mathbb{R}[s,z]$.

Lemma 3.2

For all positive integers m, the (m+1)x3 matrix

$$P(s,z) = \left[\begin{array}{c|c|c} s-a_1 & z-a_2 & b_1 \\ \hline -a_3 & -a_4 & b_2 \end{array}\right] \begin{matrix} 1 \\ m \end{matrix}$$

is equivalent to its Smith form over $\mathbb{R}[s,z]$ if and only if it has no zeros.

proof: by induction on m.

By lemma 2.3 the result is true for $m = 1$.
Now assume that the result is true for $m = k$, that is any matrix

$$P(s,z) = \left[\begin{array}{c|c|c} s-a_1 & z-a_2 & b_1 \\ \hline -a_3 & -a_4 & b_2 \end{array}\right] \begin{matrix} 1 \\ k \end{matrix} \qquad (3.2)$$

which has no zeros, is equivalent to its Smith form over $\mathbb{R}[s,z]$.

Now consider the matrix

$$R(s,z) = \left[\begin{array}{c|c|c} s-a_1 & z-a_2 & b_1 \\ \hline -a_3 & -a_4 & b_2 \\ \hline r_1 & r_2 & r_3 \end{array}\right] \begin{matrix} 1 \\ k \\ 1 \end{matrix} \qquad (3.3)$$

where $r_1, r_2, r_3 \in R$ and $R(s,z)$ has no zeros. Trivially adding a row of zeros will maintain equivalence over $R[s,z]$, therefore assume that not all the r_i are zero.

If $r_3 \neq 0$, then using r_3 as a pivot

$$R(s,z) \sim \left[\begin{array}{c|c|c} 1 & 0 & 0 \\ \hline 0 & s-a_1' & z-a_2' \\ \hline 0 & -a_3' & -a_4' \end{array}\right] \tag{3.4}$$

which, by (3.3) (with $b_1 = b_2 = 0$) and lemma 3.1, is equivalent to its Smith form over $R[s,z]$ as $R(s,z)$ has no zeros.

Now if $r_3 = 0$, $r_1 \neq 0$, then using r_1 as a pivot

$$R(s,z) \sim \left[\begin{array}{c|c|c} 1 & 0 & 0 \\ \hline 0 & -a_4' & b_2 \\ \hline 0 & g_1s+z-a_2' & b_1 \end{array}\right] \tag{3.5}$$

If $[\ -a_4 \ \vdots\ b_2\ ]$ has rank zero, then it can be reduced by row and column operations to $[\ 0 \ \vdots\ 0\ ]$ leaving the two polynomials on the bottom row

$$g_1's+h_1z+a_2'' \quad , \quad g_2s+h_2z+b_1' \tag{3.6}$$

which, as $R(s,z)$ has no zeros, must have no common zeros and

so R(s,z) can be further reduced to its Smith form.

If $[\ -a_4 \ \vdots \ b_2\ ]$ has rank 1 or 2 the reduction will leave 1 or 2 constants on the diagonal which can be used to further reduce R(s,z) to its Smith form.

Now if $r_3 = 0$, $r_1 = 0$, $r_2 \neq 0$, then using r_2 as pivot

$$R(s,z) \sim \left[\begin{array}{c|c|c} 1 & 0 & 0 \\ \hline 0 & s-a_1 & b_1 \\ \hline 0 & -a_3 & b_2 \end{array}\right] \tag{3.7}$$

which is equivalent to its Smith form over $\mathbb{R}[s,z]$, as it is a single variable polynomial matrix.

Therefore R(s,z) is equivalent to its Smith form over $\mathbb{R}[s,z]$, and so the result is true for $m = k+1$.

Therefore, by mathematical induction on m, the result is true for all positive integer m.

Lemma 3.3

For all positive integers m and 1, the (m+1)x(1+2) matrix

$$P(s,z) = \begin{array}{c} \begin{array}{ccc} 1 & 1 & 1 \end{array} \\ \left[\begin{array}{c|c|c} s-a_1 & z-a_2 & b_1 \\ \hline -a_3 & -a_4 & b_2 \end{array}\right] \end{array} \begin{array}{c} 1 \\ m \end{array}$$

is equivalent to its Smith form over $\mathbb{R}[s,z]$ if and only if it has no zeros.

proof: by induction on l.

By lemma 3.2 the result is true for l = 1.

Now assume that the result is true for l = k, that is for all integer m the matrix

$$P(s,z) = \begin{array}{c} \begin{array}{ccc} 1 & 1 & k \end{array} \\ \left[\begin{array}{c|c|c} s-a_1 & z-a_2 & b_1 \\ \hline -a_3 & -a_4 & b_2 \end{array}\right] \end{array} \begin{array}{c} l \\ m \end{array} \qquad (3.8)$$

which has no zeros, is equivalent to its Smith form over $\mathbb{R}[s,z]$.

Now consider the matrix

$$R(s,z) = \begin{array}{c} \begin{array}{cccc} 1 & 1 & 1 & 1 \end{array} \\ \left[\begin{array}{c|c|c|c} s-a_1 & z-a_2 & b_1 & r_1 \\ \hline -a_3 & -a_4 & b_2 & r_2 \end{array}\right] \end{array} \begin{array}{c} l \\ m \end{array} \qquad (3.9)$$

where $r_1 \in \mathbb{R}$, $r_2 \in \mathbb{R}^m$, and R(s,z) has no zeros. Trivially a column of zeros can be added and equivalence will be maintained, therefore assume that at least on of the r_i is non zero.

If $r_2 \neq 0$, that is at least one element of r_2 is non

zero, then using this as a pivot,

$$R(s,z) \sim \left[\begin{array}{c|c|c|c} 1 & 0 & 0 & 0 \\ \hline 0 & s-a_1' & z-a_2' & b_1' \\ \hline 0 & -a_3' & -a_4' & b_2' \end{array}\right] \tag{3.10}$$

which, by (3.8), is equivalent to its Smith form over $\mathbb{R}[s,z]$.

Now if $r_2 = 0$, $r_1 \neq 0$, then by using r_1 as a pivot,

$$R(s,z) \sim \left[\begin{array}{c|c|c|c} 1 & 0 & 0 & 0 \\ \hline 0 & -a_3 & -a_4 & b_2 \end{array}\right] \tag{3.11}$$

which is a constant matrix and so can be further reduced to an identity block matrix, the Smith form of R(s,z) over $\mathbb{R}[s,z]$.

Therefore R(s,z) is equivalent to its Smith form over $\mathbb{R}[s,z]$, and so the result is true for $l = k+1$.

Therefore, by mathematical induction on l, the result is true for all positive integers m and l.

Lemma 3.4

For all positive integers m and l, the (m+2)x(l+l) matrix

$$P(s,z) = \begin{array}{c} \begin{matrix} \quad 1 & \quad 1 \end{matrix} \\ \left[\begin{array}{c:c} s-a_1 & -a_2 \\ \hdashline z-a_3 & -a_4 \\ \hdashline -c_1 & -c_2 \end{array}\right] \end{array} \begin{matrix} 1 \\ 1 \\ m \end{matrix}$$

is equivalent to its Smith form over $\mathbb{R}[s,z]$ if and only if it has no zeros.

proof

By considering the transpose of the matrix in lemma 3.3.

After proving these intermediate results it is now possible to apply the method of mathematical induction on the general Roesser matrix (3.1).

Lemma 3.5

For all positive integers m, the (m+2)x3 matrix

$$P(s,z) = \left[\begin{array}{c|c|c} s-a_1 & -a_2 & b_1 \\ \hline -a_3 & z-a_4 & b_2 \\ \hline -c_1 & -c_2 & d \end{array}\right] \begin{array}{l} 1 \\ 1 \\ m \end{array}$$

is equivalent to its Smith form over $\mathbb{R}[s,z]$ if and only if it has no zeros.

proof: by induction on m.

By lemma 2.4, the result is true for $m = 1$.

Assume that the result is true for $m = k$, that is any matrix

$$P(s,z) = \left[\begin{array}{c|c|c} s-a_1 & -a_2 & b_1 \\ \hline -a_3 & z-a_4 & b_2 \\ \hline -c_1 & -c_2 & d \end{array}\right] \begin{array}{l} 1 \\ 1 \\ k \end{array} \qquad (3.11)$$

which has no zeros is equivalent to its Smith form over $\mathbb{R}[s,z]$.

Now consider the matrix

$$R(s,z) = \left[\begin{array}{c|c|c|l} s-a_1 & -a_2 & b_1 & 1 \\ \hline -a_3 & z-a_4 & b_2 & 1 \\ \hline -c_1 & -c_2 & d & k \\ \hline r_1 & r_2 & r_3 & 1 \end{array}\right] \tag{3.12}$$

where $r_1, r_2, r_3 \in \mathbb{R}$ and $R(s,z)$ has no zeros.

Trivially adding a row of zeros will maintain equivalence, therefore assume that not all of the r_i are zero.

If $r_3 \neq 0$, then using r_3 as a pivot

$$R(s,z) \sim \left[\begin{array}{c|c|c} 1 & 0 & 0 \\ \hline 0 & s-a_1' & -a_2' \\ \hline 0 & -a_3' & z-a_4' \\ \hline 0 & -c_1' & -c_2' \end{array}\right] \tag{3.13}$$

which, from (3.12) and lemma 3.1, is equivalent to its Smith form over $\mathbb{R}[s,z]$.

Now if $r_3 = 0$, $r_2 \neq 0$, then using r_2 as pivot

$$R(s,z) \sim \left[\begin{array}{c|c|c} 1 & 0 & 0 \\ \hline 0 & s-a_1' & b_1 \\ \hline 0 & z-a_2' & b_2 \\ \hline 0 & -c_1' & d \end{array}\right] \tag{3.14}$$

which, from lemma 3.4, is equivalent to its Smith form over $\mathbb{R}[s,z]$.

Finally if $r_3 = 0$, $r_2 = 0$, $r_1 \neq 0$, then using r_1 as pivot

$$R(s,z) \sim \left[\begin{array}{c|c|c} 1 & 0 & 0 \\ \hline 0 & z-a_4 & b_2 \\ \hline 0 & -a_2 & b_1 \\ \hline 0 & -c_2 & d \end{array}\right] \tag{3.15}$$

which is equivalent to its Smith form over $\mathbb{R}[s,z]$ as it is a single variable polynomial matrix.

Therefore $R(s,z)$ is equivalent to its Smith form over $\mathbb{R}[s,z]$ and so the result is true for $m = k+1$.

Therefore by mathematical induction on m the result is true for all positive integers m.

Lemma 3.6

For all positive integers m and l, the (m+2)x(l+2) matrix

$$P(s,z) = \left[\begin{array}{c|c|c} s-a_1 & -a_2 & b_1 \\ \hline -a_3 & z-a_4 & b_2 \\ \hline -c_1 & -c_2 & d \end{array}\right] \begin{array}{c} l \\ l \\ m \end{array}$$

is equivalent to its Smith form over $\mathbb{R}[s,z]$ if and only if it has no zeros.

proof: by induction on l.

By lemma 3.5 the result is true for $l = 1$, and for all m.

Assume that the result is true for $l = k$, that is any matrix

$$P(s,z) = \left[\begin{array}{c|c|c} s-a_1 & -a_2 & b_1 \\ \hline -a_3 & z-a_4 & b_2 \\ \hline -c_1 & -c_2 & d \end{array}\right] \begin{array}{c} l \\ l \\ m \end{array} \qquad (3.16)$$

which has no zeros is equivalent to its Smith form over $\mathbb{R}[s,z]$ for all integer m.

Consider the matrix

$$R(s,z) = \left[\begin{array}{c|c|c|c} s-a_1 & -a_2 & b_1 & r_1 \\ \hline -a_3 & z-a_4 & b_2 & r_2 \\ \hline -c_1 & -c_2 & d & r_3 \end{array}\right] \begin{matrix} 1 \\ 1 \\ m \end{matrix} \qquad (3.17)$$

where $r_1, r_2 \in \mathbb{R}$, $r_3 \in \mathbb{R}^m$ and $R(s,z)$ has no zeros.

Trivially, adding a column of zeros will maintain equivalence, therefore assume that not all the r_i are zero.

If $r_3 \neq 0$, that is at least one element of r_3 is non zero, then using that element as pivot

$$R(s,z) \sim \left[\begin{array}{c|c|c|c} 1 & 0 & 0 & 0 \\ \hline 0 & s-a_1' & -a_2' & b_1' \\ \hline 0 & -a_3' & z-a_4' & b_2' \\ \hline 0 & -c_1' & -c_2' & d' \end{array}\right] \begin{matrix} 1 \\ 1 \\ 1 \\ m-1 \end{matrix} \qquad (3.18)$$

which, from (3.16), is equivalent to its Smith form over $\mathbb{R}[s,z]$, as (3.16) is true for all positive integers m.

Now if $r_3 = 0$, $r_1 \neq 0$, then using r_1 as pivot

$$R(s,z) \sim \left[\begin{array}{c|c|c|c} 1 & 0 & 0 & 0 \\ \hline 0 & s-a_3' & z-a_4' & b_1' \\ \hline 0 & -c_1 & -c_2 & d \end{array}\right] \tag{3.19}$$

which, from lemma 3.3, is equivalent to its Smith form over $\mathbb{R}[s,z]$.

Finally if $r_3 = 0$, $r_1 = 0$, $r_2 \neq 0$, then using r_2 as pivot

$$R(s,z) \sim \left[\begin{array}{c|c|c|c} 1 & 0 & 0 & 0 \\ \hline 0 & s-a_1 & -a_2 & b_1 \\ \hline 0 & -c_1 & -c_2 & d \end{array}\right] \tag{3.20}$$

which is equivalent to its Smith form over $\mathbb{R}[s,z]$ as it is a single variable polynomial matrix.

Therefore $R(s,z)$ is equivalent to its Smith form over $\mathbb{R}[s,z]$ and so the result is true for $l = k+1$.

Therefore by mathematical induction on l, the result is true for all positive integers m and l.

Before continuing the induction process it is necessary to give a few general results. It is also worth noting that up to lemma 3.6 it has not been necessary to apply conditions on the Smith form of the Roesser matrix. But for

further progress some specilaization is required.

Lemma 3.7

If the m x n matrix P(s,z) is equivalent to its Smith form, $S_P(s,z)$, over $\mathbb{R}[s,z]$ where

$$S_P(s,z) = \left[\begin{array}{c|c} I_q & 0 \\ \hline 0 & 0 \end{array}\right] \text{ or } \left[\begin{array}{c|c} I_q & 0 \\ \hline 0 & \det(\, P(s,z)\,) \end{array}\right]$$

and if the matrix

$$R(s,z) = \left[\begin{array}{c} P(s,z) \\ \hline r(s,z) \end{array}\right]$$

where r(s,z) is a row vector and R(s,z) has Smith form, $S_R(s,z)$,

$$S_R(s,z) = \left[\begin{array}{c|c} I_{q+1} & 0 \\ \hline 0 & 0 \end{array}\right] \text{ or } \left[\begin{array}{c|c} I_q & 0 \\ \hline 0 & \det(\, R(s,z)\,) \end{array}\right]$$

then R(s,z) is equivalent to $S_R(s,z)$ over $\mathbb{R}[s,z]$ if it has no zeros.

proof

Consider the Smith forms $S_p(s,z)$ and $S_R(s,z)$ to be

$$S_p(s,z) = \left[\begin{array}{c|c} I_q & 0 \\ \hline 0 & 0 \end{array}\right] \tag{3.21}$$

$$S_R(s,z) = \left[\begin{array}{c|c} I_{q+1} & 0 \\ \hline 0 & 0 \end{array}\right] \tag{3.22}$$

Now as $P(s,z)$ is equivalent to $S_p(s,z)$ over $\mathbb{R}[s,z]$ then there exist unimodular matrices $M(s,z)$ and $N(s,z)$ over $R[s,z]$ such that

$$M(s,z)\ P(s,z)\ N(s,z) = S_p(s,z) \tag{3.23}$$

Therefore

$$\left[\begin{array}{c|c} M(s,z) & 0 \\ \hline 0 & 1 \end{array}\right] R(s,z)\ N(s,z) = \left[\begin{array}{c} S_p(s,z) \\ \hline r'(s,z) \end{array}\right] \tag{3.24}$$

where $R(s,z)$ has no zeros. Now

$$\left[\begin{array}{c} S_p(s,z) \\ \hline r'(s,z) \end{array}\right] = \left[\begin{array}{c|c} I_q & 0 \\ \hline 0 & 0 \\ \hline r"'(s,z) & r"(s,z) \end{array}\right]$$

$$\sim \left[\begin{array}{c|c} I_q & 0 \\ \hline 0 & r"(s,z) \\ \hline 0 & 0 \end{array}\right] \qquad (3.25)$$

as this has no zeros, then the elements of r"(s,z) must have no common zeros, and so there exist polynomials

$$a_1(s,z),\ldots,a_{n-q}(s,z)$$

such that

$$a_1(s,z)\ r"_1(s,z) + \ldots + a_{n-q}(s,z)\ r"_{n-q}(s,z) = 1 \qquad (3.26)$$

and thus further equivalence will transform R(s,z) into its Smith form $S_R(s,z)$.

Similar arguments can be applied for the other cases, and so prove the lemma.

Lemma 3.8

If the m x n matrix P(s,z) has Smith form $S_p(s,z)$,

$$S_p(s,z) = \left[\begin{array}{c|c} I_q & 0 \\ \hline 0 & 0 \end{array}\right] \text{ or } \left[\begin{array}{c|c} I_q & 0 \\ \hline 0 & \det(\ P(s,z)\ ) \end{array}\right]$$

and the (m+1) x n matrix R(s,z),

$$R(s,z) = \left[\begin{array}{c} P(s,z) \\ \hline r(s,z) \end{array}\right]$$

where r(s,z) is a row vector, and R(s,z) has Smith form $S_R(s,z)$,

$$S_R(s,z) = \left[\begin{array}{c|c} I_{q+1} & 0 \\ \hline 0 & 0 \end{array}\right] \quad \text{or} \quad \left[\begin{array}{c|c} I_q & 0 \\ \hline 0 & \det(\, R(s,z)\,) \end{array}\right]$$

and if R(s,z) has no zeros, then P(s,z) has no zeros.

<u>proof</u>: by contradiction.

Assume

$$S_P(s,z) = \left[\begin{array}{c|c} I_q & 0 \\ \hline 0 & 0 \end{array}\right] \tag{3.26}$$

$$S_R(s,z) = \left[\begin{array}{c|c} I_{q+1} & 0 \\ \hline 0 & 0 \end{array}\right] \tag{3.27}$$

R(s,z) has no zeros and P(s,z) has an ith order zero ($i \leq q$).

Consider the (i+1)th order minors of R(s,z), they can be expressed as linear combinations of the ith order minors

of P(s,z), which are all simultaneously zero for some (s,z). Therefore R(s,z) has an (i+1)th order zero, as the ith order determinantal divisor is unity.

A similar argument can be applied for the other cases and so prove the lemma.

Lemma 3.9

For all positive integers m, 1, and n the (n+m+1)x(n+1+1) matrix

$$P(s,z) = \begin{array}{c} \begin{matrix} n & 1 & 1 \end{matrix} \\ \left[\begin{array}{c|c|c} sI_n - a_1 & -a_2 & b_1 \\ \hline -a_3 & z - a_4 & b_2 \\ \hline -c_1 & -c_2 & d \end{array}\right] \end{array} \begin{matrix} n \\ 1 \\ m \end{matrix}$$

which has Smith form $S_p(s,z)$

$$S_p(s,z) = \left[\begin{array}{c|c} I_q & 0 \\ \hline 0 & 0 \end{array}\right] \text{ or } \left[\begin{array}{c|c} I_q & 0 \\ \hline 0 & \det(\, P(s,z)\,) \end{array}\right] \qquad (3.28)$$

is equivalent to its Smith form over R[s,z] if and only if it has no zeros.

proof: by induction on n.

By lemma 3.6 the result is true for $n = 1$.

Assume that the result is true for $n = k$, that is for all positive integers m and l the matrix

$$P(s,z) = \left[\begin{array}{c|c|c} sI_k - a_1 & -a_2 & b_1 \\ \hline -a_3 & z - a_4 & b_2 \\ \hline -c_1 & -c_2 & d \end{array}\right] \begin{array}{l} k \\ l \\ m \end{array} \qquad (3.29)$$

is equivalent to its Smith form over $\mathbb{R}[s,z]$ if and only if it has no zeros and has Smith form of the form (3.28).

Now consider the matrix

$$R(s,z) = \left[\begin{array}{c|c|c|c} sI_k - a_1 & -a_2 & b_1 & g_1 \\ \hline -a_3 & z - a_4 & b_2 & g_2 \\ \hline -c_1 & -c_2 & d & g_3 \\ \hline r_1 & r_2 & r_3 & s - a_{11} \end{array}\right] \begin{array}{l} k \\ l \\ m \\ l \end{array} \qquad (3.30)$$

where $g_1, r_1 \in \mathbb{R}^k$, $g_2, r_2 \in \mathbb{R}$, $g_3 \in \mathbb{R}^m$ and $r_3 \in \mathbb{R}^l$ and R(s,z) has no zeros, and has Smith form of the form (3.28).

As (3.29) holds for all l it can be seen that R(s,z) is of the form

$$R(s,z) = \left[\begin{array}{c} P'(s,z) \\ \hline r(s,z) \end{array}\right] \qquad (3.31)$$

and, from lemma 3.7, P'(s,z) has no zeros and so is equivalent to its Smith form, and so the result is true for n = k+1.

Therefore, by mathematical induction on n, the result is true for all positive integers m, l, and n.

It is now possible to conclude this section with the following result for the general (n+p+m)x(n+p+l) Roesser matrix with a particular Smith form.

Theorem 3.1

For all positive integers m, l, n, and p the (n+p+m)x(n+p+l) matrix

$$P(s,z) = \left[\begin{array}{c|c|c} sI_n - a_1 & -a_2 & b_1 \\ \hline -a_3 & zI_p - a_4 & b_2 \\ \hline -c_1 & -c_2 & d \end{array}\right]$$

which has Smith form $S_p(s,z)$

$$S_p(s,z) = \left[\begin{array}{c|c} I_q & 0 \\ \hline 0 & 0 \end{array}\right] \quad \text{or} \quad \left[\begin{array}{c|c} I_q & 0 \\ \hline 0 & \det(\,P(s,z)\,) \end{array}\right]$$

is equivalent to its Smith form over $\mathbb{R}[s,z]$ if and only if it has no zeros.

proof: by induction on p.

Similar to that for lemma 3.9.

4.4 Results

A number of Roesser type matrices were used to test the algorithms. It was certainly found that although the initial Roesser matrices seem simple in format, and have low degrees of the elements, the required transforming matrices were far from trivial. In some cases the degree of the elements and the magnitude of the coefficients were large. This is due to the fact that after the first iteration of the transformation the matrix is no longer in simple form.

Example 4.1

Consider the 4x5 Roesser matrix

$$\begin{bmatrix} s-5 & 1 & 4 & 1 & 2 \\ 2 & s-3 & 0 & 1 & 3 \\ 1 & 1 & z & 1 & 4 \\ 0 & 2 & 0 & z+5 & 1 \end{bmatrix}$$

It was not possible to directly find the Smith form over $\mathbb{R}[s,z]$.

The Smith form over $\mathbb{R}[z][s]$ was

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & z^4-5.455z^3-85.64z^2-37.09z+663.3 & 0 \end{bmatrix}$$

The Smith form over $\mathbb{R}[s][z]$ was

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & s^4-22.47s^2+198.5s^2-786.1s+1136 & 0 \end{bmatrix}$$

This gives the correct Smith form over $\mathbb{R}[s,z]$

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

Example 4.2

Consider the 6x5 Roesser matrix

$$\begin{bmatrix} 4 & 3 & 0 & 0 & 0 \\ 7 & 3 & 7 & 4 & 8 \\ 7 & 6 & 4 & 5 & 13 \\ 1 & 2 & s+2 & 6 & 0 \\ 2 & 1 & 8 & s-1 & 5 \\ 1 & 5 & 3 & 11 & z+7 \end{bmatrix}$$

It was not possible to directly find the Smith form over $\mathbb{R}[s,z]$.

The Smith form over $\mathbb{R}[z][s]$ was:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & z^2-52.93z+657.7 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The Smith form over $\mathbb{R}[s][z]$ was:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & s^2+0.9s-46.1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

This gives the correct Smith form over $\mathbb{R}[s,z]$

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

Example 4.3

Consider the 3x3 Roesser matrix

$$\begin{bmatrix} s-5 & 2 & 6 \\ 1 & z+2 & 2 \\ 0 & 5 & 1 \end{bmatrix}$$

It was possible to find the correct Smith form over $\mathbb{R}[s,z]$,

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & sz-8s-5z+68 \end{bmatrix}$$

The equivalence matrices over $\mathbb{R}[s,z]$ were

$$\begin{bmatrix} 1 & 0 & 0 \\ -2.5 & 0 & 1 \\ 0.0357\,1z-0.2857 & 1 & -0.2143z-0.2857 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 28 \\ 0.5 & 0.2143 & s-5 \\ 0 & -0.07143 & -5s+25 \end{bmatrix}$$

Example 4.4

Consider the 4x3 Roesser matrix

$$\begin{bmatrix} s-5 & 2 & 6 \\ 1 & z+2 & 2 \\ 0 & 5 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

It was possible to directly find the Smith form over $\mathbb{R}[s,z]$.

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

The equivalence matrices over $\mathbb{R}[s,z]$ were

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -2.5 & 0 & 1 & 0 \\ 0 & 1 & -0.2z-0.4 & 0.2z-1.6 \\ -0.1786 & 0.1786s-0.8929 & (-0.3571sz-0.07143s & (0.03571sz \\ & & +0.1786z+0.4286) & -0.2857s \\ & & & -0.1786z \\ & & & +2.429) \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 1 \\ 0.5 & 0.2143 & 0.03571s-0.1786 \\ 0 & -0.07143 & -0.1786s+0.8929 \end{bmatrix}$$

The results show that although the initial Roesser matrices are simple in format, the transforming matrices may be more complicated than expected.

Also the restriction on the Smith form for Lemma 3.9 and theorem 3.1 does not seem to be too strict, as the examples show that a number of the Smith forms are of the form

$$\left[\begin{array}{c|c} I & 0 \\ \hline 0 & 0 \end{array}\right] \quad \text{or} \quad \left[\begin{array}{c|c} I & 0 \\ \hline 0 & \det(\, R(s,z)\,) \end{array}\right]$$

where R(s,z) is the initial Roesser matrix.

CHAPTER 5

THE REALIZATION OF A TWO VARIABLE RATIONAL TRANSFER FUNCTION MATRIX

5.1 Introduction

In the introduction it was shown that a linear constant differential system of the form

$$\dot{x}(t) = A\,x(t) + B\,u(t) \tag{1.1}$$

$$y(t) = C\,x(t) + D\,u(t) \tag{1.2}$$

may be represented by the state-space system matrix

$$P(s) = \left[\begin{array}{c|c} sI - A & B \\ \hline -C & D \end{array}\right] \tag{1.3}$$

and has transfer function matrix

$$G(s) = C\,(sI - A)^{-1}\,B + D \tag{1.4}$$

However, if it is the transfer function matrix which is known then a system matrix of the form (1.3) is said to be a state-space realization of the rational transfer function

matrix G(s).

The realization is said to be minimal if the matrix A is of least dimension satisfying (1.4), and the dimension of A is called the order of the realization.

It can be shown (see, for example, Barnett 1971) that a necessary and sufficient condition for minimality is that the realization is controllable and observable. This is of course a very desirable system property.

In achieving the minimal state-space realization there are at present two main methods. Firstly there is that of Rosenbrock (1967, 1970) which has been implemented by Munro and McLeod (1971). Then there is the method of Pace and Barnett (1974b) which is shown to be the more efficient of the two methods. Both these methods are discussed in the next section.

In section 5.3 the method of Pace and Barnett is extended to realize a transfer function matrix over $\mathbb{R}[s,z]$. The property of minimality of the realization is discussed. Also it is not always possible to obtain a state-space realization over $\mathbb{R}[s,z]$, and the reasons for this are discussed.

The algorithm which has been developed is then implemented and tested with various transfer function

matrices and the results analysed.

5.2 The realization of a one variable transfer function matrix

For one variable transfer function matrices it is always possible to find a minimal, controllable and observable state-space realization. However the two main methods find the minimal realizations in different ways, and indeed may produce different realizations as the minimal realization is not unique.

5.2.1 The method of Rosenbrock (see, for example, Munro and McLeod 1971)

Given an m x 1 transfer function matrix G(s) which is such that

$$G(s) = G_p(s) + D(s) \qquad (2.1.1)$$

where D(s) is a polynomial matrix and $G_p(s)$ is proper.

Let $d_i(s)$ be the monic least common denominator of the ith row of $G_p(s)$ so that $G_p(s)$ can be expressed in terms of its rows.

$$G_p(s) = \begin{bmatrix} h_{1j}(s) / d_1(s) \\ \vdots \\ h_{mj}(s) / d_m(s) \end{bmatrix} \qquad (2.1.2)$$

for $j = 1,\ldots,l$.

The $d_i(s)$ and $h_{ij}(s)$ are

$$d_i(s) = s^{ri} + a_{ri-1}{}^i\, s^{ri-1} + \ldots + a_0{}^i \qquad (2.1.3)$$

$$h_{ij}(s) = h^{ij}{}_{ri-1}\, s^{ri-1} + \ldots + h^{ij}{}_0 \qquad (2.1.4)$$

Then a system matrix in state-space form giving rise to $G(s)$ is

$$P(s) = \left[\begin{array}{cccc:c} sI_{r1}-A_1 & 0 & \ldots & 0 & B_1 \\ 0 & sI_{r2}-A_2 & \ldots & 0 & B_2 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \ldots & sI_{rm}-A_m & B_m \\ \hdashline -C_1 & -C_2 & \ldots & -C_m & D(s) \end{array}\right] \qquad (2.1.5)$$

in which the A_i are companion matrices

$$A_i = \begin{bmatrix} 0 & 0 & \ldots & 0 & -a^i{}_0 \\ 1 & 0 & \ldots & 0 & -a^i{}_1 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & -a^i{}_{ri-1} \end{bmatrix} \qquad (2.1.6)$$

$$B_i = \begin{bmatrix} h^{il}{}_0 & h^{i2}{}_0 & \cdots & h^{il}{}_0 \\ h^{il}{}_1 & h^{i2}{}_1 & \cdots & h^{il}{}_1 \\ \vdots & \vdots & & \vdots \\ h^{il}{}_{ri-1} & h^{i2}{}_{ri-1} & \cdots & h^{il}{}_{ri-1} \end{bmatrix} \qquad (2.1.7)$$

$$C_i = \begin{bmatrix} 0 & \cdots & 0 & \\ \vdots & & \vdots & e_i \\ 0 & \cdots & 0 & \end{bmatrix} \qquad (2.1.8)$$

where e_i is the ith column of I_m.

This system is observable, but may have input-decoupling zeros and therefore may not be controllable. So the next stage is to remove the input-decoupling zeros, if there are any, whilst preserving the transfer function matrix. This will leave a minimal realization of the transfer function matrix G(s).

This method has been successfully implemented by Munro and McLeod but, as will be shown in the next section, it is not the most efficient method.

5.2.2 The method of Pace and Barnett (1974b)

The strategy of this method is to form an initial controllable (or observable) polynomial realization. Then the realization is made minimal, and finally expanded into state-space form.

Consider an m x n transfer function matrix G(s),

$$G(s) = G_p(s) + W(s) \qquad (2.2.1)$$

where $G_p(s)$ is proper and W(s) is a polynomial matrix. Now by expressing $G_p(s)$ in terms of its least common denominators, then

$$G(s) = V(s)\, T^{-1}(s)\, I_n + W(s)$$

where the matrix T(s) is diagonal and consists of the least common denominators of the columns of $G_p(s)$.

This gives the initial realization

$$P(s) = \left[\begin{array}{c|c} T(s) & I_n \\ \hline -V(s) & W(s) \end{array}\right] \qquad (2.2.2)$$

which is controllable as

$$[\ T(s) \ \vdots \ I_n\ ]$$

has full rank for all s.

The next stage is to make the polynomial realization observable. If

$$T(s) = T_1(s)\, D(s) \qquad (2.2.3)$$
$$V(s) = V_1(s)\, D(s)$$

that is they have a greatest common right divisor D(s) then

$$G(s) = V_1(s)\, T_1^{-1}(s)\, I_n + W(s) \qquad (2.2.4)$$

which implies that the system is observable since

$$\begin{bmatrix} T_1(s) \\ \hline -V_1(s) \end{bmatrix}$$

has full rank for all s.

Therefore

$$P(s) = \left[\begin{array}{c|c} T_1(s) & I_n \\ \hline -V_1(s) & W(s) \end{array}\right] \qquad (2.2.5)$$

is a minimal polynomial realization of G(s).

Finally by employing Rosenbrock's system matrix formulation (Rosenbrock 1970) as described in the previous

section, Pace and Barnett are able to reduce the polynomial minimal realization to a state-space minimal realization in a minimum of operations.

Pace and Barnett have given careful consideration to the form of the matrices T(s), V(s) after each operation, so that the method can be made as efficient as possible.

This method was successfully implemented by Pace and Barnett, and in comparison with the other realization algorithms, particularly that of Munro and McLeod (1971), was proved to be the most efficient.

5.3 The realization of a transfer function matrix over $\mathbb{R}[s,z]$

As the method of Pace and Barnett is the most efficient for the one variable realization problem, it was thought to be a good basis for the two variable realization problem. However, as has been shown in the previous chapters, the algebraic extension from $\mathbb{R}[s]$ to $\mathbb{R}[s,z]$ is not complete.

One of the main problems again is the difference between factors over $\mathbb{R}[s]$, and factors and zeros over $\mathbb{R}[s,z]$ (see, for example, Frost 1979). Rosenbrock (1970) gives a method for removing input (or output) decoupling zeros from a system matrix over $\mathbb{R}[s]$. Frost (1979) has shown that this method can be extended for the removal of input (or ouput) decoupling factors over $\mathbb{R}[s,z]$. Frost has also shown that it is not always possible to remove both input-decoupling and ouput-decoupling zeros. It is possible to remove all the input-decoupling zeros or all the ouput-decoupling zeros. But attemts to remove any further zeros may introduce different zeros of the opposite kind, that is trying to remove further input-decoupling zeros may introduce new output-decoupling zeros.

The implication of this is that the realization may not always be controllable and observable, however it will always be either controllable or observable. The concept of minimality will be preserved in the sense that when a

state-space realization is possible it can be made to be of least dimension.

It is not always possible to produce a state-space realization. If the least common denominator of all the elements of the transfer function matrix $G(s,z)$ is a polynomial which is monic over $\mathbb{R}[s,z]$ but not monic over $\mathbb{R}(z)[s]$, then if we consider a system matrix

$$P(s,z) = \left[\begin{array}{c|c} sI-A(z) & B(z) \\ \hline -C(z) & D(s,z) \end{array}\right] \qquad (3.1)$$

which has corresponding transfer function matrix

$$G'(s,z) = C(z)\ (\ sI-A(z)\ )^{-1}\ B(z) + D(s,z) \qquad (3.2)$$

we note that $\det(\ sI-A(z)\ )$ is a polynomial which is monic over $\mathbb{R}(z)[s]$, and so the least common denominator is monic over $\mathbb{R}(z)[s]$. Therefore it is not possible for $G(s,z)$ to have a state-space realization.

Each of the parts of the realization algorithm of Pace and Barnett are now examined further. Given the transfer function matrix $G(s,z)$, writing this as

$$G(s,z) = V(s,z)\ T^{-1}(s,z)\ I_n + W(s,z) \qquad (3.3)$$

will yield the controllable polynomial realization of G(s,z) over $\mathbb{R}[s,z]$

$$P(s,z) = \left[\begin{array}{c|c} T(s,z) & I_n \\ \hline -V(s,z) & W(s,z) \end{array}\right] \tag{3.4}$$

It is now necessary to find, and remove the greatest common right divisor of T(s,z) and V(s,z). To do this T(s,z) and V(s,z) are adjoined to form the matrix

$$A(s,z) = \left[\begin{array}{c} T(s,z) \\ \hline V(s,z) \end{array}\right] \tag{3.5}$$

Then A(s,z) can be transformed by elementary row operations only to the matrix

$$\left[\begin{array}{c} D(s,z) \\ \hline 0 \end{array}\right] \tag{3.6}$$

where D(s,z) is upper triangular and is the greatest common right divisor of T(s,z) and V(s,z). D(s,z) can be made to be unique by further transformation such that

$$\deg(d_{ij}(s,z)) < \deg(d_{ii}(s,z)) \quad ; \; j<i.$$

Thus

$$M(s,z)\left[\begin{array}{c} T(s,z) \\ \hline V(s,z) \end{array}\right] = \left[\begin{array}{c} D(s,z) \\ \hline 0 \end{array}\right] \tag{3.7}$$

or

$$\left[\begin{array}{c} T(s,z) \\ \hline V(s,z) \end{array}\right] = M^{-1}(s,z)\left[\begin{array}{c} D(s,z) \\ \hline 0 \end{array}\right] \tag{3.8}$$

which gives

$$\left[\begin{array}{c} T(s,z) \\ \hline V(s,z) \end{array}\right] = \left[\begin{array}{c|c} T_1(s,z) & X(s,z) \\ \hline V_1(s,z) & Y(s,z) \end{array}\right]\left[\begin{array}{c} D(s,z) \\ \hline 0 \end{array}\right] \tag{3.9}$$

with the obvious partitions. Therefore by consideration of the inverse operations required to find $D(s,z)$, the matrices $T_1(s,z)$ and $V_1(s,z)$ are found such that

$$T(s,z) = T_1(s,z)\, D(s,z)$$
$$V(s,z) = V_1(s,z)\, D(s,z)$$

that is, $T_1(s,z)$ and $V_1(s,z)$ are relatively right prime. However this does not imply observability, as although $T_1(s,z)$ and $V_1(s,z)$ have no common right divisor, the matrix

$$\left[\begin{array}{c} T_1(s,z) \\ \hline V_1(s,z) \end{array}\right]$$

may have zeros. That is the corresponding realization

$$\left[\begin{array}{c|c} T_1(s,z) & I_n \\ \hline -V_1(s,z) & W(s,z) \end{array}\right] \tag{3.10}$$

will have ouput-decoupling zeros, and as Frost has shown these may not be removable.

The greatest common divisor algorithm is very similar to the Smith form algorithm, and so has the same problems. The main problem is that it may not be possible to find the greatest common right divisor over $\mathbb{R}[s,z]$ but only over $\mathbb{R}(z)[s]$. However the transforming matrix may be renormalized in the following way. If

$$A(s,z) = M(s,z) \left[\begin{array}{c} D(s,z) \\ \hline 0 \end{array}\right] \tag{3.11}$$

where $M(s,z)$ is unimodular over $\mathbb{R}(z)[s]$ and $D(s,z)$ is upper triangular. By considering the least common denominator of the columns of $M(s,z)$ we have

$$A(s,z) = M(s,z)\ N(z)\ N^{-1}(z) \begin{bmatrix} D(s,z) \\ \hline 0 \end{bmatrix} \qquad (3.12)$$

and $M_1(s,z) = M(s,z)\ N(z)$ is now a matrix over $\mathbb{R}[s,z]$, not necessarily unimodular over $\mathbb{R}[s,z]$. If the least common denominator of the ith column of $M(s,z)$ is a factor of the ith row of $D(s,z)$, then $N^{-1}(z)\ D(s,z)$ is also a matrix over $\mathbb{R}[s,z]$ and the method has removed the greatest common right divisor of $T(s,z)$ and $V(s,z)$ over $\mathbb{R}[s,z]$.

The renormalization can be made possible if the equivalence transformation of $A(s,z)$ to $D(s,z)$ is over $\mathbb{R}[z][s]$ as shown in section 2.4 on the Smith form. This requires multiplying the rows of $A(s,z)$ by factors in $\mathbb{R}[z]$ to achieve the equivalence over $\mathbb{R}[z][s]$. These factors then become the denominators of the inverse equivalence matrix, as columns are divided by these factors.

In the Smith form algorithm to overcome the problem of not being able to find the equivalence over $\mathbb{R}[s,z]$, the Smith forms over $\mathbb{R}[z][s]$ and $\mathbb{R}[s][z]$ were found. However this technique cannot be applied to the greatest common right divisor algorithm. This is because the point of interest is not the greatest common right divisor but the equivalence matrix which gives the relatively right prime matrices $T_1(s,z)$ and $V_1(s,z)$. The equivalence matrices over $\mathbb{R}[z][s]$ and $\mathbb{R}[s][z]$ will be different, and although the

correct greatest common right divisor could be found there would be an added difficulty in calculating the correct relatively right prime matrices $T_1(s,z)$ and $V_1(s,z)$.

This now leaves, after removing the greatest common right divisor over $\mathbb{R}[s,z]$ ($\mathbb{R}[z][s]$), the controllable (but not necessarily observable) minimal polynomial realization

$$P(s,z) = \left[\begin{array}{c|c} T_1(s,z) & I_n \\ \hline -V_1(s,z) & W(s,z) \end{array}\right] \tag{3.13}$$

On the removal of the greatest common right divisor, $T_1(s,z)$ is upper triangular. Now if the diagonal elements of $T_1(s,z)$ are monic as polynomials over $\mathbb{R}(z)[s]$ then it is possible to find the state-space realization

$$R(s,z) = \left[\begin{array}{c|c} sI-A(z) & B(z) \\ \hline -C(z) & D(s,z) \end{array}\right] \tag{3.14}$$

Firstly elementary row operations, using the rows of $T_1(s,z)$, are applied to ensure that all the elements in the columns of the matrix

$$\left[\begin{array}{c} T_1(s,z) \\ \hline -V_1(s,z) \end{array}\right] \tag{3.15}$$

have degree in s lower than the degree in s of the corresponding diagonal element, that is

$$\deg_s(T_2(s,z)_{ij}) < \deg_s(T_2(s,z)_{jj}) \qquad i = 1,\ldots,j-1$$

$$\deg_s(-V_2(s,z)_{ij}) < \deg_s(T_2(s,z)_{jj}) \qquad i = 1,\ldots,m$$

where

$$P_1(s,z) = \left[\begin{array}{c|c} T_2(s,z) & B_1(s,z) \\ \hline -V_2(s,z) & W_1(s,z) \end{array}\right] \tag{3.16}$$

is the resulting realization.

By system equivalence, if there are any $T_2(s,z)_{ii}$ which are constant then row and column i of $P_1(s,z)$ can be deleted. Then all the other $T_2(s,z)_{jj}$, and the corresponding $T_2(s,z)_{ij}$ and $-V_2(s,z)_{ij}$, can be expanded into companion form blocks, see Pace and Barnett (1974b). This will result in the realization

$$R_1(s,z) = \left[\begin{array}{c|c} sI-A(z) & B_1(s,z) \\ \hline -C(z) & D_1(s,z) \end{array}\right] \tag{3.17}$$

Finally, using the diagonal elements of sI-A(z) as pivots, $B_1(s,z)$ can be transformed by equivalence to B(z),

giving the state-space realization

$$R(s,z) = \left[\begin{array}{c|c} sI-A(z) & B(z) \\ \hline -C(z) & D(s,z) \end{array}\right] \qquad (3.18)$$

such that

$$G(s,z) = C(z)\ (\ sI-A(z)\ )^{-1}\ B(z) + D(s,z), \qquad (3.19)$$

as required.

It can be seen that if an observable realization is required, then applying the above process on the transpose of the transfer function matrix will give a realization which when transposed will give the required observable realization.

5.4 Implementation of a two variable realization algorithm

Section 5.3 covered the development of an algorithm for the realization of a two variable transfer function matrix. This section is concerned with the implementation of this algorithm in a computer program written in Algol 68.

One of the major components of the algorithm is the greatest common right divisor algorithm. As already mentioned this is very similar to the Smith form algorithm, using Gaussian elimination on the rows only to transform the adjoined matrix

$$\begin{bmatrix} T(s,z) \\ \hline V(s,z) \end{bmatrix} \tag{4.1}$$

to the upper triangular form

$$\begin{bmatrix} D(s,z) \\ \hline 0 \end{bmatrix} \tag{4.2}$$

As with the Smith form algorithm, if equivalence over R[s,z] is not possible then the transformation will continue over R[z][s], which would involve multiplying rows of the matrix by polynomials in R[z] to allow the Gaussian elimination to be successful.

However for the greatest common right divisor algorithm

it is the inverse equivalence operations which are of importance. If we consider a series of elementary row operations on the matrix A(s,z) to transform it into the matrix B(s,z) then we have

$$R_k \ldots . R_1 \; A(s,z) = B(s,z) \tag{4.3}$$

where $R_i = R_i(s,z)$ which are unimodular over $\mathbb{R}[s,z]$ ($\mathbb{R}[z][s]$). Then

$$A(s,z) = I.R_1^{-1} \ldots . R_k^{-1} \; B(s,z) \tag{4.4}$$

which can be considered as a series of column operations, in the correct order, initially operating on the identity matrix. Therefore at the ith stage of the transformation we have

$$R_i R \; A(s,z) = A'(s,z) \tag{4.5}$$

or

$$A(s,z) = R^{-1} R_i^{-1} \; A'(s,z) \tag{4.6}$$

and so we must consider the ith inverse operation as a column operation on the matrix R. The elementary row operations over $\mathbb{R}[s,z]$ are:

1. Interchange rows i and j of A(s,z).
2. Add a multiple, p(s,z), of row i to row j of A(s,z).
3. Multiply row i of A(s,z) by a constant $k \in \mathbb{R} \neq 0$.

If equivalence is over $\mathbb{R}[z][s]$ then an extra operation is added:

4. Multiply row i of A(s,z) by a polynomial $p(z) \in \mathbb{R}[z] \not\equiv 0$

The corresponding inverse column operations over $\mathbb{R}[s,z]$ ($\mathbb{R}[z][s]$) are:

1. Interchange columns i and j of R(s,z).
2. Subtract a multiple, p(s,z), of column j from column i of R(s,z).
3. Divide column i of R(s,z) by a constant $k \in \mathbb{R} \neq 0$.
4. Divide column i of R(s,z) by a polynomial $p(z) \in \mathbb{R}[z] \not\equiv 0$.

To fully implement these inverse operations a numerator and denominator matrix representation is needed. The denominator matrix is only required if the transformation is over $\mathbb{R}[z][s]$ which means that the inverse equivalence matrix is over $\mathbb{R}(z)[s]$. This representation will make the calculation of the least common denominators easier. It is

important however to ensure that the corresponding numerator and denominator polynomials are relatively prime. This is necessary to ensure that no unnecessary factors are added to the least common denominator, as this would mean that the corresponding column has a common factor after renormalization.

The greatest common right divisor routine can now return the adjoined matrix

$$\begin{bmatrix} T_1(s,z) \\ \hline V_1(s,z) \end{bmatrix} \tag{4.7}$$

where $T_1(s,z)$ and $V_1(s,z)$ are the required relatively right prime matrices over $\mathbb{R}[s,z]$ (or $\mathbb{R}[z][s]$). It can be noted at this point that there is no real loss in having primeness over $\mathbb{R}[z][s]$ since a state-space realization would favour the s variable, and so the realization is over $\mathbb{R}[z][s]$. The inability to find the transformation over $\mathbb{R}[s,z]$ is related to the problem of zeros, as with the Smith form algorithm. In this case the zeros, if any, of the matrix

$$\begin{bmatrix} T(s,z) \\ \hline -V(s,z) \end{bmatrix} \tag{4.8}$$

are the output-decoupling zeros of the system, the presence

of which implies that the system is not observable.

The implementation of the realization algorithm itself now becomes easier with the algorithm being split into the various segments outlined in section 5.3, namely:

1. Finding the least common denominators of the columns of the transfer function matrix to give the initial realization

$$G(s,z) = V(s,z)\ T^{-1}(s,z)\ I_n + W(s,z) \qquad (4.9)$$

2. The removal of the greatest common right divisor of the matrices V(s,z) and T(s,z), leaving the relatively right prime matrices $T_1(s,z)$ and $V_1(s,z)$ and the minimal polynomial realization

$$G(s,z) = V_1(s,z)\ T_1^{-1}(s,z)\ I_n + W(s,z) \qquad (4.10)$$

3. When possible the expansion of the minimal polynomial realization into a minimal state-space realization,

$$G(s,z) = C(z)\ (\ sI-A(z)\ )^{-1}\ B(z) + D(s,z) \qquad (4.11)$$

The implementation of the third stage, using elementary

row and column operations and companion form expansion, follows exactly the steps outlined in section 5.3.

There are a number of general points about the implementation which are important. Throughout the algorithm it is necessary to find the least common denominator of a number of rational functions. This involves large use of the greatest common divisor algorithm of Bose (1976). As previously shown this algorithm is very costly in terms of computing time and stack usage. Therefore it is necessary to implement the greatest common divisor algorithm of Blankinship (1963) which finds the greatest common divisor of one variable polynomials. This can be implemented as a default method to the Bose algorithm whenever single variable polynomial greatest common divisors are required. This has particular significance in the renormalization of the equivalence matrix in the greatest common right divisor algorithm, as the denominators are polynomials in R[z].

Finally it is useful to check that the realization is correct, that is the polynomial or state-space realization found does correspond to the initial transfer function matrix. This obviously involves inverting a two variable polynomial matrix. As this is meant as a check, it is felt that the best way to evaluate the inverse matrix is to

calculate the determinant and the adjoint matrix.

5.5 Results

As already mentioned the algorithm is quite costly in time and stack usage. Also with the high overhead of checking the realization, testing the algorithm was restricted to "smaller" transfer function matrices.

Example 5.1

Consider the transfer function matrix, split into its proper and polynomial parts

$$\begin{bmatrix} z/s^2 & 1/(s+z) & 1/(z+1) \\ 1/s^2z & 1/(s-z) & s/(s^2+z^2) \end{bmatrix}$$

$$\begin{bmatrix} s+z & sz & 1 \\ s^2 & z^2 & s \end{bmatrix}$$

The correct polynomial realization was found

$$\left[\begin{array}{ccc:ccc} s^2z & 0 & 0 & 1 & 0 & 0 \\ 0 & s^2-z^2 & 0 & 0 & 1 & 0 \\ 0 & 0 & (s^2+z^2)(z+1) & 0 & 0 & 1 \\ \hdashline -z^2 & z-s & -(s^2+z^2) & s+z & sz & 1 \\ -1 & -(s+z) & -s(z+1) & s^2 & z^2 & s \end{array}\right]$$

However it was not possible to find a state-space realization.

Example 5.2

Consider the transfer function matrix, which is proper

$$\begin{bmatrix} 1/(s+z) & 0 & s/(s^2-z^2) \\ 0 & (s+z+1)/(s^2+z^2+2) & 0 \\ 3/(s-z) & 1/(s+1) & z/(s^2+z^2) \end{bmatrix}$$

The correct polynomial realization was found,

$$\left[\begin{array}{ccc:ccc} s^2-z^2 & 0 & z^2(z-s) & 1 & 0 & 0 \\ 0 & (s+1)(s^2+z^2+2) & 0 & 0 & 1 & 0 \\ 0 & 0 & (s^2+z^2)(s-z) & 0 & 0 & 1 \\ \hdashline z-s & 0 & sz-(s^2+z^2) & 0 & 0 & 0 \\ 0 & -(s+1)(s+z+1) & 0 & 0 & 0 & 0 \\ -3(s+z) & -(s^2+z^2+2) & z(4z-s) & 0 & 0 & 0 \end{array}\right]$$

Also it was possible to find the correct state-space realization,

$$\left[\begin{array}{cccccccc:ccc}
s & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-z^2 & s & 0 & 0 & 0 & z^3 & -z^2 & 0 & 1 & 0 & 0 \\
0 & 0 & s & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & s & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & z^2+2 & z^2+2 & s+1 & 0 & -1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & s & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & s & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -z^3 & z^2 & s-z & 0 & 0 & 1 \\
\hdashline
z & -1 & 0 & 0 & 0 & -z^2 & z & -1 & 0 & 0 & 0 \\
0 & 0 & -(z+1) & -(z+2) & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-3z & -3 & -(z+2) & 0 & -1 & 4z^2 & -z & 0 & 0 & 0 & 0
\end{array}\right]$$

The order of the state-space realization is 8.

Example 5.3

Consider the transfer function matrix, which although it is not proper is not split into proper and polynomial parts,

$$\begin{bmatrix} 1/(s-z) & s+2 \\ z+1 & (s+z+1)/(s^2+z^2+2) \\ 3/(s+z) & 1/(s+1) \end{bmatrix}$$

The correct polynomial realization was found,

$$\left[\begin{array}{cc:cc} s^2-z^2 & 0 & 1 & 0 \\ 0 & (s+1)(s^2+z^2+2) & 0 & 1 \\ \hdashline -(s+z) & 0 & 0 & s+2 \\ 0 & -(s+1)(s+z+1) & z+1 & 0 \\ 3(z-s) & -(s^2+z^2+2) & .0 & 0 \end{array}\right]$$

Also it was possible to find the correct state-space realization,

$$\left[\begin{array}{ccccc:cc} s & -1 & 0 & 0 & 0 & 0 & 0 \\ -z^2 & s & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & s & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & s & -1 & 0 & 0 \\ 0 & 0 & z^2+2 & z^2+2 & s+1 & 0 & 1 \\ \hdashline -z & -1 & 0 & 0 & 0 & 0 & s+2 \\ 0 & 0 & -(z+1) & -(z+2) & -1 & z+1 & 0 \\ 3z & -3 & -(z^2+2) & 0 & -1 & 0 & 0 \end{array}\right]$$

The order of the state-space realization is 5.

The algorithm successfully finds the polynomial and, when possible, the state-space realizations. An added result is that the algorithm will successfully split a transfer function matrix into its proper and polynomial parts, as shown by example 5.3.

CHAPTER 6

CONCLUSIONS AND SUGGESTIONS FOR FURTHER WORK

The thesis has extensively investigated the algebraic extensions to the ring R[s,z] particularly for the 2x2 two variable polynomial matrix. A number of different approaches have been given to try to resolve the problem of equivalence of a general two variable polynomial matrix with its Smith form over R[s,z]. These approaches have been a direct analysis of the equivalence transformation such that, if possible, a constructive method for attaining the equivalence may be obtained. However this direct approach was not totally successful. A solution to this problem may be found by a deep investigation into abstract algebra and the theory of rings and domains. This may supply necessary and sufficient conditions for the equivalence over R[s,z] to exist.

As it was not possible to find the required equivalence conditions for a general polynomial matrix over R[s,z], more specialized matrices were considered. Firstly the extension of the result of Lee and Zak (1981) found that the concepts of a cyclic matrix and a non-derogatory matrix over R[s,z] are not equivalent. This is a fundamental difference, and again the actual reasons for the difference

between the two concepts over R[s,z] may be found in the theory of abstract algebra. However the result of Lee and Zak does not provide a better constructive method than the one developed, as it is as difficult to find a cyclic vector.

The second specialization was to consider the Roesser matrix. The approach was to directly consider the equivalence transformation, and to use mathematical induction to extend the result for a 3x3 Roesser matrix to the general (n+p+m) x (n+p+1) Roesser matrix. Unfortunately it was not possible to continue the direct algebraic approach in the last stages of the induction without making a special condition on the Smith form of the Roesser matrix. The results show that this specialization is not too strict.

The problems encountered in the algebraic investigations were highlighted in the algorithmic development. The algorithms were designed to overcome the problems of the algebraic extensions. The matrices chosen proved a good test of the algorithms accuracy and reliability. Checks within the algorithms proved that the equivalence matrices found, correctly transformed the initial matrix into the particular Smith form.

The Smith-McMillan form and realization algorithms were also shown to be accurate by using checks within the algorithms.

There is however the problem of running time for the algorithms. An immediate problem is that the number of multiplications to multiply two polynomials over $\mathbb{R}[s,z]$ is the square of the multiplications to multiply two polynomials over $\mathbb{R}[s]$. Also it is necessary to use algorithms such as that of Bose (1976) for calculating the greatest common divisor of two multivariable polynomials. This is far more time consuming than the method of Blankinship (1963). Added to this is the problem of not being able to achieve the equivalence immediately.

There could be no real improvement to the algorithms in this respect without major changes in the strategy of all the algorithms, unless a new constructive method could be found by resolving the algebraic problems.

CHAPTER 7

REFERENCES

Barnett S., 1971, Matrices in Control Theory.
Van-Nostrand Reinhold.

Blankinship W.A., 1963, "A new version of the Euclidean Algorithm."
American Math. Monthly, 70, 742-745

Bose N.K., 1976, "An algorithm for GCF extraction from two multivariable polynomials."
Proc I.E.E.E., 64, 1, 185-186

Fornasini E., & Marchesini G., 1975, Variable Structure Systems.
Springer-Verlag, pp64-82

Frost M.G., 1979, Polynomial Matrices associated with Linear Constant Multivariable Delay-Differential Systems.
Ph.D Thesis, Loughborough University of Technology

Frost M.G., & Storey C., 1978, "Equivalence of a matrix over R[s,z] with its Smith form."
I.J.C., 28, 5, 665-671

Frost M.G., 1981, Private communication to E.B. Lee.

Fuhrmann P.A., 1977, "On strict system equivalence and similarity."
I.J.C., 25, 5

Givone D.D., & Roesser R.P., 1972, "Multidimensional linear iterative circuits - general properties."
I.E.E.E. Trans. Comp., C-21, 1067-1073

Givone D.D., & Roesser R.P., 1973, "Minimization of multidimensional linear iterative circuits."
I.E.E.E. Trans. Comp., C-22, 7, 673-678

Kung S-Y., Levy B.C., Morf M., & Kailath T., 1977, "New results in 2-D systems theory, part II: 2-D state-space models - realization and the notions of controllabilty, observability and minimality."
Proc. I.E.E.E., 65, 6, 945-961

Lee E.B., & Zak S.H., 1981, "Smith forms over $\mathbb{R}[z_1,...z_q]$ and their applications."
University of Minnesota report.

Morf M., Levy B.C., & Kung S-Y., 1977, "New results in 2-D systems theory, part I: 2-D polynomial matrices, factorization and coprimeness."
Proc. I.E.E.E., 65, 6, 861-872

Munro N., & McLeod R.S., 1971, "Minimal realization of transfer function matrices using the system matrix."
Proc. I.E.E., 118, 9, 1298-1301

Pace I.S., & Barnett S., 1974a, "Efficient algorithms for linear systems calculation part I: Smith form and common divisor of polynomial matrices."
I.J. Sys. Sci., 5, 5, 413-424

Pugh A.C., & Shelton A.K., 1978, "On a new definition of strict system equivalence."
I.J.C., 27, 5, 657-672

Rosenbrock H.H., 1967, "On linear system theory."
Proc. I.E.E., 114, 1353-1359

Rosenbrock H.H., 1970, State-Space and Multivariable Theory
Nelson

Rosenbrock H.H., & Storey C., 1970,
Mathematics of Dynamical Systems.
Nelson

Van Der Waerden B.L., 1964, Modern Algebra.
Frederick Ungar

Zakian V., & Williams N.S., 1973, "Canonical state equations of delay-differential systems."
Cont. Syst. Centre report 213., U.M.I.S.T.

APPENDIX A

PROGRAM LISTINGS:

POLYNOMIAL OPERATORS OVER R[s,z] OR R[z][s]

```
!BEGIN!

!MODE!!POLY!=!STRUCT!([0:0!FLEX!,0:0!FLEX!]!REAL!P);
!MODE!!DEG!=[1:6]!INT!;
!POLY!DUMMY;
!DEG!D1;
!BOOL!RAT←!FALSE!;


!PROC!DF=(!REF!!DEG!D,!POLY!A):
!C!   FINDS THE DEGREE OF A POLYNOMIAL   !C!
!BEGIN!
!INT!D1←1!UPB!P!OF!A,D2←2!UPB!P!OF!A,I,J,LT←0;
!BOOL! RUN←!TRUE!;
!FOR!R!FROM!D1+D2!BY!-1!TO!0!WHILE!RUN!DO!
     !BEGIN!
     I←(R>=D1|D1[R):J←R-I;
     !WHILE!J<=D2!AND!I>=0!AND!RUN!DO!
          (!ABS!(P!OF!A)[I,J]>1.0&-6|RUN←!FALSE!|J!PLUS!1;I!MINUS!1)
     !END!;
RUN←!TRUE!;
!FOR!L!FROM!D2!BY!-1!TO!0!WHILE!RUN!DO!
     (!ABS!(P!OF!A)[D1,L]>1.0&-6|RUN←!FALSE!;LT←L);
!C!
   D[1]= UPPER BOUND OF THE FIRST VARIABLE
   D[2]= UPPER BOUND OF THE SECOND VARIABLE
   D[3],D[4]= POSITION OF THE LEADING COEFFICIENT
   D[5]= POSITION OF THE LEADING COEFFICIENT IF RATIONAL
   D[6]= DEGREE OVER R[S,Z] OR R[Z][S]
!C!
D←(D1,D2,I,J,LT,(RAT|D1|I←J))
!END!;


!OP!£=(!POLY!A)!BOOL!:
!C!   CHECKS IF A POLYNOMIAL IS ZERO   !C!
!BEGIN!
!DEG!D;
DF(D,A);
(D[1]=0!AND!D[2]=0!AND!!ABS!(P!OF!A)[0,0]<1.0&-6)
!END!;


!OP!%=(!POLY!A)!POLY!:
!C!   TRIMS THE POLYNOMIAL A   !C!
!BEGIN!
!POLY!B;
!DEG!D;DF(D,A);
!BOOL!COL←!TRUE!,ROW←!TRUE!;
!WHILE!COL!DO!
     !BEGIN!
     !FOR!I!FROM!0!TO!D[1]!DO!
          (!ABS!(P!OF!A)[I,D[2]]>1.0&-6|COL←!FALSE!);
     (D[2]=0|COL←!FALSE!);
     (COL|D[2]!MINUS!1)
     !END!;
!C!   TRIMMED THE COLUMNS   !C!
!WHILE!ROW!DO!
     !BEGIN!
     !FOR!J!FROM!0!TO!D[2]!DO!
          (!ABS!(P!OF!A)[D[1],J]>1.0&-6|ROW←!FALSE!);
```

```
      (D[1]=0|POW←'FALSE');
      (ROW[1]'PLUS'1)
      'END';
!C!   TRIMMED THE ROWS   !C!
P!OF!A←(P!OF!A)[0:D[1]!AT!0,0:D[2]!AT!0];
B
!END!;


!OP!'CLEAR'=('REF'!POLY'A):('CLEAR'P!OF!A);


!PROC!REDIM=('REF'!POLY'A,'INT'D1,D2);
!C!
   RE-DIMENSIONS A POLYNOMIAL TO ACCEPT MORE COEFFICIENTS
!C!
!BEGIN!
!DEG!D;
DF(D,A);
!POLY'B;
P!OF!B←[0:D1,0:D2]'REAL';
!CLEAR!B;
(P!OF!B)[0:D[1],0:D[2]]←(P!OF!A);
A←B
!END!;


!OP!'/=('REF'!POLY'A,'POLY'B)!POLY';
!C!
   OPERATOR FOR THE DIVISION OF TWO POLYNOMIALS   A/B   DELIVERING A
   QUOTIENT POLYNOMIAL AND ALTERRING POLYNOMIAL A TO BE THE REMAINDER
!C!
!BEGIN!
!INT!DEG,I,J,K,L;
!REAL!MC;
!DEG!DA,DB,DQ;
DF(DA,A);DF(DB,B);
!BOOL!RUN←'TRUE';
!POLY!Q;
!IF!      £A
      'OR' £B
      'OR' DA[1]<DB[1]
      'OR' DA[6]<DB[6]
      'OR' (DA[1]+DA[5]<DB[1]+DB[5]!AND!RAT)
!THEN!
      'C'   NO DIVISION REQUIRED  SO SET  Q=0  R=A   'C'
      'CLEAR'Q
!ELSE!
      DEG←DA[2]-DB[2];
      (DEG<0|DEG←0);
      DQ←(DA[1]-DB[1],DEG,0,0,0,0);
      P!OF!Q←[0:DQ[1],0:DQ[2]]'REAL';
      'CLEAR'Q;
      (RAT|H←DB[1];L←DB[5]|H←DB[3];L←DB[4]);
      'WHILE'RUN!DO'
            'BEGIN'
            'C'   MAIN LOOP TO CARRY OUT LONG DIVISION   'C'
            A←%A;
            DF(DA,A);DF(DQ,Q);
            (RAT|I←DA[1];J←DA[5]|I←DA[3];J←DA[4]);
            'IF'      £A
                'OR' I<H
```

```
            'OR' J<L
            'OR' I+J<H+L
        'THEN'
            RUN←'FALSE'
        'ELSE'
            MC←(P'OF'A)[I,J]/(P'OF'B)[H;L];
            (DA[2]-J<DB[2]-L'REDIM'(A,DA[1];DB[2]+J-L);
                                REDIM'(Q,DQ[1];DQ[2]+DB[2]-DA[2]+J-L));
            (DA[1]-I<DB[1]-H'REDIM'(A,DB[1]+I-H,DB[2]);
                                REDIM'(Q,DQ[1]+DB[1]-DA[1]+I-H,DQ[2]));
            'FOR'II'FROM'0'TO'DB[1]'DO'
                'FOR'JJ'FROM'0'TO'DB[2]'DO'
                    (P'OF'A)[II+I-H,JJ+J-L]'MINUS'
                    MC*(P'OF'B)[II,JJ];
                'C'   SUBTRACTS MULTIPLE OF  B  FROM  A   'C'
            (P'OF'Q)[I-H,J-L]←MC
            'C'   FORMS THE QUOTIENT POLYNOMIAL   'C'
        'FI'
        'END'
'FI';
A←%A;
%Q
'END';


'OP'*←('POLY'A,B)'POLY';
'C'   MULTIPLIES TWO POLYNOMIALS   AND   'C'
'BEGIN'
'DEG'DA,DB;
DF(DA,A);DF(DB,B);
'POLY'PROD;
'IF' £A'OR'£B
'THEN'
    'C'   PRODUCT IS ZERO   'C'
    'CLEAR'PROD
'ELSE'
    P'OF'PROD←[0:DA[1]+DB[1];0:DA[2]+DB[2]]'REAL';
    'CLEAR'PROD;
    'FOR'I'FROM'0'TO'DA[1]'DO'
        'FOR'J'FROM'0'TO'DA[2]'DO'
            'FOR'H'FROM'0'TO'DB[1]'DO'
                'FOR'L'FROM'0'TO'DB[2]'DO'
                    (P'OF'PROD)[I+H,J+L]'PLUS'
                    (P'OF'A)[I,J]*(P'OF'B)[H,L]
'FI';
%PROD
'END';


'OP'*←('POLY'A;'REAL'MC)'POLY';
'C'   MULTIPLIES A POLYNOMIAL  A  BY A CONSTANT  MC   'C'
'BEGIN'
'POLY'B←A;
'DEG'D;DF(D,A);
'FOR'I'FROM'0'TO'D[1]'DO'
    'FOR'J'FROM'0'TO'D[2]'DO'          (P'OF'B)[I,J]'TIMES'MC;
%B
'END';


'OP'+←('POLY'A,B)'POLY';
'C'   ADDS TWO POLYNOMIALS   A+B   'C'
```

```
!BEGIN!
!DEG!DA,DB;
DF(DA,A);DF(DB,B);
!INT!DEG1,DEG2;
DEG1←(DA[1]>DB[1]!DA[1]!DB[1]);
DEG2←(DA[2]>DB[2]!DA[2]!DB[2]);
!POLY!SUM;
P!OF!SUM←[0:DEG1,0:DEG2]!REAL';
!CLEAR!SUM;
!FOR!I!FROM!0!TO!DA[1]!DO!
     !FOR!J!FROM!0!TO!DA[2]!DO!
          (P!OF!SUM)[I,J]←(P!OF!A)[I,J];
!FOR!I!FROM!0!TO!DB[1]!DO!
     !FOR!J!FROM!0!TO!DB[2]!DO!
          (P!OF!SUM)[I,J]!PLUS!(P!OF!B)[I,J];
%SUM
!END!;


!OP!-=(!POLY!A)!POLY!;
!C!  MONADIC MINUS    !C!
!BEGIN!
!POLY!B;
!DEG!D;
DF(D,A);
P!OF!B←[0:D[1],0:D[2]]!REAL';
!FOR!I!FROM!0!TO!D[1]!DO!
     !FOR!J!FROM!0!TO!D[2]!DO!    (P!OF!B)[I,J]←-(P!OF!A)[I,J];
%B
!END!;


!OP!-=(!POLY!A,B)!POLY!;
!C!   SUBTRACTS TWO POLYNOMIALS    A-B    !C!
!BEGIN!
!DEG!DA,DB;
DF(DA,A);DF(DB,B);
!INT!DEG1,DEG2;
DEG1←(DA[1]>DB[1]!DA[1]!DB[1]);
DEG2←(DA[2]>DB[2]!DA[2]!DB[2]);
!POLY!DIFF;
P!OF!DIFF←[0:DEG1,0:DEG2]!REAL';
!CLEAR!DIFF;
!FOR!I!FROM!0!TO!DA[1]!DO!
     !FOR!J!FROM!0!TO!DA[2]!DO!
          (P!OF!DIFF)[I,J]←(P!OF!A)[I,J];
!FOR!I!FROM!0!TO!DB[1]!DO!
     !FOR!J!FROM!0!TO!DB[2]!DO!
          (P!OF!DIFF)[I,J]!MINUS!(P!OF!B)[I,J] ;
%DIFF
!END!;


!OP!!PLUS!=(!REF!!POLY!A,!POLY!B);(A←A+B);


!OP!!MINUS!=(!REF!!POLY!A,!POLY!B);(A←A-B);


!OP!!TIMES!=(!REF!!POLY!A,!POLY!B);(A←A*B);
```

```
!OP!'TIMES'=('REF'POLY'A,'REAL'RC):(A←AWRC);


!OP!'CLEAR'=('REF'[,]'POLY'PM);
!C!   CLEARS A POLYNOMIAL MATRIX   'C'
!BEGIN'
!INT'M←1'UPB'PM,N←2'UPB'PM;
!POLY'ZEROPOLY;
!CLEAR'ZEROPOLY;
!FOR'I'TO'M'DO'
     'FOR'J'TO'N'DO'
          PM[I,J]←ZEROPOLY
!END';



!OP!*=([,]'POLY'K1,K2)[,]'POLY';
!C!   MULTIPLIES TWO POLYNOMIAL MATRICES   'C'
!BEGIN'
!INT'M1←1'UPB'K1,M1←2'UPB'K1,M2←1'UPB'K2,N2←2'UPB'K2;
[1:M1,1:N2]'POLY'PROD;'CLEAR'PROD;
!IF'N1=M2
!THEN'
     'FOR'I'TO'M1'DO'
          'FOR'J'TO'N2'DO'
               'FOR'K'TO'N1'DO'
                    PROD[I,J]'PLUS'(K1[I,K]*K2[K,J])
!FI';
PROD
!END';


!PROC'READPOLYMX=('REF'[,]'POLY'K);
!C!   INPUTS A POLYNOMIAL MATRIX   'C'
!BEGIN'
!INT'M←1'UPB'K,N←2'UPB'K;
!FOR'I'TO'M'DO'
     'FOR'J'TO'N'DO'
          'BEGIN'
          'INT'DEG1,DEG2;
          READ((DEG1,DEG2));
          POLY'K[I,J]←[0:DEG1,0:DEG2]'REAL';
          READ(POLY'K[I,J]);
          K[I,J]←%K[I,J]
          'C'
             ENSURES THAT THE INPUTTED POLYNOMIAL IS OF LOWEST FORM
          'C'
          'END'
!END';


!PROC'NL=('INT'N):('FOR'I'TO'N'DO'NEWLINE(STAND OUT));


!PROC'SP=('INT'N):('FOR'I'TO'N'DO'SPACE(STAND OUT));


!PROC'PRC=('CHAR'C,'INT'N):('FOR'I'TO'N'DO'PRINT(C);NL(1));


!PROC'PRS=('STRING'S):(PRINT(S);NL(1));
```

```
!PROC!PRSI=('STRING'S,'INT'I):(PRINT(S);OUTF(STAND OUT,S=2VLS,I))!


!PROC!PRINTPOLY=('POLY'A);
!C!   OUTPUTS A POLYNOMIAL   !C!
!BEGIN!
!INT!II,JJ;
!DEG!D;DF(D,A);
!BOOL!RUN←'TRUE';
II←0;JJ←4;
!WHILE!RUN!DO!
     !BEGIN!
     (JJ>=D[2]|JJ←D[2];RUN←'FALSE');
     OUTF(STAND OUT,SN(D[1]+1)(LN(JJ-II+1)(-D.3DXE=2UVX))S,
     (P!OF!A)[,II:JJ]);
     (RUN|II!PLUS!5;JJ!PLUS!5;NL(2))
     !END!;
NL(1)
!END!;

!PROC!PRINTPOLYMX=([,]!POLY!K);
!C!   OUTPUTS A POLYNOMIAL MATRIX   !C!
!BEGIN!
!INT!M←1!UPB!K,N←2!UPB!K;
!FOR!I!TO!M!DO!
     !FOR!J!TO!N!DO!
          !BEGIN!
          !IF!NOT!EK[I,J]
          !THEN!
               OUTF(STAND OUT,SL"["2SV",",2SV"]"S,(I,J));
               PRINTPOLY(K[I,J])
          !FI!
          !END!;
NL(1);PRC("*",60);NL(1)
!END!;


!PROC!MONIC=(!REF![,]!POLY!K,PREN,!INT!R);
!C!
   DIVIDES THE PIVOTAL ROW TO MAKE THE PIVOT MONIC OVER R[S,Z]
   OR R[Z][S]
!C!
!BEGIN!
!INT!M←1!UPB!K,N←2!UPB!K;
!REAL!CONST;
!BOOL!RUN←'TRUE';
!DEG!D;
DF(D,K[R,R]);
CONST←(RAT((P!OF!K[R,R])[D[1],D[5]]|(P!OF!K[R,R])[D[3],D[4]]);
!FOR!I!TO!M!DO!
     PREN[I,R]!TIMES!CONST;
!FOR!J!FROM!R!TO!N!DO!
     K[R,J]!TIMES!(1.0/CONST)
!C!
   DIVIDES ROW R TO MAKE PIVOT MONIC
!C!
!END!;


!PROC!LSTCOLDEG=(!REF![,]!POLY!K,PREN,PRED,!INT!R,!REF!!BOOL!ZERO,
                 CHANGE);
```

```
'C'
   FINDS THE POLYNOMIAL OF LEAST DEGREE IN COLUMN  R
   AND MOVES IT TO POSITION  (R,R)
'C'
'BEGIN'
'INT'K←1'UPB'K;N←2'UPB'K,II←R,DG,DR;
'DEG'D;
ZERO←'TRUE';CHANGE←'FALSE';
'FOR'I'FROM'R'TO'N'DO'
     'BEGIN'
     'POLY'A←K[I,R];
     DF(D,A);
     'IF'    'NOT'ΣA
     'THEF'  ZERO
     'THEN'          DG←D[6];DR←(RAT|D[5]|D[1]);
                     II←I;ZERO←'FALSE'
     'ELSF'  D[6]<DG
     'THEN'          DG←D[6];DR←(RAT|D[5]|D[1]);II←I
     'ELSF'  D[6]=DG 'AND' ((D[5]<DR'AND'RAT)'OR'(D[1]<DR'AND''NOT'RAT)
     'THEN'          DG←D[6];DR←(RAT|D[5]|D[1]);II←I
     'FI'
     'END';
'C'
   FIRSTLY SEARCHES COLUMN  R  FOR THE FIRST NON-ZERO ELEMENT
   THEN CONTINUES FROM THERE TO SEARCH FOR THE NON-ZERO ELEMENT
   OF LEAST DEGREE AT POSITION  (II,R)
   IF THERE ARE NO NON-ZERO ELEMENTS THE  BOOL  ZERO  WILL REMAIN TRUE
'C'
'IF' II≠R
'THEN'
     [1:N]'POLY'A←K[II,];K[II,]←K[R,];K[R,]←A;
     [1:M]'POLY'B←PREN[,II];PREN[,II]←PREN[,R];PREN[,R]←B;
     B←PRED[,II];PRED[,II]←PRED[,R];PRED[,R]←B;
     CHANGE←'TRUE'
'FI'
'C'   INTERCHANGES ROWS II,R   'C'
'END';


'PROC'DET=([,]'POLY'K)'POLY';
'C'
   FINDS THE DETERMINANT OF A POLYNOMIAL MATRIX BY EXPANSION ALONG THE
   FIRST ROW AND RECURSION FOR LOWER ORDER DETERMINANTS
'C'
'BEGIN'
'INT'N←'UPB'K;
'POLY'DTRM;
'CLEAR'DTRM;
'REAL'SIGN←1.0;
'IF'        N>2
'THEN'
      'FOR'I'TO'N'DO'
            'BEGIN'
            [1:N-1,1:N-1]'POLY'SL;
            'IF'          I=1
            'THEN'              SL←K[2:N,2:N]
            'ELSF'        I=N
            'THEN'              SL←K[2:N,1:N-1]
            'ELSE'              SL[,1:I-1]←K[2:N,1:I-1];
                                SL[,I:N-1]←K[2:N,I+1:N]
            'FI';
            ('NOT'ΣK[1,I]|DTRM'PLUS'(K[1,I]*DET(SL)*SIGN));
```

```
            SIGN←-SIGN
            'END'
'ELSF'      N←1
'THEN'      DTRM←K[1,1]
'ELSF'      DTRM←K[1,1]*K[2,2]-K[1,2]*K[2,1]
'FI';
%DTRM
'END';


'PROC'INNER=('REF'[,]'POLY'K);
'C'
   FINDS THE INNER SQUARE OF A MATRIX
'C'
'BEGIN'
'INT'N←'UPB'K;
[1:N-2,1:N-2]'POLY'INN←K[2:N-1,2:N-1];
K←INN
'END';


'PROC'SUBRES=('REF'[,]'POLY'K,'POLY'A,B);
'C'
   FINDS THE SUBRESULTANT OF THE DIGRADIENT MATRIX USED IN FINDING THE
   GREATEST COMMON DIVISOR
'C'
'BEGIN'
'DEG'DA,DB:DF(DA,A);DF(DB,B);
'INT'N←'UPB'K,NN←(DA[1]+DB[1]-N)'/'2;
[1:N,1:N]'POLY'SBRS←K;
'POLY'S;
P'OF'S←[0:1,0:0]'REAL';
(P'OF'S)[,0]←(0.0,1.0);
'IF' (DB[1]-NN)>0
'THEN'
      SBRS[DB[1]-NN,N]←A;
      ((DB[1]-NN)>1|'FOR'I'FROM'DB[1]-NN-1'BY'-1'TO'1'DO'
                  SBRS[I,N]←S*SBRS[I+1,N])
'FI';
'IF' (DA[1]-NN)>0
'THEN'
      SBRS[DB[1]-NN+1,N]←B;
      ((DA[1]-NN)>1|'FOR'I'FROM'N-DA[1]+NN+2'TO'N'DO'
                  SBRS[I,N]←S*SBRS[I-1,N])
'FI';
K←SBRS
'END';


'PROC'VAR=('REF''POLY'A);
'C'
   CHANGES THE VARIABLES OF A POLYNOMIAL AROUND  S TO Z & Z TO S
'C'
'BEGIN'
'DEG'D:DF(D,A);
'POLY'B;
P'OF'B←[0:D[2],0:D[1]]'REAL';
'FOR'L'FROM'0'TO'D[1]'DO'
     (P'OF'B)[,L]←(P'OF'A)[L,];
A←B
'END';
```

```
!PROC!VARCH=(!REF![,]!POLY!K):
!C!
   CHANGES THE VARIABLES OF A POLYNOMIAL MATRIX
!C!
!BEGIN!
!INT!N←1!UPB!K,N←2!UPB!K;
!FOR!I!TO!M!DO!
     !FOR!J!TO!N!DO!
          VAR(K[I,J])
!END!;


!PROC!GCD1=(!POLY!A1,B1)!POLY!:
!C!
   FINDS THE GREATEST COMMON DIVISOR OF TWO POLYNOMIALS IN ONE
   VARIABLE USING BLANKINSHIPS ALGORITHM
!C!
!BEGIN!
!POLY!A←A1,B←B1,Q;
!DEG!DA,DB;
!WHILE! !NOT!EA !AND! !NOT!EB !DO!
     !BEGIN!
     DF(DA,A);DF(DB,B);
     Q←(DA[1]<DB[1]|B/A|A/B)
     !END!;
(EA|B|A)
!END!;


!PROC!GCD=(!POLY!A1,B1)!POLY!;
!C!
   FINDS THE GREATEST COMMON DIVISOR OF TWO POLYNOMIALS USING THE
   SYLVESTER MATRIX OR BIGRADIENT
!C!
!BEGIN!
!POLY!A←A1,B←B1;
!DEG!DA,DB,DFAC;DF(DA,A);DF(DB,B);
!INT!DIM←(DA[1]+DA[2]+DB[1]+DB[2]=0|0
          |:DA[2]+DB[2]=0!AND!DA[1]+DB[1]>0|1
          |:DA[1]+DB[1]=0!AND!DA[2]+DB[2]>0|3|2);
!BOOL!CH←!FALSE!;
(DIM=3|VAR(A);VAR(B);DF(DA,A);DF(DB,B):CH←!TRUE!;DIM=1);
!POLY!FACTOR,CONTENT,CHECK,CONTA,CONTB,DUMMY;
!IF!EA!OR!EB
!THEN!
     !C!   GREATEST COMMON DIVISOR IS ZERO   !C!
     !CLEAR!FACTOR
!ELSE!
     !IF!  DIM=2
     !THEN!
          [1:DA[1]+DB[1]!FLEX!]1:DA[1]+DB[1]!FLEX!]!POLY!BIGRAD;
          !CLEAR!BIGRAD;
          !C!
             REMOVE CONTENT OF A & B AND FIND GCD(CONT(A),CONT(B))
          !C!
          !BOOL!ZEROA←!TRUE!,ZEROB←!TRUE!;
          !FOR!I!FROM!0!TO!DA[1]!DO!
               !BEGIN!
               P!OF!CHECK←[0:0,0:DA[2]]!REAL!;
               (P!OF!CHECK)[0,]←(P!OF!A)[I,];CHECK←%CHECK;
               (!NOT!ECHECK|:ZEROA|CONTA←CHECK:ZEROA←!FALSE!
```

```
                              !CONTA←GCD(CONTA,CHECK))
      'END';
      'C'
         FINDS CONTENT OF A =CONTA
      'C'
'FOR'I'FROM'0'TO'DB[1]'DO'
      'BEGIN'
      P'OF'CHECK←[0:0,0:DB[2]]'REAL';
      (P'OF'CHECK)[0,]←(P'OF'B)[I,];CHECK←%CHECK;
      ('NOT'ECHECK|:ZEROB|CONTB←CHECK;ZEROB←'FALSE'
                              !CONTB←GCD(CONTB,CHECK))
      'END';
      'C'
         FINDS CONTENT OF B =CONTB
      'C'
A←A/CONTA;DF(DA,A);
B←B/CONTB;DF(DB,B);
CONTENT←GCD(CONTA,CONTB);
'C'
   SETS UP THE BIGRADIENT MATRIX
'C'
'FOR'I'TO'DB[1]'DO'
      'FOR'J'FROM'0'TO'DA[1]'DO'
           (P'OF'DUMMY←[0:0,0:DA[2]]'REAL';
           'FOR'L'FROM'0'TO'DA[2]'DO'
                (P'OF'DUMMY)[0,L]←(P'OF'A)[DA[1]-J,L];
           BIGRAD[I,I+J]←%DUMMY);
'FOR'I'TO'DA[1]'DO'
      'FOR'J'FROM'0'TO'DB[1]'DO'
           (P'OF'DUMMY←[0:0,0:DB[2]]'REAL';
           'FOR'L'FROM'0'TO'DB[2]'DO'
                (P'OF'DUMMY)[0,L]←(P'OF'B)[DB[1]-J,L];
           BIGRAD[DB[1]+I,J-I+DA[1]+1]←%DUMMY);
'IF' EDET(BIGRAD)
'THEN'
      'BEGIN'
      'C'
         REDUCES THE BIGRADIENT TO FIND THE SUB-RESULTANT
         AND SO THE GREATEST COMMON DIVISOR
      'C'
      'BOOL'RUN←'TRUE','EQUAL←'FALSE';
      'WHILE'RUN'DO'
            'BEGIN'
            'POLY'DTRM;
            'IF' 'UPR'BIGRAD=2
            'THEN'
                 RUN←'FALSE';DTRM←DET(BIGRAD);
                 (EDTRM|EQUAL←'TRUE')
            'ELSE'
                 INNER(BIGRAD);DTRM←DET(BIGRAD);
                 ('NOT'EDTRM|RUN←'FALSE')
            'FI'
            'END';
      SUBRES(BIGRAD,A,B);
      FACTOR←DET(BIGRAD);
      (EQUAL|FACTOR←A);
      'C'   REMOVE CONTENT OF FACTOR   'C'
      DF(DFAC,FACTOR);ZEROA←'TRUE';
      'FOR'I'FROM'0'TO'DFAC[1]'DO'
           'BEGIN'
           P'OF'CHECK←[0:0,0:DFAC[2]]'REAL';
           (P'OF'CHECK)[0,]←(P'OF'FACTOR)[I,];
```

```
                    CHECK←%CHECK;
                    ('NOT'ECHECK|:ZEROA|CONTA←CHECK;ZEROA←'FALSE'
                                        |CONTA←GCD(CONTA,CHECK))
                    'END';
               FACTOR←FACTOR/CONTA
               'END'
          'ELSE'
               'C'   COMMON FACTOR IS UNITY   'C'
               (P'OF'FACTOR)[0,0]←1.0
          'FI'
     'ELSE'
          FACTOR←(DIM=0|A|GCD1(A,B))
     'FI';
     (DIM=2|FACTOR'TIMES'CONTENT);
     (CK|VAR(FACTOR));
     DF(DFAC,FACTOR);
     FACTOR'TIMES'(1.0/(P'OF'FACTOR)[DFAC[3],DFAC[4]])
'FI';
%FACTOR
'END';


'PROC'LCMC=([,]'POLY'K,'INT'R)'POLY';
'C'
    FINDS THE LEAST COMMON MULTIPLE OF A COLUMN  R  OF A MATRIX  K
'C'
'BEGIN'
'INT'M←1'UPB'K,N←2'UPB'K;
'POLY'LCM,A;
'BOOL'STRT←'FALSE';
'FOR'I'TO'M'DO'
     'BEGIN'
     'IF'  STRT
     'THEN'
          'IF' 'NOT'EK[I,R]
          'THEN'
               A←LCM*K[I,R];
               LCM←A/GCD(LCM,K[I,R]);
               ('NOT'EA|PRS("ERROR IN CALULATING THE LCM"))
          'FI'
     'ELSE'
          ('NOT'EK[I,R]|LCM←K[I,R];STRT←'TRUE')
     'FI'
     'END';
LCM
'END';
```

APPENDIX B

PROGRAM LISTINGS

THE SMITH FORM ALGORITHM OVER $\mathbb{R}[s,z]$ OR $\mathbb{R}[z][s]$

```
!PROC!MONIC=(!REF![,]!POLY!K,POST,!INT!R):
!C!
   DIVIDES THE PIVOTAL COLUMN TO MAKE THE PIVOT MONIC OVER R[S,Z]
   OR R[Z][S]
!C!
!BEGIN!
!INT!N←1!UPB!K,N←2!UPB!K;
!REAL!CONST;
!BOOL!RUN←!TRUE!;
!DEG!D:
DF(D,K[R,R]);
CONST←(RAT(P!OF!K[R,R])[D[1],D[5]],(P!OF!K[R,R])[D[3],D[4]]);
!FOR!I!TO!N!DO!
     POST[I,R]!TIMES!(1.0/CONST);
!FOR!I!FROM!R!TO!N!DO!
     K[I,R]!TIMES!(1.0/CONST)
!C!
   DIVIDES COLUMN R TO MAKE PIVOT MONIC
!C!
!END!;


!PROC!LSTDEG=(!REF![,]!POLY!K,PRE,POST,!INT!R,!REF!!BOOL!ZERO,CHANGE):
!C!
   FINDS THE POLYNOMIAL OF LEAST DEGREE AND MOVES IT TO POSITION (R,R)
!C!
!BEGIN!
!INT!N←1!UPB!K,N←2!UPB!K,II←R,JJ←R,D1,DG,DR;
!DEG!D:
```

```
ZERO←'TRUE';CHANGE←'FALSE';
'FOR'I'FROM'R'TO'N'DO'
     'FOR'J'FROM'R'TO'N'DO'
          'BEGIN'
          'POLY'A←K[I,J];
          DF(D,A);
          'IF'     'NOT'EA
          'THEF'   ZERO
          'THEN'              D1←D[1];DG←(RAT|D[5]|D[3]←D[4]);
                              II←I;JJ←J;ZERO←'FALSE'
          'ELSF'   RAT'AND'D[1]<D1
          'THEN'              D1←D[1];DG←D[5];II←I;JJ←J
          'ELSF'   RAT'AND'D[1]=D1'AND'D[5]<DG
          'THEN'              DG←D[5];II←I;JJ←J
          'ELSF'   'NOT'RAT 'AND' D[3]←D[4]<DG
          'THEN'              DG←D[3]←D[4];D1←D[1];II←I;JJ←J
          'ELSF'   'NOT'RAT 'AND' D[3]←D[4]=DG 'AND' D[1]<D1
          'THEN'              D1←D[1];II←I;JJ←J
          'FI'
          'END';
'C'
   FIRSTLY SEARCHES THE SUB-MATRIX FOR THE FIRST NON-ZERO ELEMENT
   THEN CONTINUES FROM THERE TO SEARCH FOR THE NON-ZERO ELEMENT
   OF LEAST DEGREE AT POSITION  (II,JJ)
   IF THERE ARE NO NON-ZERO ELEMENTS THE  BOOL  ZERO  WILL REMAIN TRUE
'C'
'IF' II#R
'THEN'
     [1:N]'POLY'A←K[II,];K[II,]←K[R,];K[R,]←A;
     [1:M]'POLY'B←PRE[I,];PRE[II,]←PRE[R,];PRE[R,]←B;
     CHANGE←'TRUE'
'FI';
'C'   INTERCHANGES ROWS II,R   'C'
'IF'JJ#R
'THEN'
     [1:M]'POLY'A←K[,JJ];K[,JJ]←K[,R];K[,R]←A;
     [1:N]'POLY'B←POST[,JJ];POST[,JJ]←POST[,R];POST[,R]←B;
     CHANGE←'TRUE'
'FI'
'C'
   INTERCHANGES COLUMNS JJ,R
   AND HAS MOVED ELEMENT OF LEAST DEGREE TO POSITION (R,R)
'C'
'END';


'PROC'GAUSS=('REF'[,]'POLY'K,PRE,POST,'INT'R);
'C'
   CARRIES OUT ONE FULL STEP OF GAUSSIAN ELIMINATION
'C'
'BEGIN'
'POLY'PIV←K[R,R],Q;
'INT'M←1'UPR'K,N←2'UPB'K,I,J;
'BOOL'CHANGE,DEGEN←'FALSE',NOREM←'TRUE','RUN←'TRUE',ZERO;
'DEG'D;
I←R;J←R+1;
(J>N|I←R+1;J←R);
(J=R'AND'I>M|RUN←'FALSE');
'WHILE'RUN'DO'
     'BEGIN'
     (J>N|I←R+1;J←R);
     'IF''NOT'EK[I,J]
```

```
    'THEN'
        'IF'DEGEN
        'THEN'
            'POLY'PZ;
            DF(D,PIV);
            P'OF'PZ←[0:0,0:D[5]]'REAL';
            'FOR'JJ'FROM'0'TO'D[5]'DO'
                (P'OF'PZ)[0,JJ]←(P'OF'PIV)[D[1],JJ];
            'IF'I=R
            'THEN'
                'FOR'II'TO'N'DO'
                    POST[II,J]'TIMES'PZ;
                'FOR'II'FROM'R'TO'M'DO'
                    K[II,JJ]'TIMES'PZ
                'C'   MULTIPLIES COLUMN J BY PZ   'C'
            'ELSE'
                'FOR'JJ'TO'M'DO'
                    PRE[I,JJ]'TIMES'PZ;
                'FOR'JJ'FROM'R'TO'N'DO'
                    K[I,JJ]'TIMES'PZ
                'C'   MULTIPLIES ROW I BY PZ   'C'
            'FI'
        'FI';
        Q←K[I,J]/PIV;
        'IF'NOT'£Q
        'THEN'
            'IF'I=R
            'THEN'
                'FOR'II'TO'N'DO'
                    POST[II,J]'MINUS'(Q*POST[II,R]);
                'FOR'II'FROM'R+1'TO'M'DO'
                    K[II,JJ]'MINUS'(Q*K[II,R])
                'C'   SUBTRACTS Q TIMES COLUMN R FROM COLUMN J   'C'
            'ELSE'
                'FOR'JJ'TO'M'DO'
                    PRE[I,JJ]'MINUS'(Q*PRE[R,JJ]);
                'FOR'JJ'FROM'R+1'TO'N'DO'
                    K[I,JJ]'MINUS'(Q*K[R,JJ])
                'C'   SUBTRACTS Q TIMES ROW R FROM ROW I   'C'
            'FI';
            LSTDEG(K,PRE,POST,R,ZERO,CHANGE);
            (CHANGE|I←R;J←R+1;PIV←K[R,R];NOREM←'TRUE'
                |(¬'NOT'£K[I,JJ]NOREM←'FALSE');
                 (I=PIJ'PLUS'1|I'PLUS'1))
        'ELSE'
            NOREM←'FALSE';(I=R|J'PLUS'1|I'PLUS'1)
        'FI'
    'ELSE'
        (I=R|J'PLUS'1|I'PLUS'1)
    'FI';
    (I=R'AND'J>N|J←R;I←R+1);
    (J=R'AND'I>M:NOREM|RUN←'FALSE'
                    |I←R;J←R+1;NOREM←'TRUE';
                     DEGEN←'TRUE'|RAT←'TRUE')
    'END';
MONIC(K,POST,R)
'END';


'PROC'FACCHECK=('REF'[,]'POLY'K,POST,'INT'R,'REF''BOOL'FACTOR);
'C'
   CHECKS WHETHER THE PIVOT IS A FACTOR OF THE REMAINING SUB-MATRIX
```

```
'C'
'BEGIN'
'INT'M←1'UPB'K,N←2'UPB'K,JJ;
FACTOR←'TRUE';
'POLY'FAC←K[R,R];
'FOR'I'FROM'R←1'TO'M'WHILE'FACTOR'DO'
     'FOR'J'FROM'R←1'TO'N'WHILE'FACTOR'DO'
          'BEGIN'
          'POLY'A←K[I,J];
          'IF' 'NOT'£A
          'THEN'
               'BEGIN'
               'POLY'Q←A/FAC;
               'IF' 'NOT'£A
               'THEN'    FACTOR←'FALSE':JJ←J
               'FI'
               'END'
          'FI'
          'END';
'IF'  'NOT'FACTOR
'THEN'
     'FOR'I'TO'M'DO'  K[I,R]'PLUS'K[I,JJ];
     'FOR'I'TO'N'DO'  POST[I,R]'PLUS'POST[I,JJ]
'FI'
'C'   IF REMAINDER≠0 AND COL JJ TO COL R   'C'
'END';


'PROC'SMITHFORM=('REF'[,]'POLY'K,PRE,POST);
'BEGIN'
'INT'M←1'UPB'K,N←2'UPB'K,ORD←(M>N|N|M);
'BOOL'FACTOR,RESTRT←'TRUE',CHANGE,ZERO;
RST: 'FOR'R'TO'ORD'DO'
     'BEGIN'
     FACTOR←'FALSE';
     'WHILE''NOT'FACTOR'DO'
          'BEGIN'
          LSTDEG(K,PRE,POST,R,ZERO,CHANGE);
          (ZERO|'GOTO'XIT);
          GAUSS(K,PRE,POST,R);
          (RAT'AND'RESTRT|RESTRT←'FALSE';'GOTO'RST);
          FACCHECK(K,POST,R,FACTOR)
          'END'
     'END';
XIT: (RAT|PRC("=",37);PRS("FULL POLYNOMIAL EQUIVALENCE NOT FOUND");
      PRC("=",37);NL(3))
'END';


'PROC'DET=([,]'POLY'K)'POLY';
'C'
   FINDS THE DETERMINANT OF A POLYNOMIAL MATRIX BY EXPANSION ALONG THE
   FIRST ROW AND RECURSION FOR LOWER ORDER DETERMINANTS
'C'
'BEGIN'
'INT'N←'UPB'K;
'POLY'DTRM;
P'OF'DTRM←[0:0,0:0]'REAL';
(P'OF'DTRM)[0,0]←0.0;
'REAL'SIGN←1.0;
'IF'       N>2
'THEN'
```

```
      'FOR'I'TO'N'DO'
            'BEGIN'
            [1:N-1,1:N-1]'POLY'SL;
            'IF'          I=1
            'THEN'              SL←K[2:N,2:N]
            'ELSF'        I=N
            'THEN'              SL←K[2:N,1:N-1]
            'ELSE'              SL[,1:I-1]←K[2:N,1:I-1];
                                SL[,I:N-1]←K[2:N,I+1:N]
            'FI';
            ('NOT'EK[1,I]|DTRM'PLUS'(K[1,I]*DET(SL)*SIGN));
            SIGN←-SIGN
            'END'
'ELSF'      N=1
'THEN'      DTRM←K[1,1]
'ELSE'      DTRM←K[1,1]*K[2,2]-K[1,2]*K[2,1]
'FI';
%DTRM
'END';


'PROC'INNER=('REF'[,]'POLY'K);
'C'
   FINDS THE INNER SQUARE OF A MATRIX
'C'
'BEGIN'
'INT'N←'UPB'K;
[1:N-2,1:N-2]'POLY'INN←K[2:N-1,2:N-1];
K←INN
'END';


'PROC'SUBRES=('REF'[,]'POLY'K,'POLY'A,B);
'C'
   FINDS THE SUBRESULTANT OF THE BIGRADIENT MATRIX USED IN FINDING THE
   GREATEST COMMON DIVISOR
'C'
'BEGIN'
'DEG'DA,DB:DF(DA,A):DF(DB,B);
'INT'N←'UPB'K,NN←(DA[1]+DB[1]-N)'/'2;
[1:N,1:N]'POLY'SBRS←K;
'POLY'S;
P'OF'S←[0:1,0:0]'REAL';
(P'OF'S)[,0]←(0.0,1.0);
'IF' (DB[1]-NN)>0
'THEN'
      SBRS[DB[1]-NN,N]←A;
      ((DB[1]-NN)>1|'FOR'I'FROM'DB[1]-NN-1'BY'-1'TO'1'DO'
                  SBRS[I,N]←S*SBRS[I+1,N])
'FI';
'IF' (DA[1]-NN)>0
'THEN'
      SBRS[DB[1]-NN+1,N]←B;
      ((DA[1]-NN)>1|'FOR'I'FROM'N-DA[1]+NN+2'TO'N'DO'
                  SBRS[I,N]←S*SBRS[I-1,N])
'FI';
K←SBRS
'END';


'PROC'VAR=('REF''POLY'A):
'C'
```

```
   CHANGES THE VARIABLES OF A POLYNOMIAL AROUND  S TO Z & Z TO S
!C!
!BEGIN!
!DEG!D;DF(D,A);
!POLY!B;
P!OF!B←[0:D[2],0:D[1]]!REAL!;
!FOR!L!FROM!0!TO!D[1]!DO!
     (P!OF!B)[,L]←(P!OF!A)[L,];
A←B
!END!;


!PROC!VARCH=(!REF![,]!POLY!K);
!C!
   CHANGES THE VARIABLES OF A POLYNOMIAL MATRIX
!C!
!BEGIN!
!INT!M←1!UPB!K,N←2!UPB!K;
!FOR!I!TO!M!DO!
     !FOR!J!TO!N!DO!
          VAR(K[I,J])
!END!;


!PROC!GCD1=(!POLY!A1,B1)!POLY!;
!C!
   FINDS THE GREATEST COMMON DIVISOR OF TWO POLYNOMIALS IN ONE
   VARIABLE USING BLANKINSHIPS ALGORITHM
!C!
!BEGIN!
!POLY!A←A1,B←B1,Q;
!DEG!DA,DB;
!WHILE! !NOT!EA !AND! !NOT!EB !DO!
     !BEGIN!
     DF(DA,A);DF(DB,B);
     Q←(DA[1]<DB[1]|B/A|A/B)
     !END!;
(EA|B|A)
!END!;


!PROC!GCD=(!POLY!A1,B1)!POLY!;
!C!
   FINDS THE GREATEST COMMON DIVISOR OF TWO POLYNOMIALS USING THE
   SYLVESTER MATRIX OR DIGRADIENT
!C!
!BEGIN!
!POLY!A←A1,B←B1;
!DEG!DA,DB,DFAC;DF(DA,A);DF(DB,B);
!INT!DIM←(DA[1]+DA[2]+DB[1]+DB[2]=0|0
          |:DA[2]+DB[2]=0!AND!DA[1]+DB[1]>0|1
          |:DA[1]+DB[1]=0!AND!DA[2]+DB[2]>0|3|2);
!BOOL!CH←!FALSE!;
(DIM=3|VAR(A);VAR(B);DF(DA,A);DF(DB,B);CH←!TRUE!;DIM=1);
!POLY!FACTOR,CONTENT,CHECK,CONTA,CONTB,DUMMY;
!IF!EA!OR!EB
!THEN!
     !C!   GREATEST COMMON DIVISOR IS ZERO   !C!
     !CLEAR!FACTOR
!ELSE!
     !IF!  DIM=2
     !THEN!
```

```
[1:DA[1]←DB[1]'FLEX';1:DA[T]←DB[1]'FLEX']'POLY'BIGRAD;
'CLEAR'BIGRAD;
'C'
   REMOVE CONTENT OF A & B AND FIND GCD(CONT(A);CONT(B))
'C'
'BOOL'ZERCA←'TRUE',ZERCB←'TRUE';
'FOR'I'FROM'0'TO'DA[1]'DO'
      'BEGIN'
      P'OF'CHECK←[0:0;0:DA[2]]'REAL';
      (P'OF'CHECK)[0,]←(P'OF'A)[I;];CHECK←%CHECK;
      ('NOT'ECHECK|:ZERCA|CONTA←CHECK;ZEROA←'FALSE'
                        |CONTA←GCD(CONTA,CHECK))
      'END';
      'C'
         FINDS CONTENT OF A =CONTA
      'C'
'FOR'I'FROM'0'TO'DB[1]'DO'
      'BEGIN'
      P'OF'CHECK←[0:0;0:DB[2]]'REAL';
      (P'OF'CHECK)[0,]←(P'OF'B)[I;];CHECK←%CHECK;
      ('NOT'ECHECK|:ZEROB|CONTB←CHECK;ZEROB←'FALSE'
                        |CONTB←GCD(CONTB,CHECK))
      'END';
      'C'
         FINDS CONTENT OF B =CONTB
      'C'
A←A/CONTA;DF(DA,A);
B←B/CONTB;DF(DB,B);
CONTENT←GCD(CONTA,CONTB);
'C'
   SETS UP THE BIGRADIENT MATRIX
'C'
'FOR'I'TO'DB[1]'DO'
      'FOR'J'FROM'0'TO'DA[1]'DO'
           (P'OF'DUMMY←[0:0,0:DA[2]]'REAL';
           'FOR'L'FROM'0'TO'DA[2]'DO'
                (P'OF'DUMMY)[0,L]←(P'OF'A)[DA[1]-J,L];
           BIGRAD[I,I+J]←%DUMMY);
'FOR'I'TO'DA[1]'DO'
      'FOR'J'FROM'0'TO'DB[1]'DO'
           (P'OF'DUMMY←[0:0,0:DB[2]]'REAL';
           'FOR'L'FROM'0'TO'DB[2]'DO'
                (P'OF'DUMMY)[0,L]←(P'OF'B)[DB[1]-J,L];
           BIGRAD[DB[1]+I,J=I+DA[1]+1]←%DUMMY);
'IF' EDET(BIGRAD)
'THEN'
      'BEGIN'
      'C'
         REDUCES THE BIGRADIENT TO FIND THE SUB=RESULTANT
         AND SO THE GREATEST COMMON DIVISOR
      'C'
      'BOOL'RUN←'TRUE',EQUAL←'FALSE';
      'WHILE'RUN'DO'
            'BEGIN'
            'POLY'DTRM;
            'IF'  'UPB'BIGRAD=2
            'THEN'
                 RUN←'FALSE';DTRM←DET(BIGRAD);
                 (EDTRM|EQUAL←'TRUE')
            'ELSE'
                 INNER(BIGRAD);DTRM←DET(BIGRAD);
                 ('NOT'EDTRM|RUN←'FALSE')
```

```
                    'FI'
                    'END';
               SUBRES(BIGMAD,A,B);
               FACTOR←DET(BIGMAD);
               (EQUAL|FACTOR←A);
               'C'   REMOVE CONTENT OF FACTOR   'C'
               DF(DFAC,FACTOR);ZEROA←'TRUE';
               'FOR'I'FROM'0'TO'DFAC[1]'DO'
                    'BEGIN'
                    P'OF'CHECK←[0:0,0:DFAC[2]]'REAL';
                    (P'OF'CHECK)[0,]←(P'OF'FACTOR)[I,];
                    CHECK←%CHECK;
                    ('NOT'CHECK|:ZEROA|CONTA←CHECK;ZEROA←'FALSE'
                                        |CONTA←GCD(CONTA,CHECK))
                    'END';
               FACTOR←FACTOR/CONTA
               'END'
          'ELSE'
               'C'   COMMON FACTOR IS UNITY   'C'
               (P'OF'FACTOR)[0,0]←1.0
          'FI'
     'ELSE'
          FACTOR←(DIM=0|A|GCD1(A,B))
     'FI';
     (DIM=2|FACTOR'TIMES'CONTENT);
     (CH|VAR(FACTOR));
     DF(DFAC,FACTOR);
     FACTOR'TIMES'(1.0/(P'OF'FACTOR)[DFAC[3],DFAC[4]])
'FI';
%FACTOR
'END';


'PROC'SMITH=('REF'[,]'POLY'K);
'C'
   MAIN PROCEDURE TO FIND THE SMITH FORM OF AN INPUTTED POLYNOMIAL
   MATRIX  K  OF DIMENSIONS  M*N
'C'
'BEGIN'
'INT'M←1'UPB'K,N←2'UPB'K,ORD←(M<N|M|N),LA1←0,LB1←0;
[1:M,1:N]'POLY'K1←K,K2←K,K3←K;
[1:M,1:M]'POLY'PRE,PRE1;'CLEAR'PRE;
[1:N,1:N]'POLY'POST,POST1;'CLEAR'POST;
'POLY'UNIT;
(P'OF'UNIT)[0,0]←1.0;
'FOR'I'TO'M'DO'  PRE[I,I]←UNIT;
'FOR'I'TO'N'DO'  POST[I,I]←UNIT;
PRE1←PRE;POST1←POST;
VARCH(K1);
'BOOL'RAT←'FALSE';
SMITHFORM(K,PRE,POST);
'IF'  RAT
'THEN'
   'BEGIN'
   PRS("PRE-MULTIPLYING EQUIVALENCE MATRIX");PRC("=",34);
   PRINTPOLYMX(PRE);
   PRS("DETERMINANT(PRE)");PRC("=",16);
   PRINTPOLY(DET(PRE));PRC("+",60);NL(1);
   PRS("POST-MULTIPLYING EQUIVALENCE MATRIX");PRC("=",35);
   PRINTPOLYMX(POST);
   PRS("DETERMINANT(POST)");PRC("=",17);
   PRINTPOLY(DET(POST));PRC("+",60);NL(1);
```

```
     PRS("SMITH FORM OVER R[Z][S]");PRC("=",23);
     PRINTPOLYMX(K);
     PRS("PRE * K * POST");PRC("-",14);
     K3←PRE*K2*POST;
     PRINTPOLYMX(K3);
     NL(4);PRC("-",33);PRS("SEARCH FOR SMITH FORM OVER R[Z,S]");
     PRC("-",33);NL(2);
     RAT←'FALSE';
     SMITHFORM(K1,PRE1,POST1);
     VARCH(PRE1);VARCH(POST1);VARCH(K1);
     PRS("PRE-MULTIPLYING EQUIVALENCE MATRIX");PRC("=",34);
     PRINTPOLYMX(PRE1);
     PRS("DETERMINANT(PRE)");PRC("-",16);
     PRINTPOLY(DET(PRE1));PRC("+",60);NL(1);
     PRS("POST-MULTIPLYING EQUIVALENCE MATRIX");PRC("-",35);
     PRINTPOLYMX(POST1);
     PRS("DETERMINANT(POST)");PRC("-",17);
     PRINTPOLY(DET(POST1));PRC("+",60);NL(1);
     'IF' RAT
     'THEN'
     PRS("SMITH FORM OVER R[S][Z]");PRC("=",23)
     'ELSE'
     PRS("CORRECT SMITH FORM OVER R[S,Z]");PRC("=",30)
     'FI';
     PRINTPOLYMX(K1);
     PRS("PRE * K * POST");PRC("-",14);
     K3←PRE1*K2*POST1;
     PRINTPOLYMX(K3);
     'IF'RAT
     'THEN'
          'FOR'I'TO'ORD'DO'     K[I,I]←GCD(K[I,I],K1[I,I]);
          PRS("CALCULATED SMITH FORM OVER R[S,Z]");PRC("=",33);
          PRINTPOLYMX(K)
     'FI'
     'END'
'ELSE'
     PRS("PRE-MULTIPLYING EQUIVALENCE MATRIX");PRC("=",34);
     PRINTPOLYMX(PRE);
     PRS("DETERMINANT(PRE)");PRC("-",16);
     PRINTPOLY(DET(PRE));PRC("+",60);NL(1);
     PRS("POST-MULTIPLYING EQUIVALENCE MATRIX");PRC("-",35);
     PRINTPOLYMX(POST);
     PRS("DETERMINANT(POST)");PRC("-",17);
     PRINTPOLY(DET(POST));PRC("+",60);NL(1);
     PRS("CORRECT SMITH FORM OVER R[S,Z]");PRC("=",30);
     PRINTPOLYMX(K);
     K3←PRE*K2*POST;
     PRS("PRE * K * POST");PRC("-",14);
     PRINTPOLYMX(K3)
'FI'
'END';


'PROC'RUNSMITH='VOID';
'C'
   RUNS THE SMITH FORM ALGORITHM
   READING IN THE INITIAL POLYNOMIAL MATRIX  K
'C'
'BEGIN'
'INT'M,N;
'C'
   READ IN THE POLYNOMIAL MATRIX  K
```

```
'C'
READ((M,N));
[1:M,1:N]'POLY'K;
READPOLYMX(K);
SP(13);PRS("SSSSS M   M III TTTTT H   H   A");
SP(13);PRS("S     MM MM  I     T   H   H  A A");
SP(13);PRS("SSSSS M M M  I     T   HHHHH AAAAA");
SP(13);PRS("    S M   M  I     T   H   H A   A");
SP(13);PRS("SSSSS M   M III    T   H   H A   A");
NL(3);PRC("-",20);PRS("TOLERANCE SET=1.0E-6");PRC("-",20);NL(2);
PRC("=",37);PRS("INITIAL POLYNOMIAL MATRIX OVER R[S,Z]");PRC("=",37);
PRINTPOLYMX(K);
SMITH(K);
NL(3);PRC("A",60);PRC("S",60)
'END';
```

APPENDIX C

PROGRAM LISTINGS

THE SMITH FORM ALGORITHM FROM THE DEFINITION

```
'END';


'PROC'GCD=('POLY'A1,B1)'POLY';
'C'
   FINDS THE GREATEST COMMON DIVISOR OF TWO POLYNOMIALS USING THE
   SYLVESTER MATRIX OR BIGRADIENT
'C'
'BEGIN'
'POLY'A←A1,B←B1;
'DEG'DA,DB,DFAC:DF(DA,A):DF(DB,B);
'INT'DIM←(DA[1]+DA[2]=0'OR'DB[1]+DB[2]=0|0
          |:DA[2]+DB[2]=0'AND'DA[1]+DB[1]>0|1
          |:DA[1]+DB[1]=0'AND'DA[2]+DB[2]>0|3|2);
'BOOL'CH←'FALSE';
(DIM=3|VAR(A):VAR(B):DF(DA,A):DF(DB,B):CH←'TRUE';DIM=1);
[1:DA[1]+DB[1]'FLEX',1:DA[1]+DB[1]'FLEX']'POLY'BIGRAD;
'CLEAR'BIGRAD;
'POLY'FACTOR,CONTENT,CHECK,CONTA,CONTB,DUMMY;
'IF'EA'OR'EB
'THEN'
     'C'   GREATEST COMMON DIVISOR IS ZERO   'C'
     'CLEAR'FACTOR
'ELSE'
     'IF' DIM#0
     'THEN'
       'IF' DIM=2
          'THEN'
               'C'
                  REMOVE CONTENT OF A & B AND FIND GCD(CONT(A),CONT(B));
               'C'
               'BOOL'ZEROA←'TRUE',ZEROB←'TRUE';
               'FOR'I'FROM'0'TO'DA[1]'DO'
                    'BEGIN'
                    P'OF'CHECK←[0:0,0:DA[2]]'REAL';
                    (P'OF'CHECK)[0,]←(P'OF'A)[I,];CHECK←%CHECK;
                    ('NOT'ECHECK|:ZEROA|CONTA←CHECK;ZEROA←'FALSE'
                                        |CONTA←GCD(CONTA,CHECK))
                    'END';
                    'C'
                      FINDS CONTENT OF A =CONTA
                    'C'
               'FOR'I'FROM'0'TO'DB[1]'DO'
                    'BEGIN'
                    P'OF'CHECK←[0:0,0:DB[2]]'REAL';
                    (P'OF'CHECK)[0,]←(P'OF'B)[I,];CHECK←%CHECK;
                    ('NOT'ECHECK|:ZEROB|CONTB←CHECK;ZEROB←'FALSE'
                                        |CONTB←GCD(CONTB,CHECK))
                    'END';
                    'C'
                      FINDS CONTENT OF B =CONTB
                    'C'
               A←A/CONTA;DF(DA,A);
               B←B/CONTB;DF(DB,B);
               CONTENT←GCD(CONTA,CONTB)
          'FI';
          'FOR'I'TO'DB[1]'DO'
               'FOR'J'FROM'0'TO'DA[1]'DO'
                    (P'OF'DUMMY←[0:0,0:DA[2]]'REAL';
                    'FOR'L'FROM'0'TO'DA[2]'DO'
                       (P'OF'DUMMY)[0,L]←(P'OF'A)[DA[1]-J,L];
                    BIGRAD[I,I+J]←%DUMMY);
```

```
        'FOR'I'TO'DA[1]'DO'
            'FOR'J'FROM'C'TO'DB[1]'DO'
                (P'OF'DUMMY←[0:0,0:DA[2]1'REAL';
                'FOR'L'FROM'0'TO'DB[2]'DO'
                   (P'OF'DUMMY)[0,L]←(P'OF'L)[DB[1]-J,L];
              BIGRAD[DB[1]+I,J-I+DA[1]+1]←%DUMMY);
        'C'
           SETS UP THE BIGRADIENT MATRIX
        'C'
        'IF' EDET(BIGRAD)
        'THEN'
            'BEGIN'
            'BOOL'RUN←'TRUE',EQUAL←'FALSE';
            'WHILE'RUN'DO'
                  'BEGIN'
                  'POLY'DTRM;
                  'IF' 'UPB'BIGRAD=2
                  'THEN'
                       RUN←'FALSE';DTRM←DET(BIGRAD);
                       (EDTRM|EQUAL←'TRUE')
                  'ELSE'
                       INNER(BIGRAD);DTRM←DET(BIGRAD);
                       ('NOT'EDTRM|RUN←'FALSE')
                  'FI';
                  'END';
            SUBRES(BIGRAD,A,B);
            FACTOR←DET(BIGRAD);
            (EQUAL|FACTOR←A)
            'C'
              REDUCES THE BIGRADIENT TO FIND THE SUB-RESULTANT
              AND SO THE GREATEST COMMON DIVISOR
            'C'
            'END'
        'ELSE'
             'C'   COMMON FACTOR IS UNITY   'C'
            P'OF'FACTOR←[0:0,0:0]'REAL';(P'OF'FACTOR)[0,0]←1.0
        'FI'
    'ELSE'
        FACTOR←(DA[1]+DA[2]=0|A|B)
    'FI';
    (DIM=2|FACTOR'TIMES'CONTENT);
    (CH|VAR(FACTOR));
    DF(DFAC,FACTOR);
    'MONIC'FACTOR
'FI';
%FACTOR
'END';


'PROC'SEARCHCOMB=('INT'N,R)[,]'INT';
'C'
   EVALUATES ALL THE COMBINATIONS OF  R  ELEMENTS FROM  N  ELEMENTS
   IN A SET ORDER STARTING WITH  (1,2,....,R)
   AND FINISHING WITH  (N-R+1,N-R+2,....,N)
'C'
'BEGIN'
'INT'NCOM←1,RFAC←1,COUNT←0,DUMMY;
'FOR'I'TO'R'DO'  (RFAC'TIMES'I;NCOM'TIMES'(N-I+1));
NCOM←NCOM'/'RFAC;
[1:NCOM,1:R]'INT'LIST;'CLEAR'LIST;
[1:R]'INT'COM;
'FOR'I'TO'R'DO'  COM[I]←I;
```

```
LIST[1,]←COM;
'FOR'I'FROM'2'TO'NCOM'DO'
     'BEGIN'
     'IF' COM[R]=N
     'THEN'
          'WHILE' COM[R-COUNT]=N-COUNT 'DO' COUNT'PLUS'1;
          COM[R-COUNT]'PLUS'1;
          DUMMY←COM[R-COUNT];
          'FOR'J'FROM'R-COUNT+1'TO'R'DO'
               (COM[J]←DUMMY;DUMMY'PLUS'1);
          COUNT←0
     'ELSE'
          COM[R]'PLUS'1
     'FI';
     LIST[I,]←COM
     'END';
LIST
'END';


'PROC'DTRM=([,]'INT'RN,CN;RN1,CN1;[,]'POLY'MINN,K,'REF'[,]'POLY'MINN1);
'C'
   FINDS THE DETERMINANTS OF ALL THE  N+1  TH ORDER MINORS BY EXPANSIO
   ALONG THE FIRST ROW AND USE OF THE PREVIOUSLY CALCULATED
   DETERMINANTS OF ALL THE  N  TH ORDER MINORS
'C'
'BEGIN'
'INT'NR←1'UPB'MINN;NC←2'UPB'MINN,NR1←1'UPB'MINN1;NC1←2'UPB'MINN1;
     N←2'UPB'RN,R,C;II,JJ;
'BOOL'RUN;
[1:N]'INT'ROW,COL;
'POLY'A;
'REAL'SIGN←1.0;
'FOR'I'TO'NR1'DO'
     'FOR'J'TO'NC1'DO'
          'BEGIN'
          'C'
             CONSIDER THE MINOR CORRESPONDING TO THE  I  TH SET
             OF ROWS AND THE  J  TH SET OF COLUMNS
             EXPAND ALONG THE FIRST ROW OF THE MINOR USING THE
             LOWER ORDER MINORS DEFINED BY  RN,CN,MINN
          'C'
          'CLEAR'A;
          SIGN←1.0;
          ROW←RN1[I,2:N+1];R←RN1[I,1];
          RUN←'TRUE';
          'FOR'M'TO'NR'WHILE'RUN'DO'
               'C'   FIND THE CORRESPONDING ROW SET   'C'
               (ROW=RN[M];RUN←'FALSE';II←M);
          'FOR'L'TO'N+1'DO'
               'BEGIN'
               C←CN1[J,L];
               (L=1|COL←CN1[J,2:N+1]
                   |:L=N+1|COL←CN1[J,1:N]
                          |COL[1:L-1]←CN1[J,1:L-1];COL[L:N]←CN1[J,L+1:
               RUN←'TRUE';
               'FOR'M'TO'NC'WHILE'RUN'DO'
                    'C'   FIND THE CORRESPONDING COLUMN SET   'C'
                    (COL=CN[M];RUN←'FALSE';JJ←M);
               A'PLUS'(K[R,C]×MINN[II,JJ]×SIGN);
               SIGN←-SIGN
               'END';
```

```
          MINN1[I,J]←A
          'END'
'END';


'PROC'SMITH=('REF'[,]'POLY'K);
'C'
   MAIN PROCEDURE TO FIND THE DETERMINANTAL DIVISORS OF A
   POLYNOMIAL MATRIX  K  OF DIMENSIONS  M*N
'C'
'BEGIN'
'INT'M←1'UPB'K,N←2'UPB'K,ORD←(M<MIN),ROW,COL;
'BOOL'RANK←'TRUE';
[1:0'FLEX',1:0'FLEX']'INT'RN,CN,RN1,CN1;
[1:0'FLEX',1:0'FLEX']'POLY'MINN,MINN1;
[1:M,1:N]'POLY'K1;'CLEAR'K1;
'POLY'DETFAC,DETFACR;'CLEAR'DETFAC;
P'OF'DETFACR←[0:0,0:0]'REAL';(P'OF'DETFACR)[0,0]←1.0;
'C'
   INITIALIZE  RN1,CN1,MINN1  AS 1ST ORDER MINORS OF  K
'C'
MINN1←K;
RN1←[1:M,1:1]'INT';
'FOR'I'TO'M'DO'  RN1[I,1]←I;
CN1←[1:N,1:1]'INT';
'FOR'I'TO'N'DO'  CN1[I,1]←I;
'FOR'R'TO'ORD'WHILE'RANK'DO'
     'BEGIN'
     'BOOL'UNIT←'FALSE',ZERO←'TRUE';
     MINN←MINN1;
     RN←RN1;CN←CN1;
     'IF'R≠1
     'THEN'
          RN1←SEARCHCOMB(M,R);
          CN1←SEARCHCOMB(N,R);
          MINN1←[1:1'UPB'RN1,1:1'UPB'CN1]'POLY';'CLEAR'MINN1;
          DTRM(RN,CN,RN1,CN1,MINN,K,MINN1)
     'FI';
     ROW←1'UPB'MINN1;
     COL←2'UPB'MINN1;
     'FOR'I'TO'ROW'WHILE''NOT'UNIT'DO'
          'FOR'J'TO'COL'WHILE''NOT'UNIT'DO'
               'BEGIN'
               'IF' 'NOT'ZERO
               'THEN'
                    ('NOT'@MINN1[I,J]DETFAC←GCD(DETFAC,MINN1[I,J]));
                    (1'UPB'P'OF'DETFAC=0'AND'2'UPB'P'OF'DETFAC=0
                     'UNIT←'TRUE')
               'ELSE'
                    ('NOT'@MINN1[I,J]DETFAC←MINN1[I,J];ZERO←'FALSE')
               'FI'
               'END';
     'IF' ZERO
     'THEN'
          RANK←'FALSE';
          PRC("=",22);PRSI("NOT FULL RANK,RANK=",R-1);PRC("=",22);NL(2
     'ELSE'
          'IF' 'NOT'UNIT
          'THEN'
               'FOR'I'TO'ROW'DO'
                    'FOR'J'TO'COL'DO'
                         MINN1[I,J]←MINN1[I,J]/DETFAC;
```

```
            'C'   REMOVE THE  R  TH ORDER DETERMINANTAL DIVISOR   'C
            DETFACR'TIMES'DETFAC
        'FI';
        'MONIC'DETFAC;'MONIC'DETFACR;
        PRC("=",9);PRSI("ORDER=",R);PRC("=",9);
        NL(2);PRS("DETERMINANTAL DIVISOR");PRC("=",21);
        PRINTPOLY(DETFACR);
        NL(2);PRS("INVARIANT POLYNOMIAL");PRC("=",20);
        PRINTPOLY(DETFAC);PRC("=",60);NL(3);
        K1[R,R]←DETFAC
    'FI'
    'END';
K←K1
'END';


'PROC'RUNSMITH='VOID';
'C'
   RUNS THE SMITH FORM ALGORITHM
   READING IN THE INITIAL POLYNOMIAL MATRIX  K  OF DIMENSIONS  M*N
'C'
'BEGIN'
'INT'M,N;
'C'
   READ IN THE POLYNOMIAL MATRIX  K
'C'
READ((M,N));
[1:M,1:N]'POLY'K;
READPOLYMX(K);
SP(16);PRS("SSSSS M   M III TTTTT H   H");
SP(16);PRS("S     MM MM  I    T   H   H");
SP(16);PRS("SSSSS M M M  I    T   HHHHH");
SP(16);PRS("    S M   M  I    T   H   H");
SP(16);PRS("SSSSS M   M III   T   H   H");
NL(3);PRC("=",25);PRS("INITIAL POLYNOMIAL MATRIX");PRC("=",25);
NL(2);PRSI("NO OF ROWS=",M);
PRSI("NO OF COLS=",N);
PRSI("MAX POSSIBLE RANK=",(N<M|M|N));NL(2);
PRINTPOLYMX(K);
SMITH(K);
NL(5);PRC("S",60)
'END';


NL(1)
'END'
'KEEP' 'POLY','DEG',DF,E,%,'CLEAR',READ,/,*,+,-,'PLUS','MINUS','TIMES
       =,READPOLYMX,NL,SP,PRC,PRS,PRSI,PRINTPOLY,PRINTPOLYMX,'MONIC',DI
       INNER,SUBRES,VAR,GCD,SEARCHCOND,DTRM,SMITH,RUNSMITH
'FINISH'
****
```

APPENDIX D

PROGRAM LISTINGS

POLYNOMIAL OPERATORS AND THE SMITH FORM ALGORITHM OVER R[s,z] OR R(z)[s]

```
!BEGIN!


!MODE!!POLY!=!STRUCT!([0:C!FLEX!,0:0!FLEX!]!REAL!P);
!MODE!!DEG!=[1:8]!INT!;
!POLY!DUMMY;
!DEG!D1;
!BOOL!RAT←!FALSE!;


!PROC!DF=(!REF!!DEG!D,!POLY!A);
!C!   FINDS THE DEGREE OF A POLYNOMIAL   !C!
!BEGIN!
!INT!D1←1!UPB!P!OF!A,D2←2!UPB!P!OF!A,
     L1←1!LWB!P!OF!A,L2←2!LWB!P!OF!A,I,J,LB←0,LT←0;
!BOOL! RUN←!TRUE!;
!FOR!R!FROM!D1+D2!BY!-1!TO!L1+L2!WHILE!RUN!DO!
     !BEGIN!
     I←(R>=D1+L2|D1|R-L2);J←R-I;
     !WHILE!J<=D2!AND!I>=L1!AND!RUN!DO!
          (!ABS!(P!OF!A)[I,J]>1.0&-6|RUN←!FALSE!|J!PLUS!1;I!MINUS!1)
     !END!;
RUN←!TRUE!;
!FOR!L!FROM!L2!TO!D2!WHILE!RUN!DO!
     (!ABS!(P!OF!A)[D1,L]>1.0&-6|RUN←!FALSE!;LB←L);
RUN←!TRUE!;
!FOR!L!FROM!D2!BY!-1!TO!L2!WHILE!RUN!DO!
     (!ABS!(P!OF!A)[D1,L]>1.0&-6|RUN←!FALSE!;LT←L);
(LT<LB|LT←LB);
!C!
   D[1]= UPPER BOUND OF THE FIRST VARIABLE
   D[2]= UPPER BOUND OF THE SECOND VARIABLE
   D[3]= LOWER BOUND OF THE FIRST VARIABLE
   D[4]= LOWER BOUND OF THE SECOND VARIABLE
   D[5],D[6]= POSITION OF THE LEADING COEFFICIENT
   D[7]= POSITION OF THE LEADING COEFFICIENT IF RATIONAL
   D[8]= RANGE OF THE POLYNOMIAL COEFFICIENT IN Z OF THE
         HIGHEST TERM IN S
!C!
D←(D1,D2,L1,L2,I,J,LT,LT-LB)
!END!;


!OP!E=(!POLY!A)!BOOL!;
!C!   CHECKS IF A POLYNOMIAL IS ZERO   !C!
!BEGIN!
!DEG!D;
DF(D,A);
(D[1]=0!AND!D[2]=0!AND!D[3]=0!AND!D[4]=0!AND!!ABS!(P!OF!A)[0,0]<1.0&-6)
!END!;


!OP!%=(!POLY!A)!POLY!;
!C!   TRIMS THE POLYNOMIAL A   !C!
!BEGIN!
!POLY!B,C←A;
!DEG!D:DF(D,C);
!BOOL!COL←!TRUE!,ROW←!TRUE!;
!WHILE!COL!DO!
     !BEGIN!
     !FOR!I!FROM!0!TO!D[1]!DO!
          (!ABS!(P!OF!C)[I,D[2]]>1.0&-6|COL←!FALSE!);
```

```
      (D[2]=0|COL←'FALSE');
      (COL|D[2]'MINUS'1)
      'END';
(D[4]<0|COL←'TRUE');
'WHILE'COL'DO'
      'BEGIN'
      'FOR'I'FROM'0'TO'D[1]'DO'
            ('ABS'(P'OF'C)[I,D[4]]>1.0&-6|COL←'FALSE');
      (COL|D[4]'PLUS'1);
      (D[4]=0|COL←'FALSE')
      'END';
'C'    TRIMMED THE COLUMNS    'C'
'WHILE'ROW'DO'
      'BEGIN'
      'FOR'J'FROM'D[4]'TO'D[2]'DO'
            ('ABS'(P'OF'C)[D[1],J]>1.0&-6|ROW←'FALSE');
      (D[1]=0|ROW←'FALSE');
      (ROW|D[1]'MINUS'1)
      'END';
(D[3]<0|ROW←'TRUE');
'WHILE'ROW'DO'
      'BEGIN'
      'FOR'J'FROM'D[4]'TO'D[2]'DO'
            ('ABS'(P'OF'C)[D[3],J]>1.0&-6|ROW←'FALSE');
      (ROW|D[3]'PLUS'1);
      (D[3]=0|ROW←'FALSE')
      'END';
'C'    TRIMMED THE ROWS    'C'
P'OF'B←(P'OF'C)[D[3]:D[1]'AT'D[3],D[4]:D[2]'AT'D[4]];
B
'END';


'OP''CLEAR'=('REF''POLY'A);('CLEAR'P'OF'A);


'PROC'REDIM=('REF''POLY'A,'INT'D2,L2);
'C'
   RE-DIMENSIONS A POLYNOMIAL TO ACCEPT MORE COEFFICIENTS
'C'
'BEGIN'
'DEG'D;
DF(D,A);
'POLY'B;
P'OF'B←[0:D[1],L2:D2]'REAL';
'CLEAR'B;
(P'OF'B)[,D[4]:D[2]]←(P'OF'A);
P'OF'A←[0:D[1],L2:D2]'REAL';
A←B
'END';


'OP'/=('REF''POLY'A,'POLY'B)'POLY';
'C'
   OPERATOR FOR THE DIVISION OF TWO POLYNOMIALS    A/B   DELIVERING A
   QUOTIENT POLYNOMIAL AND ALTERRING POLYNOMIAL A TO BE THE REMAINDER
'C'
'BEGIN'
'INT'LOW,DEG,I,J,H,L;
'REAL'NC;
'DEG'DA,DB,DG;
DF(DA,A);DF(DB,B);
```

```
'BOOL'RUN←'TRUE';
'POLY'Q;
'IF'      EA
     'OR' EB
     'OR' DA[1]<DB[1]
     'OR' (DA[5]+DA[6]<DB[5]+DB[6] 'AND' 'NOT'RAT)
     'OR' (DA[1]+DA[7]<DB[1]+DB[7]'AND'RAT'AND'DA[8]<DB[8])
'THEN'
     'C' - NO DIVISION REQUIRED  SO SET  Q=0  R=A   'C'
     'CLEAR'Q
'ELSE'
     LOW←DA[4]-DB[4];
     (LOW>0|LOW←0);
     DEG←DA[2]-DB[2];
     (DEG<0|DEG←0);
     DQ←(DA[1]-DB[1],DEG,0,LOW,0,0,0,0);
     P'OF'Q←[0;DQ[1],DQ[4];DQ[2]]'REALT';
     'CLEAR'Q;
     (RATIH←DB[1];L←DB[7]|H←DB[5];L←DB[6]);
     'WHILE'RUN'DO'
          'BEGIN'
          'C'  MAIN LOOP TO CARRY OUT THE LONG DIVISION   'C'
          A←%A;
          DF(DA,A);DF(DQ,Q);
          (RAT|I←DA[1];J←DA[7]|I←DA[5];J←DA[6]);
          'IF'     EA
               'OR' I<H
               'OR' (I+J<H+L'AND'('NOT'RAT'OR'(RAT'AND'DA[8]<DB[8])))
               'OR' (J<L'AND''NOT'RAT)
          'THEN'    RUN←'FALSE'
          'ELSE'
               MC←(P'OF'A)[I,J]/(P'OF'B)[H,L];
               (DA[2]-J<DB[2]-L|REDIM(A,DB[2]+J-L,DA[4]);
                               REDIM(Q,DQ[2]+DB[2]-DA[2]+J-L,DQ[4]));
               (J-DA[4]<L-DB[4]'AND'RAT|REDIM(A,DA[2],DB[4]-L+J);
                                   REDIM(Q,DQ[2],DQ[4]+DB[4]-DA[4]-L+J);
               'FOR'II'FROM'0'TO'DB[1]'DO'
                    'FOR'JJ'FROM'DB[4]'TO'DB[2]'DO'
                         (P'OF'A)[II+I-H,JJ+J-L]'MINUS'
                         MC*(P'OF'B)[II,JJ];
                         'C'   SUBTRACTS MULTIPLE OF  B  FROM  A   'C'
               (P'OF'Q)[I-H,J-L]←MC
               'C'   FORMS THE QUOTIENT POLYNOMIAL   'C'
          'FI'
          'END'
'FI';
A←%A;
%Q
'END';


'OP'*=('POLY'A,B)'POLY';
'C'   MULTIPLIES TWO POLYNOMIALS   A*B   'C'
'BEGIN'
'DEG'DA,DB;
DF(DA,A);DF(DB,B);
'POLY'PROD;
'IF' EA'OR'EB
'THEN'
     'C'   PRODUCT IS ZERO   'C'
     'CLEAR'PROD
'ELSE'
```

```
    P'OF'PROD←[DA[3]+DB[3]:DA[1]+DB[1],DA[4]+DB[4]:DA[2]+DB[2]]'REAL';
    'CLEAR'PROD;
    'FOR'I'FROM'DA[3]'TO'DA[1]'DO'
        'FOR'J'FROM'DA[4]'TO'DA[2]'DO'
            'FOR'H'FROM'DB[3]'TO'DB[1]'DO'
                'FOR'L'FROM'DB[4]'TO'DB[2]'DO'
                    (P'OF'PROD)[I+H,J+L]'PLUS'
                    (P'OF'A)[I,J]*(P'OF'B)[H,L]
'FI';
%PROD
'END';


'OP'*=('POLY'A;'REAL'MC)'POLY';
'C'  MULTIPLIES A POLYNOMIAL  A  BY A CONSTANT  MC   'C'
'BEGIN'
'POLY'B←A;
'DEG'D;DF(D,B);
'FOR'I'FROM'D[3]'TO'D[1]'DO'
    'FOR'J'FROM'D[4]'TO'D[2]'DO'
        (P'OF'B)[I,J]'TIMES'MC;
%B
'END';


'OP'+=('POLY'A;B)'POLY';
'C'   ADDS TWO POLYNOMIALS   A+B   'C'
'BEGIN'
'DEG'DA,DB;
DF(DA,A);DF(DB,B);
'INT'DEG1,DEG2,LOW1,LOW2;
DEG1←(DA[1]>DB[1]|DA[1]|DB[1]);
DEG2←(DA[2]>DB[2]|DA[2]|DB[2]);
LOW1←(DA[3]<DB[3]|DA[3]|DB[3]);
LOW2←(DA[4]<DB[4]|DA[4]|DB[4]);
'POLY'SUM;
P'OF'SUM←[LOW1:DEG1,LOW2:DEG2]'REAL';
'CLEAR'SUM;
'FOR'I'FROM'DA[3]'TO'DA[1]'DO'
    'FOR'J'FROM'DA[4]'TO'DA[2]'DO'
        (P'OF'SUM)[I,J]←(P'OF'A)[I,J];
'FOR'I'FROM'DB[3]'TO'DB[1]'DO'
    'FOR'J'FROM'DB[4]'TO'DB[2]'DO'
        (P'OF'SUM)[I,J]'PLUS'(P'OF'B)[I,J];
%SUM
'END';


'OP'-=('POLY'A)'POLY';
'C'  MONADIC MINUS   'C'
'BEGIN'
'POLY'B;
'DEG'D;
DF(D,A);
P'OF'B←[D[3]:D[1],D[4]:D[2]]'REAL';
'FOR'I'FROM'D[3]'TO'D[1]'DO'
    'FOR'J'FROM'D[4]'TO'D[2]'DO'   (P'OF'B)[I,J]←-(P'OF'A)[I,J];
%B
'END';


'OP'-=('POLY'A,B)'POLY';
```

```
'C'   SUBTRACTS TWO POLYNOMIALS   A-B   'C'
'BEGIN'
'DEC'DA,DB;
DF(DA,A);DF(DB,B);
'INT'DEG1,DEG2,LOW1,LOW2;
DEG1←(DA[1]>DB[1]|DA[1]|DB[1]);
DEG2←(DA[2]>DB[2]|DA[2]|DB[2]);
LOW1←(DA[3]<DB[3]|DA[3]|DB[3]);
LOW2←(DA[4]<DB[4]|DA[4]|DB[4]);
'POLY'DIFF;
P'OF'DIFF←[LOW1:DEG1,LOW2:DEG2]'REAL';
'CLEAR'DIFF;
'FOR'I'FROM'DA[3]'TO'DA[1]'DO'
     'FOR'J'FROM'DA[4]'TO'DA[2]'DO'
          (P'OF'DIFF)[I,J]←(P'OF'A)[I,J];
'FOR'I'FROM'DB[3]'TO'DB[1]'DO'
     'FOR'J'FROM'DB[4]'TO'DB[2]'DO'
          (P'OF'DIFF)[I,J]'MINUS'(P'OF'B)[I,J] ;
%DIFF
'END';


'OP''PLUS'=('REF''POLY'A,'POLY'B):(A←A+B);


'OP''MINUS'=('REF''POLY'A,'POLY'B):(A←A-B);


'OP''TIMES'=('REF''POLY'A,'POLY'B):(A←A*B);


'OP''TIMES'=('REF''POLY'A,'REAL'MC):(A←A*MC);


'OP''CLEAR'=('REF'[,]'POLY'PM);
'C'   CLEARS A POLYNOMIAL MATRIX   'C'
'BEGIN'
'INT'M←1'UPB'PM,N←2'UPB'PM;
'POLY'ZEROPOLY;'CLEAR'ZEROPOLY;
'FOR'I'TO'M'DO'
     'FOR'J'TO'N'DO'
          PM[I,J]←ZEROPOLY
'END';



'OP'*=([,]'POLY'K1,K2)[,]'POLY';
'C'   MULTIPLIES TWO POLYNOMIAL MATRICES   'C'
'BEGIN'
'INT'M1←1'UPB'K1,M1←2'UPB'K1,M2←1'UPB'K2,N2←2'UPB'K2;
[1:M1,1:N2]'POLY'PROD;'CLEAR'PROD;
'IF'M1=M2
'THEN'
     'FOR'I'TO'M1'DO'
          'FOR'J'TO'N2'DO'
               'FOR'K'TO'N1'DO'
                    PROD[I,J]'PLUS'(K1[I,K]*K2[K,J])
'FI';
PROD
'END';
```

```
!PROC!READPOLYMX=(!REF![,]!POLY!K);
!C!   INPUTS A POLYNOMIAL MATRIX   !C!
!BEGIN!
!INT!M=1!UPB!K,N=2!UPB!K;
!FOR!I!TO!M!DO!
     !FOR!J!TO!N!DO!
          !BEGIN!
          !INT!DEG1,DEG2;
          READ((DEG1,DEG2));
          P!OF!K[I,J]=[0:DEG1,0:DEG2]!REAL!;
          READ(P!OF!K[I,J]);
          K[I,J]=%K[I,J]
          !C!
             ENSURES THAT THE INPUTTED POLYNOMIAL IS OF LOWEST FORM
          !C!
          !END!
!END!;

!PROC!NL=(!INT!N);(!FOR!I!TO!N!DO!NEWLINE(STAND OUT));

!PROC!SP=(!INT!N);(!FOR!I!TO!N!DO!SPACE(STAND OUT));

!PROC!PRC=(!CHAR!C,!INT!N);(!FOR!I!TO!N!DO!PRINT(C)!NL(1));

!PROC!PRS=(!STRING!S);(PRINT(S);NL(1));

!PROC!PRINTPOLY=(!POLY!A);
!C!   OUTPUTS A POLYNOMIAL   !C!
!BEGIN!
!INT!II,JJ;
!DEG!D;DF(D,A);
!BOOL!RUN=!TRUE!;
II=D[4];JJ=D[4]+4;
!WHILE!RUN!DO!
     !BEGIN!
     (JJ>=D[2];JJ=D[2];RUN=!FALSE!);
     !IF! D[3]<0 !OR!D[4]<0
     !THEN!
          !BEGIN!
          !FORMAT!FT1=SL-2WV2X,N(JJ-II+1)(-D.3DXE-2WVX)S;
          NL(1);SP(5);
          !FOR!L!FROM!II!TO!JJ!DO!
               OUTF(STAND OUT,S3X-2WV6XS,L);
          NL(1);
          !FOR!L!FROM!D[3]!TO!D[1]!DO!
              OUTF(STAND OUT,FT1,(L,(P!OF!A)[L,II:JJ]))
          !END!
     !ELSE!
     OUTF(STAND OUT,SN(D[1]-1)(LN(JJ-II+1)(-D.3DXE-2WVX))S,
     (P!OF!A)[,II:JJ])
     !FI!;
     (RUN|II!PLUS!5;JJ!PLUS!5;NL(2))
     !END!;
NL(1)
!END!;

!PROC!PRINTPOLYMX=([,]!POLY!K);
```

```
'C'   OUTPUTS A POLYNOMIAL MATRIX   'C'
'BEGIN'
'INT'L=1'UPB'K,N=2'UPB'K;
'FOR'I'TO'M'DO'
     'FOR'J'TO'N'DO'
          'BEGIN'
          'IF''NOT'EK[I,J]
          'THEN'
               OUTF(STAND OUT,$L"["2SV",",2SV"]"$,(I,J));
               PRINTPOLY(K[I,J])
          'FI'
          'END';
NL(1);PRC("*",60);NL(1)
'END';


'PROC'MONIC=('REF'[,]'POLY'K,POST,'INT'R):
'C'
   DIVIDES THE PIVOTAL COLUMN TO MAKE THE PIVOT MONIC OVER R[S,Z]
   OR R(Z)[S]
'C'
'BEGIN'
'INT'M=1'UPB'K,N=2'UPB'K,J=0;
'REAL'CONST;
'POLY'MON,A=K[R,R];
'BOOL'RUN='TRUE';
'DEG'D;
DF(D,A);
(RAT'CONST=(P'OF'A)[D[1],D[7]]:J=D[7]'CONST=(P'OF'A)[D[5],D[6]]);
P'OF'MON=(J>0'[0:0]=J:0]'REAL'[0:0,0:J]'REAL');'CLEAR'P'OF'MON;
(P'OF'MON)[0,-J]=1.0/CONST:MON=%MON;
'FOR'I'TO'N'DO'
     POST[I,R]'TIMES'MON;
'FOR'I'FROM'R'TO'M'DO'
     K[I,R]'TIMES'MON
'C'
   DIVIDES COLUMN R TO MAKE PIVOT MONIC
'C'
'END';


'PROC'LSTDEG=('REF'[,]'POLY'K,PRE,POST,'INT'R,'REF''BOOL'ZERO,CHANGE):
'C'
   FINDS THE POLYNOMIAL OF LEAST DEGREE AND MOVES IT TO POSITION (R,R)
'C'
'BEGIN'
'INT'M=1'UPB'K,N=2'UPB'K,II=R,JJ=R,D1,DG,DR;
'DEG'D;
ZERO='TRUE';CHANGE='FALSE';
'FOR'I'FROM'R'TO'M'DO'
     'FOR'J'FROM'R'TO'N'DO'
          'BEGIN'
          'POLY'A=K[I,J];
          DF(D,A);
          'IF'     'NOT'EA
          'THEF'   ZERO
          'THEN'             D1=D[1];DG=(RAT'D[7]'D[5]+D[6])'DR=D[8];
                             II=I;JJ=J;ZERO='FALSE'
          'ELSF'   RAT'AND'D[1]<D1
          'THEN'             D1=D[1];DG=D[7];DR=D[8];II=I;JJ=J
          'ELSF'   RAT'AND'D[1]=D1'AND'D[7]<DG
          'THEN'             DG=D[7];DR=D[8];II=I;JJ=J
```

```
          'ELSF'  RAT'AND'D[1]=D1'AND'D[7]=DG'AND'D[8]<DR
          'THEN'          DR←D[8];II←I;JJ←J
          'ELSF'  'NOT'RAT 'AND' D[5]←D[6]<DG
          'THEN'          DG←D[5]←D[6];D1←D[1];II←I;JJ←J
          'ELSF'  'NOT'RAT 'AND' D[5]←D[6]=DG 'AND' D[1]<D1
          'THEN'          D1←D[1];II←I;JJ←J
          'FI'
          'END';
'C'
   FIRSTLY SEARCHES THE SUB-MATRIX FOR THE FIRST NON-ZERO ELEMENT
   THEN CONTINUES FROM THERE TO SEARCH FOR THE NON-ZERO ELEMENT
   OF LEAST DEGREE AT POSITION  (II,JJ)
   IF THERE ARE NO NON-ZERO ELEMENTS THE  BOOL  ZERO  WILL REMAIN TRUE
'C'
'IF' II#R
'THEN'
     [1:N]'POLY'A←K[II,];K[II,]←K[R,];K[R,]←A;
     CHANGE←'TRUE';
     [1:N]'POLY'B←PRE[II,];PRE[II,]←PRE[R,];PRE[R,]←B
'FI';
'C'   INTERCHANGES ROWS II,R   'C'
'IF'JJ#R
'THEN'
     [1:N]'POLY'A←K[,JJ];K[,JJ]←K[,R];K[,R]←A;
     CHANGE←'TRUE';
     [1:N]'POLY'B←POST[,JJ];POST[,JJ]←POST[,R];POST[,R]←B
'FI'
'C'
   INTERCHANGES COLUMNS JJ,R
   AND HAS MOVED ELEMENT OF LEAST DEGREE TO POSITION (R,R)
'C'
'END';


'PROC'GAUSS=('REF'[,]'POLY'K,PRE,POST,'INT'R);
'C'
   CARRIES OUT ONE FULL STEP OF GAUSSIAN ELIMINATION
'C'
'BEGIN'
'POLY'PIV←K[R,R],Q,PZ;
'INT'N←1'UPB'K,N←2'UPB'K,I,J;
'BOOL'CHANGE,DEGEN←'FALSE','NOREM←'TRUE','RUN←'TRUE',
      ZERO;
'DEG'D;
I←R;J←R+1;
(J>N'I←R+1;J←R);
(J=R'AND'I>N'RUN←'FALSE');
'WHILE'RUN'DO'
     'BEGIN'
     (J>N'I←R+1;J←R);
     'IF''NOT'EK[I,J]
     'THEN'
          'IF'DEGEN
          'THEN'
               DF(D,PIV);
               P'OF'PZ←[0:0,D[4]:D[2]]'REAL';
               (P'OF'PZ)[0,]←(P'OF'PIV)[D[1],];
               PZ←%PZ;
               'IF'I=R
               'THEN'
                    'FOR'II'TO'N'DO'
                         POST[II,JJ]'TIMES'PZ;
```

```
                    'FOR'II'FROM'R'TO'M'DO'
                         K[II,JJ]'TIMES'PZ
                    'C'   MULTIPLIES COLUMN J BY PZ   'C'
               'ELSE'
                    'FOR'JJ'TO'M'DO'
                         PRE[I,JJ]'TIMES'PZ;
                    'FOR'JJ'FROM'R'TO'N'DO'
                         K[I,JJ]'TIMES'PZ
                    'C'   MULTIPLIES ROW I BY PZ   'C'
               'FI'
          'FI';
          Q←K[I,JJ/PIV;
          'IF''NOT'EQ
          'THEN'
               'IF'I=R
               'THEN'
                    'FOR'II'TO'N'DO'
                         POST[II,JJ]'MINUS'(Q*POST[II,R]);
                    'FOR'II'FROM'R+1'TO'M'DO'
                         K[II,JJ]'MINUS'(Q*K[II,RJ)
                    'C'   SUBTRACTS Q TIMES COLUMN R FROM COLUMN J   'C'
               'ELSE'
                    'FOR'JJ'TO'M'DO'
                         PRE[I,JJ]'MINUS'(Q*PRE[R,JJ]);
                    'FOR'JJ'FROM'R+1'TO'N'DO'
                         K[I,JJ]'MINUS'(Q*K[R,JJ])
                    'C'   SUBTRACTS Q TIMES ROW R FROM ROW I   'C'
               'FI';
               LSTDEG(K,PRE,POST,R,ZERO,CHANGE);
               (CHANGE|I←R;J←R+1;PIV←K[R,R];NOREM←'TRUE'
                    |('NOT'£K[I,J]|NOREM←'FALSE');
                     (I=R|J'PLUS'1|I'PLUS'1))
          'ELSE'
               NOREM←'FALSE';(I=R|J'PLUS'1|I'PLUS'1)
          'FI'
     'ELSE'
          (I=R|J'PLUS'1|I'PLUS'1)
     'FI';
     (I=R'AND'J>N|J←R;I←R+1);
     (J=R'AND'I>M|NOREM|RUN←'FALSE'
                      |I←R;J←R+1;NOREM←'TRUE'|
                       DEGEN←'TRUE'|RAT←'TRUE')
     'END';
MONIC(K,POST,R)
'END';


'PROC'FACCHECK=('REF'[,]'POLY'K,POST,'INT'R,'REF''BOOL'FACTOR);
'C'
   CHECKS WHETHER THE PIVOT IS A FACTOR OF THE REMAINING SUB-MATRIX
'C'
'BEGIN'
'INT'N←1'UPB'K,N+2'UPB'K,JJ;
FACTOR←'TRUE';
'POLY'FAC←K[R,R];
'FOR'I'FROM'R+1'TO'M'WHILE'FACTOR'DO'
     'FOR'J'FROM'R+1'TO'N'WHILE'FACTOR'DO'
          'BEGIN'
          'POLY'A←K[I,J];
          'IF' 'NOT'ÉA
          'THEN'
               'BEGIN'
```

```
            'POLY'Q←A/FAC;
            'IF' 'NOT'EA
            'THEN'    FACTOR←'FALSE';JJ←J
            'FI'
            'END'
        'FI'
        'END';
'IF'  'NOT'FACTOR
'THEN'
    'FOR'I'TO'M'DO'  K[I,R]'PLUS'K[I,JJ];
    'FOR'I'TO'N'DO'  POST[I,R]'PLUS'POST[I,JJ]
'FI'
'C'   IF REMAINDER#0 ADD COL JJ TO COL R   'C'
'END';


'PROC'SMITHFORM=('REF'[,]'POLY'K,PRE,POST);
'C'   PROCEDURE WHICH FINDS THE SMITH FORM   'C'
'BEGIN'
'INT'M←1'UPB'K,N←2'UPB'K,ORD←(M>N|N|M);
'BOOL'FACTOR,RESTRT←'TRUE',CHANGE,ZERO;
RST: 'FOR'R'TO'ORD'DO'
    'BEGIN'
    FACTOR←'FALSE';
    'WHILE''NOT'FACTOR'DO'
        'BEGIN'
        LSTDEG(K,PRE,POST,R,ZERO,CHANGE);
        (ZERO|'GOTO'XIT);
        GAUSS(K,PRE,POST,R);
        (RAT'AND'RESTRT|RESTRT←'FALSE';'GOTO'RST);
        FACCHECK(K,POST,R,FACTOR)
        'END'
    'END';
XIT: (RAT|PRC("=",28);PRS("RATIONAL TERMS WERE REQUIRED");PRC("=",28);N
'END';


'PROC'DET=([,]'POLY'K)'POLY';
'C'
   FINDS THE DETERMINANT OF A POLYNOMIAL MATRIX BY EXPANSION ALONG THE
   FIRST ROW AND RECURSION FOR LOWER ORDER DETERMINANTS
'C'
'BEGIN'
'INT'N←'UPB'K;
'POLY'DTRM;'CLEAR'DTRM;
'REAL'SIGN←1.0;
'IF'        N>2
'THEN'
    'FOR'I'TO'N'DO'
        'BEGIN'
        [1:N-1,1:N-1]'POLY'SL;
        'IF'        I=1
        'THEN'          SL←K[2:N,2:N]
        'ELSF'      I=N
        'THEN'          SL←K[2:N,1:N-1]
        'ELSE'          SL[:,1:I-1]←K[2:N,1:I-1];
                        SL[:,I:N-1]←K[2:N,I+1:N]
        'FI';
        ('NOT'EK[1,I]|DTRM'PLUS'(K[1,I]*DET(SL)*SIGN));
        SIGN←-SIGN
        'END'
'ELSF'      N=1
```

```
!THEN!      DTRM←K[1,1]
!ELSE!      DTRM←K[1,1]*K[2,2]-K[1,2]*K[2,1]
!FI!;
%DTRM
!END!;


!PROC!INNER=(!REF![,]!POLY!K);
!C!
   FINDS THE INNER SQUARE OF A MATRIX
!C!
!BEGIN!
!INT!N←!UPB!K;
[1:N-2,1:N-2]!POLY!INN←K[2:N-1,2:N-1];
K←INN
!END!;


!PROC!SUBRES=(!REF![,]!POLY!K,!POLY!A,B);
!C!
   FINDS THE SUBRESULTANT OF THE BIGRADIENT MATRIX USED IN FINDING THE
   GREATEST COMMON DIVISOR
!C!
!BEGIN!
!DEG!DA,DB;DF(DA,A);DF(DB,B);
!INT!N←!UPB!K,NN←(DA[1]+DB[1]-N)!/!2;
[1:N,1:N]!POLY!SBRS←K;
!POLY!S;
P!OF!S←[0:1,0:0]!REAL!;
(P!OF!S)[,0]←(0,0,1,0);
!IF! (DB[1]-NN)>0
!THEN!
      SBRS[DB[1]-NN,N]←A;
      ((DB[1]-NN)>1|!FOR!I!FROM!DB[1]-NN-1!BY!-1!TO!1!DO!
                  SBRS[I,N]←S*SBRS[I+1,N])
!FI!;
!IF! (DA[1]-NN)>0
!THEN!
      SBRS[DB[1]-NN+1,N]←B;
      ((DA[1]-NN)>1|!FOR!I!FROM!N-DA[1]+NN+2!TO!N!DO!
                  SBRS[I,N]←S*SBRS[I-1,N])
!FI!;
K←SBRS
!END!;


!PROC!VAR=(!REF!!POLY!A);
!C!
   CHANGES THE VARIABLES OF A POLYNOMIAL AROUND  S TO Z & Z TO S
!C!
!BEGIN!
!DEG!D;DF(D,A);
!POLY!B;
P!OF!B←[D[4]:D[2],D[3]:D[1]]!REAL!;
!FOR!L!FROM!D[3]!TO!D[1]!DO!
     (P!OF!B)[,L]←(P!OF!A)[L,];
A←B
!END!;


!PROC!VARCH=(!REF![,]!POLY!K);
!C!
```

```
   CHANGES THE VARIABLES OF A POLYNOMIAL MATRIX
!C!
!BEGIN!
!INT!M←1!UPB!K;N←2!UPB!K;
!FOR!I!TO!M!DO!
     !FOR!J!TO!N!DO!
          VAR(K[I,J])
!END!;


!PROC!GCD1=(!POLY!A1,B1)!POLY!;
!C!
   FINDS THE GREATEST COMMON DIVISOR OF TWO POLYNOMIALS IN ONE
   VARIABLE USING BLANKINSHIPS ALGORITHM
!C!
!BEGIN!
!POLY!A←A1,B←B1,Q;
!DEG!DA,DB;
!WHILE! !NOT!EA !AND! !NOT!EB !DO!
     !BEGIN!
     DF(DA,A);DF(DB,B);
     Q←(DA[1]<DB[1]|B/A|A/B)
     !END!;
(EA|B|A)
!END!;


!PROC!GCD=(!POLY!A1,B1)!POLY!;
!C!
   FINDS THE GREATEST COMMON DIVISOR OF TWO POLYNOMIALS USING THE
   SYLVESTER MATRIX OR BIGRADIENT
!C!
!BEGIN!
!POLY!A←A1,B←B1;
!DEG!DA,DB,DFAC;DF(DA,A);DF(DB,B);
!INT!DIM←(DA[1]+DA[2]+DB[1]+DB[2]=0|0
          |:DA[2]+DB[2]=0!AND!DA[1]+DB[1]>0|1
          |:DA[1]+DB[1]=0!AND!DA[2]+DB[2]>0|3|2);
!BOOL!CH←!FALSE!;
(DIM=3|VAR(A);VAR(B);DF(DA,A);DF(DB,B);CH←!TRUE!;DIM=1);
!POLY!FACTOR,CONTENT,CHECK,CONTA,CONTB,DUMMY;
!IF!EA!OR!EB
!THEN!
     !C!   GREATEST COMMON DIVISOR IS ZERO   !C!
     !CLEAR!FACTOR
!ELSE!
     !IF!  DIM=2
     !THEN!
          [1:DA[1]+DB[1]!FLEX!,1:DA[1]+DB[1]!FLEX!]!POLY!BIGRAD;
          !CLEAR!BIGRAD;
          !C!
             REMOVE CONTENT OF A & B AND FIND GCD(CONT(A),CONT(B))
          !C!
          !BOOL!ZEROA←!TRUE!,ZEROB←!TRUE!;
          !FOR!I!FROM!0!TO!DA[1]!DO!
               !BEGIN!
               P!OF!CHECK←[0:0,0:DA[2]]!REAL!;
               (P!OF!CHECK)[0,]←(P!OF!A)[I,];CHECK←%CHECK;
               (!NOT!ECHECK!:ZEROA|CONTA←CHECK;ZEROA←!FALSE!
                                   |CONTA←GCD(CONTA,CHECK));
               !END!;
               !C!
```

```
          FINDS CONTENT OF A =CONTA
      'C'
'FOR'I'FROM'0'TO'DB[1]'DO'
      'BEGIN'
      P'OF'CHECK←[0:0,0:DB[2]]'REAL';
      (P'OF'CHECK)[0,]←(P'OF'B)[I,];CHECK←%CHECK;
      ('NOT'ECHECK|:ZEROB|CONTB←CHECK:ZEROB←'FALSE'
                          |CONTB←GCD(CONTB,CHECK));
      'END';
      'C'
          FINDS CONTENT OF B =CONTB
      'C'
A←A/CONTA;DF(DA,A);
B←B/CONTB;DF(DB,B);
CONTENT←GCD(CONTA,CONTB);
'C'
   SETS UP THE BIGRADIENT MATRIX
'C'
'FOR'I'TO'DB[1]'DO'
      'FOR'J'FROM'0'TO'DA[1]'DO'
           (P'OF'DUMMY←[0:0,0:DA[2]]'REAL';
           'FOR'L'FROM'0'TO'DA[2]'DO'
                (P'OF'DUMMY)[0,L]←(P'OF'A)[DA[1]-J,L];
           BIGRAD[I,I+J]←%DUMMY);
'FOR'I'TO'DA[1]'DO'
      'FOR'J'FROM'0'TO'DB[1]'DO'
           (P'OF'DUMMY←[0:0,0:DB[2]]'REAL';
           'FOR'L'FROM'0'TO'DB[2]'DO'
                (P'OF'DUMMY)[0,L]←(P'OF'B)[DB[1]-J,L];
           BIGRAD[DB[1]+I,J-I+DA[1]+1]←%DUMMY);
'IF' EDET(BIGRAD)
'THEN'
      'BEGIN'
      'C'
         REDUCES THE BIGRADIENT TO FIND THE SUB-RESULTANT
         AND SO THE GREATEST COMMON DIVISOR
      'C'
      'BOOL'RUN←'TRUE',EQUAL←'FALSE';
      'WHILE'RUN'DO'
            'BEGIN'
            'POLY'DTRM;
            'IF'  'UPB'BIGRAD=2
            'THEN'
                 RUN←'FALSE';DTRM←DET(BIGRAD);
                 (EDTRM|EQUAL←'TRUE')
            'ELSE'
                 INNER(BIGRAD);DTRM←DET(BIGRAD);
                 ('NOT'EDTRM|RUN←'FALSE')
            'FI'
            'END';
      SUBRES(BIGRAD,A,B);
      FACTOR←DET(BIGRAD);
      (EQUAL|FACTOR←A);
      'C'   REMOVE CONTENT OF FACTOR    'C'
      DF(DFAC,FACTOR);ZEROA←'TRUE';
      'FOR'I'FROM'0'TO'DFAC[1]'DO'
           'BEGIN'
           P'OF'CHECK←[0:0,0:DFAC[2]]'REAL';
           (P'OF'CHECK)[0,]←(P'OF'FACTOR)[I,];
           CHECK←%CHECK;
           ('NOT'ECHECK|:ZEROA|CONTA←CHECK;ZEROA←'FALSE'
                              |CONTA←GCD(CONTA,CHECK))
```

```
                    'END';
                FACTOR←FACTOR/CONTA
                'END'
            'ELSE'
                !C!   COMMON FACTOR IS UNITY   !C!
                (P!OF!FACTOR)[0,0]←1.0
            'FI'
        'ELSE'
            FACTOR←(DIM=0|A|GCD1(A,B))
        'FI';
        (DIM=2|FACTOR'TIMES'CONTENT);
        (CH|VAR(FACTOR));
        DF(DFAC,FACTOR);
        FACTOR'TIMES'(1.0/(P'OF'FACTOR)[DFAC[3],DFAC[4]])
!FI';
%FACTOR
!END';


!PROC'SMITH=('REF'[,]'POLY'K);
!C!
   MAIN PROCEDURE TO FIND THE SMITH FORM OF A POLYNOMIAL
   MATRIX  K  OF DIMENSIONS  M*N
!C!
!BEGIN!
!INT'M←1'UPB'K,N←2'UPB'K,ORD←(N<N|M|R),LA1←0,LB1←0;
[1:M,1:N]'POLY'K1←K,K2←K,K3←K;
[1:M,1:M]'POLY'PRE,PRE1;'CLEAR'PRE;
[1:N,1:N]'POLY'POST,POST1;'CLEAR'POST;
!POLY'UNIT;
(P!OF'UNIT)[0,0]←1.0;
!FOR'I'TO'M'DO'  PRE[I,I]←UNIT;
!FOR'I'TO'N'DO'  POST[I,I]←UNIT;
PRE1←PRE;POST1←POST;
VARCH(K1);
!BOOL'RAT←'FALSE!;
SMITHFORM(K,PRE,POST);
!IF'  RAT
!THEN!
      'BEGIN'
      PRS("PRE-MULTIPLYING EQUIVALENCE MATRIX");PRC("=",34);
      PRINTPOLYMX(PRE);
      PRS("DETERMINANT(PRE)");PRC("=",16);
      PRINTPOLY(DET(PRE));PRC("+",60);NL(1);
      PRS("POST-MULTIPLYING EQUIVALENCE MATRIX");PRC("=",35);
      PRINTPOLYMX(POST);
      PRS("DETERMINANT(POST)");PRC("=",17);
      PRINTPOLY(DET(POST));PRC("+",60);NL(1);
      PRS("SMITH FORM OVER R(Z)[S]");PRC("=",23);
      PRINTPOLYMX(K);
      PRS("PRE * K * POST");PRC("=",14);
      K3←PRE*K2*POST;
      PRINTPOLYMX(K3);
      NL(4);PRC("=",34);PRS("SEARCH FOR SMITH FORM OVER R(S)[Z]");
      PRC("=",34);NL(2);
      RAT←'FALSE';
      SMITHFORM(K1,PRE1,POST1);
      VARCH(PRE1);VARCH(POST1);VARCH(K1);
      PRS("PRE-MULTIPLYING EQUIVALENCE MATRIX");PRC("=",34);
      PRINTPOLYMX(PRE1);
      PRS("DETERMINANT(PRE)");PRC("=",16);
      PRINTPOLY(DET(PRE1));PRC("+",60);NL(1);
```

```
    PRS("POST-MULTIPLYING EQUIVALENCE MATRIX");PRC("-",35);
    PRINTPOLYMX(POST1);
    PRS("DETERMINANT(POST)");PRC("-",17);
    PRINTPOLY(DET(POST1));PRC("+",60);NL(1);
    'IF' RAT
    'THEN'
    PRS("SMITH FORM OVER R(S)[Z]");PRC("=",23)
    'ELSE'
    PRS("CORRECT SMITH FORM OVER R[S,Z]");PRC("=",30)
    'FI';
    PRINTPOLYMX(K1);
    PRS("PRE * K * POST");PRC("-",14);
    K3←PRE1*K2*POST1;
    PRINTPOLYMX(K3);
    'FOR'I'TO'ORD'WHILE'RAT'DO'
        'BEGIN'
        'C'
           PRE-NORMALIZES BOTH SMITH FORMS OVER R(Z)[S] & R(S)[Z]
           TO SMITH FORMS OVER R[Z][S] & R[S][Z] AND TAKES THE
           GCD OF THEM TO GIVE THE SMITH FORM OVER R[S,Z]
        'C'
        'POLY'A,B;
        'INT'L1,L2,LA2←0,LB2←0;
        'FOR'J'TO'M'DO'
            'BEGIN'
            L2←2'LWB'P'OF'PRE[I,J];
            L1←1'LWB'P'OF'PRE1[I,J];
            (L2<LA2|LA2←L2);
            (L1<LB2|LB2←L1)
            'END';
        LA1'PLUS'LA2;LA2←0;
        LB1'PLUS'LB2;LB2←0;
        'FOR'J'TO'N'DO'
            'BEGIN'
            L2←2'LWB'P'OF'POST[J,I];L1←1'LWB'P'OF'POST1[J,I];
            (L2<LA2|LA2←L2);
            (L1<LB2|LB2←L1)
            'END';
        LA1'PLUS'LA2;LB1'PLUS'LB2;
        P'OF'A←[0:0,0:-LA1]'REAL';'CLEAR'P'OF'A;(P'OF'A)[0,-LA1]←1.0;
        P'OF'B←[0:-LB1,0:0]'REAL';'CLEAR'P'OF'B;(P'OF'B)[-LB1,0]←1.0;
        A←A*K[I,I];
        B←B*K1[I,I];
        K[I,I]←GCD(A,B)
        'END';
    'IF'RAT
    'THEN'
    PRS("CALCULATED SMITH FORM OVER R[S,Z]");PRC("=",33);
    PRINTPOLYMX(K)
    'FI'
    'END'
'ELSE'
    PRS("PRE-MULTIPLYING EQUIVALENCE MATRIX");PRC("-",34);
    PRINTPOLYMX(PRE);
    PRS("DETERMINANT(PRE)");PRC("-",16);
    PRINTPOLY(DET(PRE));PRC("+",60);NL(1);
    PRS("POST-MULTIPLYING EQUIVALENCE MATRIX");PRC("-",35);
    PRINTPOLYMX(POST);
    PRS("DETERMINANT(POST)");PRC("-",17);
    PRINTPOLY(DET(POST));PRC("+",60);NL(1);
    PRS("CORRECT SMITH FORM OVER R[S,Z]");PRC("=",30);
    PRINTPOLYMX(K);
```

```
      K3←PRE*K2*POST!
      PRS("PRE * K * POST"):PRC("-",14):
      PRINTPOLYMX(K3)
!FI!
!END!;


!PROC!RUNSMITH='VOID';
!C!
   RUNS THE SMITH FORM ALGORITHM
   READING IN THE POLYNOMIAL MATRIX  K  OF DIMENSIONS  M*N
!C!
!BEGIN!
!INT!M,N;
!C!
   READ IN THE POLYNOMIAL MATRIX  K
!C!
READ((M,N));
[1:M,1:N]!POLY!K;
READPOLYMX(K);
SP(13);PRS("SSSSS M   M III TTTTT H   H BBBB");
SP(13);PRS("S     MM MM  I     T   H   H B   B");
SP(13);PRS("SSSSS M M M  I     T   HHHHH BBBB");
SP(13);PRS("    S M   M  I     T   H   H B   B");
SP(13);PRS("SSSSS M   M III    T   H   H BBBB");
NL(3);PRC("-",20);PRS("TOLERANCE SET=1.0&-6");PRC("-",20);
NL(2);PRC("=",37);PRS("INITIAL POLYNOMIAL MATRIX OVER R[S,Z]");PRC("=",
PRINTPOLYMX(K);
SMITH(K);
NL(5);PRC("B",60);PRC("S",60)
!END!;


NL(1)
!END!
!KEEP!'POLY','DEG',RAT,DF,E,%,REDIM,/,*,+,-,'PLUS','MINUS','TIMES','CLE
READPOLYMX,NL,SP,PRC,PRS,PRINTPOLY,PRINTPOLYMX,MONIC,LSTDEG,GAUSS,
FACCHECK,SMITHFORM,DET,INNER,SUBRES,VAR,VARCH,GCD,SMITH,RUNSMITH
!FINISH!
```

APPENDIX E

PROGRAM LISTINGS

THE SMITH MCMILLAN FORM ALGORITHM

```
BEGIN'


PROC'SMMC='VOID';
C'
  PROCEDURE TO PRODUCE THE SMITH-MCMILLAN FORM OF A RATIONAL
  MATRIX INPUTTED AS TWO POLYNOMIAL MATRICES  NUM  &  DEN
  THE LEAST COMMON DENOMINATOR WILL BE FOUND AND THEN  NUM
  REFORMED ACCORDINGLY
C'
BEGIN'
INT'M,N;
C'
  INPUT THE TWO POLNOMIAL MATRICES  NUM  &  DEN  OF SIZE  M*N
C'
EAD((M,N));
1:M,1:N]'POLY'NUM,DEN;
EADPOLYMX(NUM);READPOLYMX(DEN);
P(4);PRS("M    M    CCC M    M III L      L        A    N    N");
P(4);PRS("MM  MM   C    MM  MM  I  L      L       A A   NN   N");
P(4);PRS("M M M M  C    M M M  I  L      L      AAAAA  N N  N");
P(4);PRS("M    M   C    M    M  I  L      L      A   A  N  NN");
P(4);PRS("M    M    CCC M    M III LLLLL  LLLLL  A   A  N   N");
RC("=",24);PRS("INITIAL NUMERATOR MATRIX");PRC("=",24);PRINTPOLYMX(NUM)
RC("=",26);PRS("INITIAL DENOMINATOR MATRIX");PRC("=",26);
RINTPOLYMX(DEN);
POLY'LCM,CH,A;
BOOL'STRT←'FALSE';
C'   EVALUATES THE LEAST COMMON MULTIPLE OF THE ELEMENTS OF DEN   'C'
FOR'I'TO'M'DO'
    'FOR'J'TO'N'DO'
        'BEGIN'
        'IF'STRT
        'THEN'
            CH←LCM*DEN[I,J];
            LCM←CH/GCD(LCM,DEN[I,J]);
            ('NOT'ECH|PRC("*",24);PRS("ERROR IN CALCULATING LCM");
                PRC("*",24))
        'ELSE'
            LCM←DEN[I,J];STRT←'TRUE'
        'FI'
        'END';
C'   FORM THE NEW NUMERATOR MATRIX   'C'
FOR'I'TO'M'DO'
    'FOR'J'TO'N'DO'
        'BEGIN'
        CH←NUM[I,J]*LCM;
        NUM[I,J]←CH/DEN[I,J];
        ('NOT'ECH|PRC("*",40);
                PRS("ERROR IN EVALUATING NEW NUMERATOR MATRIX");
                PRC("*",40))
        'END';
L(3);PRS("LEAST COMMON DENOMINATOR");PRC("-",24);PRINTPOLY(LCM);PRC("+"
L(3);PRS("NEW NUMERATOR MATRIX");PRC("=",20);PRINTPOLYMX(NUM);
L(3);PRC("-",53);PRS("SEARCH TO FIND THE SMITH FORM OF THE NUMERATOR MA
RC("-",53);NL(2);
MITH(NUM);
L(4);PRS("SMITH-MCMILLAN FORM");PRC("=",19);
AT←'FALSE';
CLEAR'DEN;
FOR'I'TO'(M<N|M|N)'DO'
    'BEGIN'
```

```
    A←LCM;
    CH←GCD(NUM[I,I],LCM);
    NUM[I,I]←NUM[I,I]/CH;
    DEN[I,I]←A/CH
    'END';
S("NUMERATOR");PRC("-",6);PRINTPOLYMX(NUM);
S("DENOMINATOR");PRC("-",11);PRINTPOLYMX(DEN);
(5);PRC("M",60)
ND';


(1)
ND'
EEP'SMIC
INISH'
```

APPENDIX F

PROGRAM LISTINGS

THE REALIZATION ALGORITHM

```
'BEGIN'


'PROC'GAUSS=('REF'[,]'POLY'K,PREN,PRED,'INT'R);
'C'
   CARRIES OUT ONE FULL STEP OF GAUSSIAN ELIMINATION WITH PARTIAL PIVOT
'C'
'BEGIN'
'POLY'PIV←K[R,R],Q,PZ,PZ1;
'INT'M←1'UPB'K,N←2'UPB'K,I;
'BOOL'CHANGE,DEGEN←'FALSE',NOREM←'TRUE',RUN←'TRUE',ZERO;
'DEG'D,DI;
I←R+1;
(I>M'RUN←'FALSE');
'C'   THE FORWARD ELIMINATION   'C'
MONIC(K,PREN,R);
'WHILE'RUN'DO'
     'BEGIN'
     'IF''NOT'EK[I,R]
     'THEN'
          'IF'DEGEN
          'THEN'
               DF(D,PIV);DF(DI,K[I,R]);
               P'OF'PZ←[0:0,0:D[2]]'REAL';
               P'OF'PZ1←[0:0,0:DI[2]]'REAL';
               (P'OF'PZ)[0,]←(P'OF'PIV)[D[1],];PZ←%PZ;
               (P'OF'PZ1)[0,]←(P'OF'K[I,R])[DI[1],];PZ1←%PZ1;
               PZ1←GCD(PZ,PZ1);
               PZ←PZ/PZ1;
               'FOR'JJ'TO'M'DO'
                    (NOT'EPREN[JJ,I]|PRED[JJ,I]'TIMES'PZ);
               'FOR'JJ'FROM'R'TO'N'DO'
                    K[I,JJ]'TIMES'PZ
               'C'   MULTIPLIES ROW I BY PZ   'C'
          'FI';
          Q←K[I,R]/PIV;
          'IF''NOT'EQ
          'THEN'
               'FOR'JJ'TO'M'DO'
                    'IF'RAT
                    'THEN'
                         'POLY'FR←PRED[JJ,R],FI←PRED[JJ,I],G←GCD(FR,FI
                         FR←FR/G;
                         FI←FI/G;
                         PRED[JJ,R]'TIMES'FI;
                         PREN[JJ,R]←PREN[JJ,R]*FI-Q*PREN[JJ,I]*FR
                    'ELSE'
                         PREN[JJ,R]'PLUS'(Q*PREN[JJ,I])
                    'FI';
               'FOR'JJ'FROM'R+1'TO'N'DO'
                    K[I,JJ]'MINUS'(Q*K[R,JJ]);
               'C'   SUBTRACTS Q TIMES ROW R FROM ROW I   'C'
               LSTCOLDEG(K,PREN,PRED,R,ZERO,CHANGE);
               (CHANGE|I←R+1;MONIC(K,PREN,R);PIV←K[R,R];NOREM←'TRUE'
                    |('NOT'EK[I,R][NOREM←'FALSE')|I'PLUS'1)
          'ELSE'
               NOREM←'FALSE';I'PLUS'T
          'FI'
     'ELSE'
          I'PLUS'1
     'FI';
     (I>M|NOREM|RUN←'FALSE'|I←R+1;NOREM←'TRUE';DEGEN←'TRUE';RAT←'TRUE
```

```
                                    LSTCOLDEG(K,PREN,PRED,R,ZERO,CHANGE);
                                    (CHANGE|MONIC(K,PREN,R);PIV←K[R,R]))
     'END';
I←1;
(I<R|RUN←'TRUE');
DF(D,PIV);
P'OF'PZ←[0:0,0:D[5]]'REAL';
'FOR'JJ'FROM'0'TO'D[5]'DO'
     (P'OF'PZ)[0,JJ]←(P'OF'PIV)[D[1],JJ];
'C'   THE BACK ELIMINATION    'C'
'WHILE'RUN'DO'
     'BEGIN'
     DF(DI,K[I,R]);
     'IF''NOT'EK[I,R]'AND'DI[6]>=D[6]
     'THEN'
          'IF'RAT
          'THEN'
               'FOR'JJ'TO'M'DO'
                    ('NOT'EPREN[JJ,I]|PRED[JJ,I]'TIMES'PZ);
               'FOR'JJ'TO'N'DO'
                    K[I,JJ]'TIMES'PZ
               'C'    MULTIPLIES ROW I BY PZ    'C'
          'FI';
          Q←K[I,R]/PIV;
          'IF''NOT'EQ
          'THEN'
               'FOR'JJ'TO'M'DO'
                    'IF'FAT
                    'THEN'
                         'POLY'FR←PRED[JJ,R],FI←PRED[JJ,I],G←GCD(FR,FI)
                         FR←FR/G;
                         FI←FI/G;
                         PRED[JJ,R]'TIMES'FI;
                         PREN[JJ,R]←PREN[JJ,R]*FI←Q*PREN[JJ,I]*FR
                    'ELSE'
                         PREN[JJ,R]'PLUS'(Q*PREN[JJ,I])
                    'FI';
               'FOR'JJ'FROM'R←1'TO'N'DO'
                    K[I,JJ]'MINUS'(Q*K[R,JJ]);
               'C'    SUBTRACTS Q TIMES ROW R FROM ROW I    'C'
               DF(DI,K[I,R]);
               ('NOT'EK[I,R]'AND'DI[6]>=D[6]|NOREM←'FALSE');
               I'PLUS'1
          'ELSE'
               NOREM←'FALSE';I'PLUS'1
          'FI'
     'ELSE'
          I'PLUS'1
     'FI';
     (I>=R|NOREM|RUN←'FALSE'|I←1;NOREM←'TRUE'|RAT←'TRUE');
     'END'
'END';


'PROC'GCRD=('REF'[,]'POLY'K)[,]'POLY';
'C'
   MAIN PROCEDURE TO FIND THE GCRD OF TWO INPUTTED POLYNOMIAL
   MATRICES  A,B  ADJOINTED IN  K  OF DIMENSIONS  M*N
'C'
'BEGIN'
'INT'I←1'UPB'K,N←2'UPB'K,ORD←(M<N|M|N);
[1:M,1:N]'POLY'PREN,PRED;'CLEAR'PREN;'CLEAR'PRED;
```

```
!POLY'UNIT;
(P'OF'UNIT)[0,0]←1.0;
!FOR'I'TO'M'DO'
     'BEGIN'
     PREN[I,I]←UNIT;
     'FOR'J'TO'M'DO'
          PREN[I,J]←UNIT
     'END';
!BOOL'CHANGE,ZERO,RESTRT←'TRUE';
RST; 'FOR'R'TO'ORD'DO'
     'BEGIN'
     LSTCOLDEG(K,PREN,PRED,R,ZERO,CHANGE);
     'IF''NOT'ZERO
     'THEN'
          GAUSS(K,PREN,PRED,R);
          (RAT'AND'RESTRT;RESTRT←'FALSE';'GOTO'RST)
     'FI'
     'END';
(RAT'PRC("=",37);PRS("FULL POLYNOMIAL EQUIVALENCE NOT FOUND");PRC("=",3
     NL(3));
PRS("EQUIVALENCE MATRIX");PRC("=",18);NL(2);
'IF' RAT
'THEN'
     'FOR'I'TO'M'DO'
          'FOR'J'TO'M'DO'
               'BEGIN'
               'C'
                  MAKE THE ELEMENTS OF PREN AND PRED RELATIVELY PRI
               'C'
               'POLY'FAC←GCD(PRED[I,J],PREN[I,J]);
               'DEG'D;DF(D,PRED[I,J]);
               'REAL'MC←(P'OF'PRED[I,J])[D[1],D[5]];
               FAC'TIMES'MC;
               PRED[I,J]←PRED[I,J]/FAC;
               PREN[I,J]←PREN[I,J]/FAC
               'END';
     RAT←'FALSE';
     'FOR'R'TO'ORD'DO'
          'BEGIN'
          'POLY'LCM←LCMC(PRED,R),A;
          'FOR'I'TO'M'DO'
               'BEGIN'
               'C'   RE-NORMALIZES  PREN   'C'
               A←LCM;
               PREN[I,R]'TIMES'(A/PRED[I,R])
               'END';
          'FOR'J'TO'ORD'DO'
               'BEGIN'
               'C'   RE-NORMALIZES  GCRD   'C'
               A←K[R,J];
               K[R,J]←A/LCM;
               ('NOT'EA!PRS("RE-NORMALIZATION NOT COMPLETE");PRC("=",29
                         NL(2))
               'END'
          'END';
     PRS("AFTER RE-NORMALIZATION");PRC("-",22);NL(2)
'FI';
PRINTPOLYMX(PREN);
PRC("=",29);PRS("GREATEST COMMON RIGHT DIVISOR");PRC("=",29);PRINTPOLYM
PRS("PRE * GCRD");PRC("-",10);PRINTPOLYMX((PREN*K));
PREN
'END';
```

```
!PROC!ADJ=([,]!POLY!K)[,]!POLY!;
!C!   FINDS THE ADJOINT OF A MATRIX  K(S,Z)   !C!
!BEGIN!
!INT!N←!UPB!K;
[1:N,1:N]!POLY!K1;
[1:N-1,1:N-1]!POLY!MIN;
!REAL!SIGN←1.0;
!FOR!I!TO!N!DO!
     !FOR!J!TO!N!DO!
          !BEGIN!
          SIGN←(!ODD!(I+J)|-1.0|1.0);
          !C!
             FORM THE (N-1)TH ORDER MINOR CORRESPONDING TO ELEMENT  (I,
          !C!
          (I#1!AND!J#1|MIN[1:I-1,1:J-1]←K[1:I-1,1:J-1]);
          (I#1!AND!J#N|MIN[1:I-1,J:N-1]←K[1:I-1,J+1:N]);
          (I#N!AND!J#1|MIN[I:N-1,1:J-1]←K[I+1:N,1:J-1]);
          (I#N!AND!J#N|MIN[I:N-1,J:N-1]←K[I+1:N,J+1:N]);
          K1[J,I]←DET(MIN)*SIGN
          !END!;
K1
!END!;


!PROC!REALIZ=(!REF![,]!POLY!TFN,TFD,W)[,]!POLY!;
!C!
   THE REALIZATION ALGORITHM WHICH OPERATES ON A RATIONAL TRANSFER FUNC
   MATRIX WHICH IS INPUTTED IN TWO PARTS.  THE PROPER PART AS A NUMERAT
   AND A DENOMINATOR MATRIX  TRN,TRD  AND THE POLYNOMIAL PART  W  AND C
   THE MINIMAL REALIZATION (POLYNOMIAL OR STATE-SPACE)
!C!
!BEGIN!
!INT!M←1!UPB!TFN,N←2!UPB!TFN,ORD←0,NN,II,POSN;
!POLY!LCM,A,UNIT,DTRM,CH;
!BEG!D;
[1:M,1:N]!POLY!V;!CLEAR!V;
[1:N,1:N]!POLY!T;!CLEAR!T;
!BOOL!STATESPACE←!TRUE!;
!C!
   FORM THE FIRST PART OF THE REALIZATION  V(T INVERSE)
!C!
!FOR!J!TO!N!DO!
     !BEGIN!
     LCM←LCMC(TFD,J);
     T[J,J]←LCM;
     !FOR!I!TO!M!DO!
          (A←LCM;V[I,J]←TFN[I,J]*(A/TFD[I,J]));
     !END!;
PRS("INITIAL REALIZATION");PRC("=",19);NL(2);
PRS("V");PRC("=",1);PRINTPOLYMX(V);
PRS("T");PRC("=",1);PRINTPOLYMX(T);
[1:M+N,1:N]!POLY!TV;
TV[1:N,]←T;TV[N+1:M+N,]←V;
!C!
   REMOVES THE GCRD OF  T  AND  V
!C!
PRS("FIND THE GCRD OF T AND V");PRC("=",24);NL(2);
[1:M+N,1:M+N]!POLY!TVP←GCRD(TV);
T←TVP[1:N,1:N];
V←TVP[N+1:N+M,1:N];
```

```
[1:M+N,1:2*N]'FLEX';1;2*N]'FLEX']'POLY'RZ;'CLEAR'RZ;
RZ[1:M,1:N]←T;
RZ[N+1:M+N,N+1:2*N]←W;
(P'OF'UNIT)[0,0]←1.0;
'FOR'J'TO'N'DO'
     'BEGIN'
     RZ[J,N+J]←UNIT;
     'FOR'I'TO'M'DO'
          RZ[N+I,J]←-V[I,J]
     'END';
PRS("POLYNOMIAL REALIZATION");PRC("=",22);PRINTPOLYMX(RZ);
'C'
   CHECK IF ALL THE DIAGONAL ELEMENTS OF  T   ARE MONIC OVER R[Z][S]
   AND SO SEE IF STATE-SPACE REALIZATION IS POSSIBLE
'C'
'FOR'J'TO'N'WHILE'STATESPACE'DO'
     (DF(0,RZ[J,J]);(DF5]#0|STATESPACE←'FALSE'));
'IF' 'NOT'STATESPACE
'THEN'
     DTRM←DET(T);
     TFN←V*ADJ(T);
     'CLEAR'TFD;
     'FOR'I'TO'M'DO'
          'FOR'J'TO'N'DO'
               'BEGIN'
               CH←DTRM;
               A←GCD(CH,TFN[I,J]);
               TFN[I,J]←TFN[I,J]/A;
               TFD[I,J]←CH/A
               'END';
     PRS("CORRESPONDING TRANSFER FUNCTION MATRIX");PRC("=",38);
     PRS("NUMERATOR");PRC("=",9);PRINTPOLYMX(TFN);
     PRS("DENOMINATOR");PRC("=",11);PRINTPOLYMX(TFD);
     PRC("=",36);PRS("STATE-SPACE REALIZATION NOT POSSIBLE");PRC("=",36
'ELSE'
     'C'
       SUBTRACT MULTIPLES OF ROWS OF RZ SUCH THAT ALL ELEMENTS IN THE
       COLUMNS OF  T  AND  -V  HAVE DEGREE LOWER THAN THAT OF THE
       CORRESPONDING DIAGONAL ELEMENT OF  T
     'C'
     'POLY'S;
     'FOR'J'TO'N'DO'
          'FOR'I'TO'M+N'DO'
               'BEGIN'
               'IF'  I#J
               'THEN'
                    'IF'  'NOT'ERZ[I,J]
                    'THEN'
                         'POLY'Q←RZ[I,J]/RZ[J,J];
                         'FOR'L'FROM'J+1'TO'2*N'DO'
                              ('NOT'ERZ[J,L]|RZ[I,L]'MINUS'(Q*RZ[J,L]))
                    'FI'
               'FI'
               'END';
     'C'
       NOW REMOVE ROWS AND COLUMNS FOR WHICH THE CORRESPONDING DIAGONA
       ELEMENT IN T IS CONSTANT
     'C'
     II←1;NN←0;
     'WHILE'  II<=N-NN  'DO'
          'BEGIN'
          DF(0,RZ[II,II]);
```

```
        'IF' D[1]=0'AND'D[2]=0
        'THEN'
             [1:M+N-NN-1,1:2*N-NN-1]'POLY'RZ1;
             RZ1[1:II-1,1:II-1]←RZ[1:II-1,1:II-1];
             RZ1[1:II-1,II:2*N-NN-1]←RZ[1:II-1,II+1:2*N-NN];
             RZ1[II:M+N-NN-1,1:II-1]←RZ[II+1:M+N-NN,1:II-1];
             RZ1[II:M+N-NN-1,II:2*N-NN-1]←RZ[II+1:M+N-NN,II+1:2*N-NN]
             'C'   DELETES ROW  II  AND COLUMN  II   'C'
             NN'PLUS'1;
             RZ←RZ1
        'ELSE'
             II'PLUS'1
        'FI'
        'END';
   'C'
      DIMENSIONS ARE NOW
      T = (N-NN)*(N-NN)
      U = (N-NN)*N
      V = M*(N-NN)
      W = M*N
      THE FINAL ORDER OF THE STATE SPACE REALIZATION IS
      A = ORD*ORD
      B = ORD*N
      C = M*ORD
      D = M*N
      NOW INCREASE THE DIMENSIONS OF  T,U,V  AND FORM  T=SI-A(Z)  BY
      COMPANION FORM EXPANSION OF THE DIAGONAL ELEMENTS OF  T
   'C'
   PRS("MODIFIED POLYNOMIAL REALIZATION");PRC("=",31);PRINTPOLYMX(RZ)
   [1:N-NN]'INT'POS;
   'FOR'I'TO'N-NN'DO'
        (DF(D,RZ[I,I]);ORD'PLUS'D[1];POS[I]←ORD);
   PRSI("ORDER OF REALIZATION=",ORD);PRC("=",24);NL(2);
   POSN←ORD;
   [1:ORD+M,1:ORD+N]'POLY'SSR;'CLEAR'SSR;
   'C'   SET  D(S,Z)   'C'
   SSR[ORD+1:ORD+M,ORD+1:ORD+N]←RZ[N-NN+1:M+N-NN,N-NN+1:2*N-NN];
   'FOR'I'FROM'N-NN'BY'-1'TO'1'DO'
        'BEGIN'
        P'OF'S←[0:1,0:0]'REAL';(P'OF'S)[0;0]←(0.0,1.0);
        'C'   SET  B(S,Z)   'C'
        SSR[ORD,POSN+1:POSN+N]←RZ[I,N-NN+1:2*N-NN];
        DF(D,RZ[I,I]);
        'C'   SET  SI-A(Z)   'C'
        'FOR'J'TO'D[1]'DO'
             'BEGIN'
             'POLY'A;
             SSR[ORD-J+1,ORD-J+1]←S;
             P'OF'A←[0:0,0:D[2]]'REAL';
             (P'OF'A)[0;]←(P'OF'RZ[I,I])[D[1]-J,];
             SSR[ORD,ORD-J+1]'PLUST'A
             'END';
        'FOR'J'TO'D[1]-1'DO'
             SSR[ORD-J,ORD-J+1]←-UNIT;
        ORD'MINUS'D[1];
        'FOR'II'TO'I-1'DO'
             'IF'  'NOT'ERZ[II,I]
             'THEN'
                  DF(D,RZ[II,II]);
                  'FOR'J'FROM'0'TO'D[1]'DO'
                       'BEGIN'
                       'POLY'A;
```

```
                    P'OF'A←[0:0,0:D[2]]'REAL';
                    (P'OF'A)[0,]←(P'OF'RZ[II,I])[J,];
                    SSR[POS[II],ORD+1+J]←%A
                    'END'
          'FI';
     'C'   SET  -C(Z)   'C'
     'FOR'II'FROM'1'TO'N'DO'
          'IF'  'NOT'ERZ[N-NN+II,I]
          'THEN'
               DF(D,RZ[N-NN+II,I]);
               'FOR'J'FROM'0'TO'D[1]'DO'
                    'BEGIN'
                    'POLY'A;
                    P'OF'A←[0:0,0:D[2]]'REAL';
                    (P'OF'A)[0,]←(P'OF'RZ[N-NN+II,I])[J,];
                    SSR[POSN+II,ORD+1+J]←%A
                    'END'
          'FI'
     'END';
'C'   TRANSFORM   B(S,Z)  TO  B(Z)   'C'
'FOR'J'FROM'POSN+N'BY'-1'TO'POSN+1'DO'
     'FOR'I'FROM'POSN'BY'-1'TO'1'DO'
          'BEGIN'
          DF(D,SSR[I,J]);
          'IF'  D[1]#0
          'THEN'
               'POLY'Q←SSR[I,J]/SSR[I,I];
               'FOR'II'TO'POSN+M'DO'
                    (II#I 'AND' 'NOT'ESSR[II,I]|SSR[II,J]'MINUS'
                                                (Q*SSR[II,I]))
          'FI'
          'END';
PRC("=",23);PRS("STATE-SPACE REALIZATION");PRC("=",23);PRINTPOLYMX
DTRM←DET(SSR[1:POSN,1:POSN]);
TFN←SSR[POSN+1:POSN+M,1:POSN]*ADJ(SSR[1:POSN,1:POSN])*
     SSR[1:POSN,POSN+1:POSN+N];
'FOR'I'TO'M'DO'
     'FOR'J'TO'N'DO'
          'BEGIN'
          TFN[I,J]'TIMES'←1.0;
          CH←DTRM;
          A←GCD(CH,TFN[I,J]);
          TFN[I,J]←TFN[I,J]/A;
          TFD[I,J]←CH/A
          'END';
PRS("CORRESPONDING TRANSFER FUNCTION MATRIX");PRC("=",38);
PRS("NUMERATOR");PRC("=",9);PRINTPOLYMX(TFN);
PRS("DENOMINATOR");PRC("=",11);PRINTPOLYMX(TFD)
RZ←SSR
'FI';
RZ
'END';


'PROC'RUNREALIZ='VOID';
'C'
   RUNS THE REALIZATION ALGORITHM INPUTTING THE TRANSFER FUNCTION MATR
   IN ITS PROPER AND POLYNOMIAL PART
   IF A BOOLEAN  PROPER  IS INPUTTED AS FALSE AT THE START OF DATA THE
   TRANSFER FUNCTION MATRIX IS ASSUMED TO HAVE A POLYNOMIAL PART
'C'
'BEGIN'
```

```
!INTEGER N;
!BOOL!PROPER;
READ(PROPER);
!C!   IS THE TRANSFER FUNCTION MATRIX PROPER   !C!
READ((M,N));
[1:M,1:N]!POLY!TFN,TFD,W;!CLEAR!W;
READPOLYMX(TFN);
READPOLYMX(TFD);
(!NOT!PROPER!READPOLYMX(U));
PRS("RRRR  EEEEE   A   L      III ZZZZZ   A   TTTTT III  OOO  N   N");
PRS("R   R E      A A  L       I     Z   A A    T    I  O   O NN  N");
PRS("RRRR  EEE   AAAAA L       I    Z   AAAAA   T    I  O   O N N N");
PRS("R  R  E     A   A L       I   Z    A   A   T    I  O   O N  NN");
PRS("R   R EEEEE A   A LLLLL III ZZZZZ A   A   T   III  OOO  N   N");
PRC("=",61);PRC("=",61);NL(5);PRC("*",60);
NL(3);(PROPER!PRS("THE TRANSFER FUNCTION MATRIX IS PROPER");PRC("=",38)
PRS("TRANSFER FUNCTION MATRIX-NUMERATOR");PRC("=",34);PRINTPOLYMX(TFN)
PRS("TRANSFER FUNCTION MATRIX-DENOMINATOR");PRC("=",36);PRINTPOLYMX(TFD
(!NOT!PROPER!PRS("TRANSFER FUNCTION MATRIX-POLYNOMIAL PART");
             PRC("=",40);PRINTPOLYMX(U));
NL(2);PRC("*",60);
[1:1!FLEX!,1:1!FLEX!]!POLY!SSR←REALIZ(TFN,TFD,W);
NL(5);PRC("R",60)
!END!;


NL(1)
!END!
!KEEP! GAUSS,GCRD,REALIZ,RUNREALIZ
!FINISH!
```