Coprimeness in Multidimensional System Theory and Symbolic Computation

by

Dean S. Johnson

A doctoral thesis submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of the Loughborough University of Technology, March 1993.

Acknowledgements

I would like to express my sincere gratitude to my supervisors Dr. A. Clive Pugh and Professor Gaynor E. Taylor for their expert support and encouragement during my three years of research into this thesis, without whom this work would not have been possible. I would also like to extend this gratitude to Professor C. Storey in his role as director of research.

My sincere thanks also extends to those who assisted in the proof reading of this thesis and on a more personal note to all my family and friends for their moral support.

This work was made possible by the financial support of the Science and Engineering Research Council.

Contents

Abstract

During the last twenty years the theory of linear algebraic and high-order differential equation systems has been greatly researched. Two commonly used types of system description are the so-called matrix fraction description (MFD) and the Rosenbrock system matrix (RSM); these are defined by polynomial matrices in one indeterminate. Many of the system's physical properties are encoded as algebraic properties of these polynomial matrices. The theory is well developed and the structure of such systems is well understood. Analogues of these 1-D realisations can be set up for many dimensional systems resulting in polynomial matrices in many indeterminates. The scarcity of detailed algebraic results for such matrices has limited the understanding of such systems.

Part one of this thesis considers some of the theoretical results for one indeterminate polynomial matrices in a many indeterminate framework. The structure of coprime MFDs pertaining to a system is investigated by considering the internal structure, e.g. invariant polynomials, and also the external structure, e.g. coprime representation, of the constituent matrices. Interestingly, the type of coprime MFD for a 2-D system does not depend on the realisation but is a property of the transfer function matrix. The invariant properties, such as zero structure, of the alternative RSM realisations are investigated via transformation methods. In particular, polynomial transformations induce an equivalence relation on least order RSMs.

Part two is concerned with the symbolic computation of the greatest common divisor of two polynomial matrices in two indeterminates using the MAPLE software. This is an important problem since the reduction of two 2-D polynomial matrices to coprime form (essential to the formation of a MFD) cannot be obtained with 1-D manipulation methods. Two basic features that makes MAPLE most suited to this type of problem is the ability to manipulate and simplify expressions involving unevaluated elements. The algorithms used are based upon certain theoretical suggestions which have previously occurred in the literature. Through the implementation of these algorithms certain weaknesses of both the theoretical method and the version of MAPLE used are demonstrated. Alternative methods to overcome these weaknesses are suggested.

Glossary of Notation

F a field;

$\mathbb{R}$ the field of real numbers ($\mathbb{R}\backslash\{0\}$ denotes the real numbers excluding zero);

$\mathbb{C}$ the field of complex numbers;

$F[z_1]$ the ring of polynomials in the single indeterminate z_1 with coefficients in the field F;

$F(z_1)$ the field of rational polynomials in the single indeterminates z_1 with coefficients in the field F;

$F[z_1, z_2]$ the ring of polynomials in the two indeterminates z_1, z_2 with coefficients in the field F;

$F(z_1)[z_2]$ the ring of polynomials in the indeterminate z_2 with coefficients in the field $F(z_1)$;

$F[z_1, z_2, \ldots, z_n]$ the ring of polynomials in the n indeterminates $z_1, z_2, \ldots, z_n$ with coefficients in the field F;

$F(z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n)[z_i]$ the ring of polynomials in the indeterminate z_i with coefficients in the field $F(z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n)$, i.e. the coefficients are rational functions in the indeterminates $z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n$;

$M^{i_1,i_2,\ldots,i_k}_{j_1,j_2,\ldots,j_k}$ the kth order minor using rows $i_1, i_2, \ldots, i_k$ and columns $j_1, j_2, \ldots, j_k$;

$|M|$ or $\det(M)$ the determinant of the matrix M;

M^{-1} the inverse of the matrix M;

I_p the pth order identity matrix;

$p(z_i^c)$ a scalar polynomial independent of z_i;

$\binom{n}{r}$ the number of combinations of n selecting r at a time;

$a|b$ the scalar polynomial a divides the scalar polynomial b;

$\delta(M)$ the McMillan degree of the matrix M;

$\deg_{z_i}(a)$ the degree in z_i of the scalar polynomial a;

$\delta_{r_i}(M)$ the ith row degree of the matrix M.

PART ONE

Coprimeness in Multidimensional System Theory

Preface

During the last thirty years the theory of polynomial matrices has been extensively used in the investigation of dynamical control systems. This theory has developed from the theory of matrices, e.g. Gantmakher [1] and MacDuffee [2], and from the theory of modern algebra, e.g. van der Waerden [3], Bocher [4] and [5]–[11].

Initially research was directed towards those systems generating polynomial matrices in one indeterminate, [12], [13], [14]. More recently the importance of multidimensional systems has been recognised [15], [16] due to the application to, for example, delay-differential systems, image processing, signal processing, partial differential equations and meteorology. The mathematical investigation of such systems requires

the study of polynomial matrices in many indeterminates, i.e. matrices with scalar polynomials in many indeterminates as its elements. The analysis of such matrices is more difficult than its 1-D counterpart due to the complexity of the underlying polynomial ring structure, i.e. the polynomial ring to which the matrix elements belong. For example, the division algorithm is extensively used in the manipulation of 1-D matrices, however, the ring $\mathbb{R}[z_1, z_2]$ is not Euclidean and therefore a division algorithm does not exist. Furthermore, the basic entities used to manipulate 1-D matrices are the elementary operations, which are equivalent to pre- and post-multiplication by a unimodular matrix; however, for matrices in two or more indeterminates the elementary operations do not fully define the class of unimodular matrices.

In addition to the fundamental differences between one- and two-indeterminate polynomial rings there are fundamental differences between two and higher dimensioned polynomial rings. Thus the analysis used to investigate the structure of 2-D polynomial matrices can not always be used for n-D polynomial matrices ($n \geq 3$). This difference of many-indeterminate polynomial rings is fundamental to the definition of coprimeness (the alternative terminology "relative primeness" is commonly used in 1-D matrix theory). In fact, for n-indeterminate polynomial matrices there are two distinct definitions of coprimeness if $n = 2$ and three distinct definitions for $n > 2$. One interpretation of this difference can be attributed to 2-D polynomials defining algebraic curves and 3-D polynomials defining algebraic surfaces [17]–[23]. Further evidence of the difference between two- and higher-dimensional polynomial rings is provided by Hutchins [24] in terms of primary-ideals. Thus multidimensional systems are considered in two parts: 2-D and n-D for $n \geq 3$.

The starting point of the analysis conducted in this thesis is the transfer function matrix arising from the input-output description of a system. The elements of such matrices are rational polynomials in a number of indeterminates. Two commonly used forms of describing the transfer function matrix are the matrix fraction description (MFD) and the Rosenbrock system matrix (RSM or system matrix). Many of the system's physical properties are encoded as algebraic properties of the polynomial matrices upon which these descriptions are based. For matrices in one indeterminate the theory is well developed and the structure of such systems is well understood. However, for matrices in more than one indeterminate the situation is less clear. It is the intention of Part One of this thesis to consider, in a mathematical context, some properties of the many-indeterminate polynomial matrices arising from the MFD (Chapter 2) and RSM (Chapter 3). Thus minimal consideration is given to the practical application of the derived results.

A MFD comprises of two polynomial matrices: a denominator and a numerator. The main results pertaining to MFDs are two-fold. Firstly, internal results relating the invariant polynomials of a class of coprime MFDs. In particular, it is shown that for the class of MFDs considered the denominator matrices of a transfer function matrix essentially possess the same invariant polynomial structure. This property also extends to the invariant polynomials of the numerator matrices. Secondly, external results concerning the type of coprime MFD possessed by a particular transfer function matrix are considered. More specifically, given one coprime MFD of a transfer function matrix it is possible to deduce the type of coprimeness possessed by all other MFDs with the same transfer function matrix.

The RSM forms a partitioned polynomial matrix and is a more general way of describing the transfer function matrix. Chapter 3 begins by addressing the problem of preserving the finite and infinite zero structure of RSMs polynomial in one indeterminate and gives a number of conditions for the absence of infinite zeros. Next, a number of transformations of many indeterminate system matrices are defined. In particular, the three notions of coprimeness are used to define three types of transformation. The extent to which each transformation defines an equivalence relation is investigated together with the invariant properties of each transformation. In 1-D system theory a fundamental type of system realisation is the least order system matrix. Here the interpretation of coprimeness is used to define a type of least order realisation for many-indeterminate system matrices. Finally the chapter concludes by considering a specific type of 2-D system, the linear differential multipass process, in terms of the zeros of the system matrix and the formation of a least order system matrix.

The illustration of these results has proved to be difficult. In particular, it is difficult to prove that a matrix is factor coprime, without additionally being minor coprime. Thus standard examples derived in the literature together with some new examples are used to demonstrate the results. In theory it is easy to construct a coprime matrix by extracting a greatest common divisor (GCD); however, for matrices in more than one indeterminate the practical procedure for obtaining this GCD is not obvious. This is due to the absence of a division algorithm and the presence of unimodular matrices that are not defined by elementary operations. Thus the techniques used for 1-D matrices are not appropriate. Certain theoretical suggestions made in the literature are used in Part Two to compute a GCD of 2-D polynomial matrices using a symbolic manipulator, MAPLE. The ability of a symbolic manipulator to express

and simplify calculations in abstract algebraic entities, such as polynomials, facilitates this process.

Note: Where work is not referenced or attributed to another person or persons the work is original. This particulaly applies to the proofs of the theorems: to the best of the author's knowledge a proof is original whenever it is not referenced.

Chapter 1

Preliminary Mathematics

1.1 Introduction

The basic entity that is used when considering a system realisation such as a matrix fraction description (MFD) or a Rosenbrock system matrix is a polynomial matrix. It is the purpose of this first chapter to introduce some preliminary concepts and definitions required for the analysis of multidimensional systems using polynomial matrices.

The chapter begins by giving some definitions for general matrices, followed by a discussion about Euclidean rings and the consequence of $F[z_1, z_2, \ldots, z_n]$ being non-Euclidean, where F is a field. Finally, the concept of coprimeness of scalar and matrix

polynomials is considered. In particular, the single notion of relative primeness[1] for matrices polynomial in one indeterminate is seen to generalise to three different notions for matrices with three or more indeterminates. The characterisations of relative primeness given by Rosenbrock [12] are related to the three notions of coprimeness by considering their definitions.

[1] *The term "relative primeness" is taken from 1-D scalar polynomial theory and used by Rosenbrock [12] to describe a similar property for polynomial matrices in one indeterminate. However, for polynomial matrices in two and more indeterminates "coprime" has been adopted in the literature (and in this thesis) as the accepted terminology, possibly due to the different definitions that arise from these higher dimensioned matrices and the ensuing cumbersome terminology, e.g. factor (left) coprime as opposed to relatively factor (left) prime or some permutation of these words.*

1.2 Preliminary Definitions

A polynomial matrix, $A(z)$, of size $p \times q$ (written $A_{p\times q}(z)$) is defined to be an array of polynomials $a_{ij}(z) \in F[z]$ with p rows and q columns for $i = 1, 2, \ldots, p$ and $j = 1, 2, \ldots, q$ where z is a set of indeterminates and F is the coefficient ring, typically $\mathbb{R}$ or $\mathbb{C}$. The number of indeterminates represented by z defines the dimension of the polynomial and hence the dimension of the polynomial matrix[2]. The matrix is additionally described as being square if the number of rows and the number of columns are equal; the case of a differing number of rows and columns is described as being rectangular. Thus an n dimensional (n-D) polynomial matrix $M(z)$ of size $p \times q$ is represented by

$$M(z) = \begin{pmatrix} m_{11}(z) & m_{12}(z) & \cdots & m_{1q}(z) \\ m_{21}(z) & m_{22}(z) & \cdots & m_{2q}(z) \\ \vdots & \vdots & \ddots & \vdots \\ m_{p1}(z) & m_{p2}(z) & \cdots & m_{pq}(z) \end{pmatrix} \tag{1}$$

where $(z) = (z_1, z_2, \ldots, z_n)$ and $m_{ij} \in F[z_1, z_2, \ldots, z_n]$ for $i = 1, 2, \ldots, p$ and $j = 1, 2, \ldots, q$.

Definition 1.1: (*Principal Diagonal*)
The *principal diagonal* of a polynomial matrix, $A_{p\times q}(z)$ is defined to be the elements that lie in positions (i, i) for $i = 1, 2, \ldots, \min(p, q)$. □

Definition 1.2: (*Diagonal Polynomial Matrix*)
A *diagonal polynomial matrix*, $A_{p\times q}(z)$, is defined as a matrix with all non-zero elements appearing on the principal diagonal. □

Thus a $p \times q$ rectangular diagonal polynomial matrix has rows of zeros lying in positions $q + 1$ to p if $p > q$ or columns of zeros lying in positions $p + 1$ to q if $p < q$.

The unit matrix I_p is a $p \times p$ square diagonal matrix with p rows and columns with 1's on the principal diagonal and zeros in all other positions.

A less restricted form of polynomial matrix than the diagonal polynomial matrix is a triangular matrix.

[2] *In some texts the size of a matrix is described as its dimension, here the dimension of a polynomial matrix represents the number of indeterminates and the size represents the number of rows and columns.*

Definition 1.3: (*Upper Triangular Polynomial Matrix*)
An *upper triangular polynomial matrix*, $H_{p\times q}(z)$ is defined as having all elements below the principal diagonal identically zero, i.e.

$$H(z) = \begin{pmatrix} h_{11}(z) & h_{12}(z) & \cdots & h_{1q}(z) \\ 0 & h_{22}(z) & \cdots & h_{2q}(z) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & h_{qq}(z) \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \quad \text{if} \quad p \geq q \tag{2}$$

$$H(z) = \begin{pmatrix} h_{11}(z) & h_{12}(z) & \cdots & h_{1p}(z) & \cdots & h_{1q}(z) \\ 0 & h_{22}(z) & \cdots & h_{2p(z)} & \cdots & h_{2q}(z) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & h_{pp}(z) & \cdots & h_{pq}(z) \end{pmatrix} \quad \text{if} \quad p < q \tag{3}$$

□

Similarly a lower triangular polynomial matrix possesses zero elements above the principal diagonal.

Definition 1.4: (*Minor*)
A *minor* of order r of a polynomial matrix $M_{p\times q}(z)$ is the determinant of the polynomial matrix formed by selecting r rows and r columns of $M(z)$, where $r \leq \min(p, q)$. In particular the minor denoted by $M^{i_1,i_2,\ldots,i_r}_{j_1,j_2,\ldots,j_r}$ is formed by selecting rows $i_1, i_2, \ldots, i_r$ and columns $j_1, j_2, \ldots, j_r$. □

Note: For a matrix with p rows and q columns there are

$$\frac{p!}{r!(p-r)!}\frac{q!}{r!(q-r)!}$$

minors of order r.

The high-order minors of a matrix $M_{p\times q}(z)$ are defined to be the minors of order $\min(p, q)$. Thus if $p \leq q$ the high-order minors are formed by selecting every row of $M(z)$ together with p columns. Hence for a matrix with p rows and q columns, with $p \leq q$, there are

$$\frac{q!}{p!(q-p)!}$$

high-order minors.

Definition 1.5: (*Unimodular Polynomial Matrix*)
The polynomial matrix $U_{p\times p}(z)$ with elements in $F[z]$ is said to be *unimodular* over the ring $F[z]$ if the elements of $U^{-1}(z)$ are polynomial over the ring $F[z]$, where $U^{-1}(z)$ denotes the inverse of the matrix $U(z)$. □

Equivalently, a unimodular matrix in $F[z]$ may be defined as a matrix with a determinant that is a unit of the coefficient ring F, i.e. an element of F with multiplicative inverse also in F. For example, a unimodular matrix $U(z)$ over the ring $\mathbb{R}[z_1, z_2, \ldots, z_n]$ has constant determinant; and over $\mathbb{R}(z_1)[z_2]$ has determinant in $\mathbb{R}(z_1)$, i.e. a rational function in the indeterminate z_1. The formal definition of a unimodular matrix, $U(z)$, includes the ring structure to which it applies; when it is omitted the statement $|U(z)| = constant$ is understood.

A subclass of unimodular matrices over a polynomial ring are the elementary matrices. These are defined as the matrices effecting the elementary operations, given by:

Definition 1.6: (*Elementary Row Operations*)
Let $A_{p\times q}(z)$ be a polynomial matrix with elements in $F[z]$. Each of the following operations define an *elementary row operation* on the matrix $A(z)$. These operations are effected by pre-multiplication by the matrices E_1, E_2, E_3:

(i) Multiply row i of $A(z)$ by $a \in F$;

$$E_1 = \begin{array}{c} \\ 1 \\ i \\ \\ p \end{array} \begin{array}{c} \begin{array}{ccccccc} & & & & i & & p \end{array} \\ \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & a & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \end{array} \tag{4}$$

such that $|E_1(z)| = a \in F$ and $|E_1^{-1}(z)| = a^{-1} \in F$;

(ii) addition of row i with row j, multiplied by $a(z) \in F[z]$;

$$E_2 = \begin{array}{c} \\ 1 \\ \\ i \\ \\ j \\ \\ p \end{array} \begin{array}{c} \begin{array}{ccccccccc} & & & & i & & j & & p \end{array} \\ \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & a(z) & \cdots & 1 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & o & \cdots & 1 \end{pmatrix} \end{array} \tag{5}$$

thus $|E_2(z)| = 1 \in F$ and $|E_2^{-1}(z)| = 1^{-1} \in F$;

(iii) permutation of rows i and j;

$$E_3 = \begin{array}{c} \\ 1 \\ \\ i \\ \\ j \\ \\ p \end{array} \begin{array}{c} \begin{array}{ccccccc} & & & i & & j & p \end{array} \\ \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & \cdots & 1 & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \end{array} \tag{6}$$

thus $|E_3(z)| = 1 \in F$ and $|E_3^{-1}(z)| = 1^{-1} \in F$. □

The column operations are defined similarly and the elementary matrices effecting the operations are defined to be post-multiplicative $q \times q$ unimodular matrices of the same form given in Definition 1.6.

For general rings $F[z_1, z_2, \ldots, z_n]$ with $n \geq 2$ not all unimodular matrices can be formed as a product of elementary matrices. This fact emanates from the absence of a division algorithm. To illustrate this consider the following example due to Šebek [25].

Example 1.1: Consider the 2-D scalar equation

$$(1 - z_1 z_2)x(z_1, z_2) + z_1^2 y(z_1, z_2) = 1$$

where $x(z_1, z_2)$, $y(z_1, z_2)$ are unknown polynomials in the two indeterminates (z_1, z_2). This equation has a solution since

$$(1 - z_1 z_2 \quad z_1^2) \underbrace{\begin{pmatrix} 1 + z_1 z_2 & -z_1^2 \\ z_2^2 & 1 - z_1 z_2 \end{pmatrix}}_{U(z_1, z_2)} = (1 \quad 0).$$

Now $|U(z_1, z_2)| = 1$ thus $U(z_1, z_2)$ is a unimodular matrix in $\mathbb{R}[z_1, z_2]$. However, there are no elementary operations that may be performed on $(1 - z_1 z_2 \quad z_1^2)$ to produce the solution $(1 \quad 0)$ as z_1^2 does not divide any term of $1 - z_1 z_2$. Hence $U(z_1, z_2)$ can not be factored as a product of elementary matrices. Matrices of this form are termed *secondary matrices*. □

Some further consequences of the ring $F[z_1, z_2, \ldots, z_n]$ being non-Euclidean are discussed in the following section.

1.3 Some Non-Euclidean Consequences

The existence of a division algorithm for Euclidean polynomial rings, such as $\mathbb{R}[z_1]$, forms the basis for the algorithmic derivation of many canonical forms, such as the 1-D Smith form [26] and the solution of 1-D polynomial equations [27]. Any polynomial ring can be considered as a sub ring of a larger ring which possesses a division algorithm. The process of achieving this larger, or *generalised*, ring is to favour one of the indeterminates and to consider elements of the ring to be polynomial in this indeterminate with coefficients rational in the others. For example, consider the ring $\mathbb{R}[z_1, z_2, \ldots, z_n]$ to be a sub ring of the Euclidean ring $\mathbb{R}(z_1, z_2, \ldots, z_{n-1})[z_n]$; the degree condition of the division algorithm is then defined with respect to z_n, i.e. suppose that $a(z), b(z) \in \mathbb{R}[z_1, z_2, \ldots, z_n]$ with the degree in z_n of $a(z)$ (denoted by $\deg_{z_n}(a)$) greater than $\deg_{z_n}(b)$, thus

$$a(z) = q(z)b(z) + r(z) \tag{7}$$

where $q(z), r(z) \in \mathbb{R}(z_1, z_2, \ldots, z_{n-1})[z_n]$ and $\deg_{z_n}(r) < \deg_{z_n}(b)$. The polynomial situation may be recovered by multiplying the equation (7) by a polynomial $n \in \mathbb{R}[z_1, z_2, \ldots, z_{n-1}]$; this process in known as renormalisation. Thus a type of division algorithm is defined for non-Euclidean rings.

The process described above allows the solution of certain matrix equations [28]–[32] and also the derivation of 2-D canonical forms required for the algorithmic determination of the greatest common divisor of 2-D polynomial matrices [32], [33]–[35] (this is the subject of Part Two).

One further consequence of a ring not possessing a division algorithm is the algorithmic derivation of the Smith form. Recall, for the Euclidean ring $\mathbb{R}[z_1]$, the Smith form $S_A(z_1)$ of a polynomial matrix $A_{p\times q}(z_1)$ can be formed by pre- and post-multiplication by unimodular matrices, i.e.

$$V(z_1)A(z_1)U(z_1) = S_A(z_1)$$

Furthermore, since the polynomial ring $\mathbb{R}[z_1]$ is Euclidean, all unimodular matrices are products of elementary matrices; thus the Smith form can be derived by a series of row and column operations. (The algorithmic derivation of the Smith form is often taken to be the definition of the Smith form.) However, this is not possible for many-indeterminate polynomial matrices [36], [37]. To demonstrate this consider the following example.

Example 1.2: Let the 2-D polynomial matrix $A(z_1, z_2)$ with elements in $\mathbb{R}[z_1, z_2]$ be defined by

$$A(z_1, z_2) = \begin{pmatrix} z_1 & 0 \\ 0 & z_2 \end{pmatrix}$$

thus the determinantal divisors are

$$D_0 = 1, \quad D_1 = 1, \quad D_2 = z_1 z_2$$

and the invariant polynomials are given by

$$\epsilon_1 = \frac{D_1}{D_0} = 1, \quad \epsilon_2 = \frac{D_2}{D_1} = z_1 z_2$$

therefore the Smith form, $S_A(z_1, z_2)$, is given by

$$S_A(z_1, z_2) = \begin{pmatrix} 1 & 0 \\ 0 & z_1 z_2 \end{pmatrix}.$$

Suppose that $S_A(z_1, z_2)$ can be derived by pre- and post-multiplication by unimodular matrices, $U(z_1, z_2)$ and $V(z_1, z_2)$. Thus

$$U(z_1, z_2) A(z_1, z_2) V(z_1, z_2) = S_A(z_1, z_2)$$

Write this equation in the form

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} z_1 & 0 \\ 0 & z_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & z_1 z_2 \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix}$$

where $ad - bc = c_1$, $eh - fg = c_2$ and $c_1, c_2 \in \mathbb{R}$. Thus the constituent equations are given by

$$az_1 = e \tag{8}$$

$$bz_2 = f \tag{9}$$

$$cz_1 = z_1 z_2 g \tag{10}$$

$$dz_2 = z_1 z_2 h \tag{11}$$

Substituting (8) and (9) into $eh - fg = c_2$ it is seen that there does not exist polynomials $a(z_1, z_2)$, $b(z_1, z_2)$, $h(z_1, z_2)$, $g(z_1, z_2)$ such that

$$ahz_1 - bgz_2 = c_1$$

for all z_1, z_2. For example, a contradiction is obtained by considering the case for $z_1 = z_2 = 0$. □

Thus an alternative definition is formulated via determinantal divisors and invariant polynomials:

Definition 1.7: (*Determinantal Divisors*)
The $i \times i$ *determinantal divisor*, $D_i(z)$, of a polynomial matrix $A_{p\times q}(z)$ is defined as a greatest common divisor (GCD) of the determinants of the $i \times i$ minors of $A(z)$. □

Definition 1.8: (*Invariant Polynomials*)
The *invariant polynomials*, $\epsilon_i(z)$, of a polynomial matrix $A_{p\times q}(z)$ are defined to be the polynomials given by

$$\epsilon_i(z) = \frac{D_i(z)}{D_{i-1}(z)} \qquad \text{for} \quad i = 1, 2, \ldots, \min(p, q) \quad \text{and} \quad D_0 = 1. \qquad \square$$

Note: The polynomial $\epsilon_1(z)$ is often refered to as the first invariant polynomial and $\epsilon_{\min(p,q)}(z)$ is often refered to as the last invariant polynomial.

Definition 1.9: (*Smith Form*)
The *Smith form*, $S_{p\times q}(z)$, of a polynomial matrix $A_{p\times q}(z)$ is defined as:

(i) if $p \geq q$

$$S(z) = \begin{pmatrix} \epsilon_1(z) & 0 & \cdots & 0 \\ 0 & \epsilon_2(z) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \epsilon_q(z) \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

(ii) if $p < q$

$$S(z) = \begin{pmatrix} \epsilon_1(z) & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \epsilon_2(z) & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \epsilon_p(z) & 0 & \cdots & 0 \end{pmatrix}$$

where $\epsilon_i(z)$ is the ith invariant polynomial of the matrix $A(z)$. □

Note: A greatest common divisor is only unique modulo a multiplicative constant, hence the determinantal divisors and invariant polynomials are only unique polynomials modulo a multiplicative constant. Consequently the Smith form is only unique modulo a multiplicative constant diagonal matrix.

Lemma 1.1: *The Smith form of a polynomial matrix is a polynomial matrix.*

Proof: The proof is contained in [12]. □

1.4 Scalar Polynomials

As a preamble to the following section concerning polynomial matrices it is advantageous to briefly study the coprimeness of scalar polynomials and the types of singularities possessed by polynomials in many indeterminates. The ideas developed here will be seen to relate to the polynomial matrix case.

Recall that a set of polynomials in one indeterminate, $\alpha_1(z_1), \alpha_2(z_1), \ldots, \alpha_k(z_1)$, are described as being relatively prime if they possess no value $z_1^{[0]} \in \mathbb{C}$ such that $\alpha_1(z_1^{[0]})$, $\alpha_2(z_1^{[0]}), \ldots, \alpha_k(z_1^{[0]})$ are not all identically zero. If, however, such a value exists the polynomial $z_1 - z_1^{[0]}$ is a factor of all the polynomials in the set.

This is not the situation for polynomials in more than one indeterminate. To illustrate this consider the following example.

Example 1.3: Let the two polynomials $a(z_1, z_2)$, $b(z_1, z_2)$ be defined by

$$a(z_1, z_2) = z_1, \qquad b(z_1, z_2) = z_2.$$

Thus there exists a value of the pair (z_1, z_2) such that $a(z_1, z_2) = 0$, $b(z_1, z_2) = 0$, namely $(z_1, z_2) = (0, 0)$. However, the two polynomials do not possess a common polynomial factor because z_1 and z_2 are independent. □

Thus for scalar polynomials in more than one indeterminate the following definitions will be used.

Definition 1.10: (*Factor Coprime Polynomials*)
A set of scalar polynomials, $\alpha_1(z), \alpha_2(z), \ldots, \alpha_k(z)$, is said to be *factor coprime* if there does not exist a polynomial $g(z)$, which is a common divisor of $\alpha_1(z), \alpha_2(z), \ldots, \alpha_k(z)$. □

Definition 1.11: (*Zero Coprime Polynomials*)
A set of scalar polynomials, $\alpha_1(z), \alpha_2(z), \ldots, \alpha_k(z)$, is said to be *zero coprime* if there does not exist a value z of the indeterminates $(z_1, z_2, \ldots, z_n)$ such that the polynomials $\alpha_1(z), \alpha_2(z), \ldots, \alpha_k(z)$ are identically zero. □

Note: Zero coprime polynomials are a subset of factor coprime polynomials.

Note: The process of determining the coprimeness of two scalar polynomials is given, amongst others, in [38], [39].

A fundamental result for polynomials is the Hilbert's Nullstellensatz, derived from the theory of polynomial ideals [3].

Lemma 1.2: (*Hilbert's Nullstellensatz*)
If f is a polynomial in $F[z_1, z_2, \ldots, z_n]$, which vanishes at all zeros common to the polynomials $f_1, f_2, \ldots, f_r$ then

$$f^\rho = a_1 f_1 + a_2 f_2 + \cdots + a_r f_r$$

is valid for some integer ρ and $a_1, a_2, \ldots, a_r \in F[z_1, z_2, \ldots, z_n]$ (and conversely).

Proof: The proof is given in [3]. □

The form of Hilbert's Nullstellensatz which plays an important role in multidimensional system theory is the case when the polynomials $f_1, f_2, \ldots, f_r$ do not possess any common zeros:

Corollary: *If $f_1, f_2, \ldots, f_r \in F[z_1, z_2, \ldots, z_n]$ do not possess any common zeros then*

$$1 = a_1 f_1 + a_2 f_2 + \cdots + a_r f_r$$

for some $a_1, a_2, \ldots, a_r \in F[z_1, z_2, \ldots, z_n]$ (and conversely).

Proof: The proof is a special case of the above lemma. Clearly if the polynomials $f_1, f_2, \ldots, f_r$ do not have any common zeros then they possess the same common zeros as the polynomial '1'. Hence the corollary is proved. □

Consider now a ratio, $f(z)$, of two polynomials $a(z)$ and $b(z)$ where

$$f(z) = \frac{a(z)}{b(z)} \tag{12}$$

This is called a polynomial fraction or more commonly a *rational polynomial.* It is additionally described as being *irreducible* if $a(z)$ and $b(z)$ are factor coprime. The singularities of $f(z)$ occur when $b(z) = 0$: these have two classifications given by the following definition.

Definition 1.12: (*Polynomial Non-Essential Singularities*)
The singularities of the irreducible rational polynomial $f(z)$

$$f(z) = \frac{a(z)}{b(z)}$$

are said to be

(i) *non-essential singularities of the first kind* if $b(z) = 0$ and $a(z) \neq 0$;

(ii) *non-essential singularities of the second kind* if $b(z) = 0$ and $a(z) = 0$. □

Thus it has been seen that scalar polynomials in more than one indeterminate possess two types of coprimeness and also two types of non-essential singularities. The precise algebraic definition of these terms for polynomials in many indeterminates may be found in [17].

To conclude this section on scalar polynomials a result is given relating the division of polynomials modulo a factor coprime set of polynomials. This well known result [5] is fundamental to a number of results, in particular the MFD Structure Theorem of Chapter 2 (Theorem 2.5 and 2.12); therefore the main ideas of the proof are given.

Theorem 1.1: *Let* $a, b \in \mathbb{R}[z_1, z_2, \ldots, z_n]$ *and suppose that*

$$a|\alpha_i b \quad \text{for} \quad i = 1, 2, \ldots, m$$

where $\alpha_1, \alpha_2, \ldots, \alpha_m$ *are a factor coprime set of polynomials in* $\mathbb{R}[z_1, z_2, \ldots, z_n]$ *and* $x|y$ *denotes* x *divides* y. *Then* $a|b$.

Proof: $\mathbb{R}[z_1, z_2, \ldots, z_n]$ is a unique factorisation domain, therefore all elements of $\mathbb{R}[z_1, z_2, \ldots, z_n]$ can be decomposed as a product of powers of irreducible elements. Write

$$a = p_1^{r_1} p_2^{r_2} \ldots p_k^{r_k}$$

where $p_1, p_2, \ldots, p_k$ are the irreducible factors of a. Then

$$p_j^{r_j} | \alpha_i b \quad \text{for} \quad i = 1, 2, \ldots, m, \quad j = 1, 2, \ldots, k$$

and clearly

$$\begin{aligned} \text{either} \quad & p_j | \alpha_i \quad \text{for} \quad i = 1, 2, \ldots, m, \quad j = 1, 2, \ldots, k \\ \text{or} \quad & p_j | b \quad \text{for} \quad j = 1, 2, \ldots, k. \end{aligned}$$

Since $\alpha_1, \alpha_2, \ldots, \alpha_m$ are factor coprime there exists no irreducible element, $q \in \mathbb{R}[z_1, z_2, \ldots, z_n]$, such that $q|\alpha_i$ for $i = 1, 2, \ldots, m$, i.e. the irreducible factorisations of α_i for $i = 1, 2, \ldots, m$ in $\mathbb{R}[z_1, z_2, \ldots, z_n]$ contain neither q nor one of its associates. Thus no p_j for $j = 1, 2, \ldots, k$, the irreducible factors of a in $\mathbb{R}[z_1, z_2, \ldots, z_n]$, can be a divisor of α_i for $i = 1, 2, \ldots, m$. Hence

$$p_j | b \quad \Rightarrow \quad p_j^{r_j - 1} | \alpha_i \frac{b}{p_j} \quad \text{for} \quad i = 1, 2, \ldots, m \quad j = 1, 2, \ldots, k$$

Thus repeating the same argument for $j = 1, 2, \ldots, k$ the theorem is proved. □

1.5 Coprimeness of Polynomial Matrices

In 1-D system theory a polynomial matrix with rank degeneracies may be viewed as a product of two polynomial matrices, one with full rank and the other containing the rank degeneracies. The former of these matrices is termed, by Rosenbrock [12], a *relatively prime* matrix and the latter a *greatest common divisor*. The definition of a relatively prime matrix yields the following equivalent characterisations [12].

Theorem 1.2: *The polynomial matrix $A_{p\times q}(z_1)$ with $p \leq q$ is said to be relatively (left) prime if and only if any one of the following equivalent conditions is satisfied.*

(a) The rank of $A(z_1)$ is p for all $z_1 \in \mathbb{C}$.

(b) The Smith form of $A(z_1)$ is $[I_p|0_{(q-p)\times p}]$.

(c) There exists a polynomial matrix $X_{(q-p)\times q}(z_1)$ such that the Smith form of the square polynomial matrix $[A^T(z_1)|X^T(z_1)]^T$ is I_{p+q}.

(d) There exists a relatively (right) prime matrix $X_{q\times p}(z_1)$ such that

$$A(z_1)X(z_1) = I_p. \qquad \square$$

Similar characterisations exist for the relatively (right) prime matrix $A_{p\times q}(z_1)$ with $p \geq q$.

If these characterisations are considered in a many-dimensional framework it is immediately apparent that there exists more than one notion of relative primeness. Consider the matrix

$$B(z_1, z_2) = (\, z_1 \quad z_2 \,)$$

The characterisation (a) above suggests that $B(z_1, z_2)$ is not relatively prime because $z_1 = z_2 = 0$ is a rank degeneracy, however, the Smith form (Definition 1.9) of $B(z_1, z_2)$ is $[1|0]$ suggesting that $B(z_1, z_2)$ is relatively prime. Similar analysis of characterisations (c) and (d) also form contradicting statements.

In fact, there are three different notions of relative primeness for n-D polynomial matrices [40], termed factor coprime, minor coprime and zero coprime.

Definition 1.13: (*Matrix Coprimeness*)
Let $A_{p\times q}(z)$ be a n-D polynomial matrix with $p \leq q$. Then

(i) $A(z)$ is *factor (left) coprime* if all factorisations

$$A(z) = Q(z)A^*(z)$$

where $Q(z)$ and $A^*_{p\times q}(z)$ are polynomial matrices such that $Q(z)$ is unimodular, i.e. all left matrix divisors of $A(z)$ are unimodular.

(ii) $A(z)$ is *minor (left) coprime* if all the $p \times p$ minors of $A(z)$ form a factor coprime set of polynomials, i.e. have no polynomial factor.

(iii) $A(z)$ is *zero (left) coprime* if there exists no n-tuple z which is a zero of all the $p \times p$ minors of $A(z)$. □

Note: A matrix $A_{p\times q}(z)$ with $p \geq q$ is defined to be factor/minor/zero (right) coprime if $A^T(z)$ is factor/minor/zero (left) coprime respectively.

A polynomial matrix, $A_{p\times q}(z)$, can be partitioned to form two matrices, i.e. for $p \leq q$

$$A(z) = [A_1(z) \quad A_2(z)] \tag{13}$$

where $A_1(z)$ has size $p\times r$ and $A_2(z)$ has size $p\times(q-r)$ for some r such that $0 \leq r \leq q$. In this case, if $A(z)$ possesses a form of coprimeness the pair $A_1(z)$, $A_2(z)$ is said to form a coprime pair with the same type of coprimeness. For example, if $A(z)$ is a zero (left) coprime matrix then $A_1(z)$, $A_2(z)$ (defined by (13)) form a zero (left) coprime pair. Similarly, a right coprime matrix can be partitioned by its rows to form a right coprime pair.

It is evident from the above definitions and Theorem 1.2 that factor, minor and zero coprimeness are equivalent for the case $n = 1$. The following theorem due to Youla and Gnavi [40] formally establishes this fact together with other equivalencies for certain types of polynomial matrix.

Theorem 1.3: (*Coprimeness Equivalence*)

(i) For $n = 1$ the three definitions of coprimeness are the same.

(ii) For $n = 2$ minor and factor coprime are equivalent and zero coprime is a different notion.

(iii) For $n > 2$ none of the definitions are equivalent.

(iv) For any n zero coprime $\Rightarrow$ minor coprime $\Rightarrow$ factor coprime. □

A consequence of this theorem is that zero coprime matrices may be considered to be a subset of minor coprime matrices, which in turn may be considered to be a subset of factor coprime matrices. Thus if

S_{factor} denotes the set of factor coprime matrices;

S_{minor} denotes the set of minor coprime matrices;

S_{zero} denotes the set of zero coprime matrices;

then

$$\begin{aligned} S_{zero} \subset S_{minor} \subset S_{factor} \quad &\text{for} \quad n \geq 3; \\ S_{zero} \subset S_{minor} = S_{factor} \quad &\text{for} \quad n = 2; \\ S_{zero} = S_{minor} = S_{factor} \quad &\text{for} \quad n = 1. \end{aligned}$$

Note: Throughout this thesis the most specific type of coprimeness will be used to describe the coprimeness of a matrix, unless otherwise stated. For example, if $A(z_1, z_2)$ is zero coprime it will not, in general, be referred to as a minor coprime matrix.

Note: When matrices in two indeterminates are discussed the terms zero coprime and minor coprime will be used to describe the two notions of coprimeness. For matrices in one indeterminate the Rosenbrock [12] terminology will be used, i.e. relative primeness, or occasionally zero coprimeness.

Two of the three notions of coprimeness can be characterised by necessary and sufficient conditions [40]. These characterisations are often referred to as the Bézout identities for zero and minor coprimeness and are fundamental to many results given in subsequent chapters.

Theorem 1.4: (*Bézout Identities*)
The two polynomial matrices $A_{m\times p}(z)$ and $B_{m\times q}(z)$ with $p+q \geq m \geq 1$ are:

(i) zero (left) coprime if and only if there exist two polynomial matrices $X_{p\times m}(z)$ and $Y_{q\times m}(z)$ such that

$$A(z)X(z) + B(z)Y(z) = I_m \tag{14}$$

(ii) minor (left) coprime if and only if there exist $p \times m$ polynomial matrices $X_1(z), X_2(z), \ldots, X_n(z)$ and $q \times m$ polynomial matrices $Y_1(z), Y_2(z), \ldots, Y_n(z)$ such that

$$\left.\begin{aligned} A(z)X_1(z) + B(z)Y_1(z) &= \psi_1(z_1^c)I_m \\ A(z)X_2(z) + B(z)Y_2(z) &= \psi_2(z_2^c)I_m \\ \vdots \qquad\qquad & \vdots \qquad\qquad \vdots \\ A(z)X_n(z) + B(z)Y_n(z) &= \psi_n(z_n^c)I_m \end{aligned}\right\} \tag{15}$$

where $\psi_i(z_i^c)$ is a polynomial in the $n-1$ indeterminates $z_1, \ldots, z_{i-1}, z_{i+1}, \cdots, z_n$. □

Using the above two theorems and the definitions of coprimeness it is possible to interpret the equivalent forms for 1-D coprimeness, given in Theorem 1.2, in an n-D framework.

Firstly consider the case of a polynomial matrix, $A_{p\times q}(z_1, z_2)$, in two indeterminates. Thus the notions of factor coprimeness and minor coprimeness are equivalent, i.e. the terms "factor" and "minor" are interchangeable: as noted above, to avoid confusion the term "minor" will always be used when discussing 2-D polynomial matrices, such that all matrix divisors are unimodular and hence the high-order minors are factor coprime. The definition of a 1-D relatively prime matrix corresponds to the definition of a 2-D minor coprime matrix.

The case of left coprimeness, i.e. $p \leq q$, will be considered as the case of right coprimeness follows in an analogous way by considering the matrix $A^T(z_1, z_2)$.

(a) A factor coprime set of polynomials, $\alpha_1(z_1, z_2), \alpha_2(z_1, z_2), \ldots, \alpha_k(z_1, z_2)$, does not possess a common polynomial factor of all the elements, by definition, but this does not guarantee the absence of a pair $(z_1^{[0]}, z_2^{[0]})$ such that all $\alpha_i(z_1^{[0]}, z_2^{[0]})$ are non-zero, see Section 1.4. The high-order minors of the coprime matrix $A(z_1, z_2)$ are necessarily factor coprime, however, this does not guarantee zero coprimeness of the high-order minors, i.e. there may exist a pair $(z_1^{[0]}, z_2^{[0]})$ such that all of the high-order minors are identically zero. Hence the rank of $A(z)$ is less than p for such a pair.

In view of the above discussion the matrix $A(z)$ is zero (left) coprime if the rank of $A(z)$ is p for all $z_1, z_2 \in \mathbb{C}$.

(b) The Smith form of a polynomial matrix, $A_{p\times q}(z_1, z_2)$, is defined as a diagonal matrix $S_{p\times q}(z)$ formed by the invariant polynomials of $A(z)$, Definition 1.9. For any type of 2-D coprime polynomial matrix the high-order minors are factor coprime, thus the last invariant polynomial is a constant, taken to be 1. By the divisibility property of the invariant polynomials all other invariant polynomials may also be taken to be 1. Hence the Smith form of a minor or zero (left) coprime matrix is $[I_p|0_{(q-p)\times p}]$.

(c) Suppose that there exists a polynomial matrix $X_{(q-p)\times p}(z)$ such that

$$F_{q\times q}(z) = \begin{pmatrix} A_{p\times q}(z) \\ X_{(q-p)\times p}(z) \end{pmatrix}$$

has Smith form I_q. Hence $\det F(z) = 1$ and expanding $\det F(z)$ using Laplace expansion on the first p rows an equation of the form

$$1 = \sum_j \delta^{1,2,\ldots,p}_{j_1,j_2,\ldots,j_p} F^{1,2,\ldots,p}_{j_1,j_2,\ldots,j_p} \bar{F}^{1,2,\ldots,p}_{j_1,j_2,\ldots,j_p}$$

is obtained where, for $1 \leq j_1 < j_2 < \cdots < j_p \leq q$,

$$\delta^{1,2,\ldots,p}_{j_1,j_2,\ldots,j_p} = (-1)^{\sum_{i=1}^{p}(i+j_i)}$$

and $\bar{F}^{1,2,\ldots,p}_{j_1,j_2,\ldots,j_p}$ denotes the minor obtained by deleting rows $1,2,\ldots,p$ and columns $j_1,j_2,\ldots,j_p$. Thus if the high-order minors $F^{1,2,\ldots,p}_{j_1,j_2,\ldots,j_p}$ of $A(z)$ possess a zero, a contradiction is obtained. Therefore $A(z)$ is zero (left) coprime.

Conversely, if $A(z)$ is zero coprime by the completion theorem[3], there exists a polynomial matrix $X_{(q-p)\times q}(z)$ such that $[A^T(z)|X^T(z)]$ is unimodular. Therefore it has Smith form I_q.

(d) By Theorem 1.4 a Bézout identity, having right-hand-side as an appropriately sized identity matrix, exists if and only if $A(z)$ is zero (left) coprime. However, by Theorem 1.4, it is possible to modify the definition of a Bézout identity to provide necessary and sufficient conditions for minor coprimeness. In effect, the statements that form the Bézout identity are two 1-D Bézout identities formed firstly with respect to $F(z_2)[z_1]$ and secondly with respect to $F(z_1)[z_2]$ together with renormalisation to recover the polynomial situation.

Consider now the general case for any $n \geq 3$. All three definitions of coprimeness are not equivalent, therefore for each statement there are three possibilities, either factor, minor or zero coprime. By using the same arguments as above, the 2-D statements hold for n-D matrices, except for the definition of a 1-D relatively prime matrix, which corresponds to a factor coprime matrix in n-D (Definition 1.13(i)). Thus in summary, (a) corresponds to a statement of zero coprimeness, (b) corresponds to a statement of minor coprimeness (and therefore zero coprimeness by Theorem 1.3(iv)), (c) corresponds to a statement of zero coprimeness and finally (d), with unit matrix as the right-hand-side, corresponds to zero coprimeness. The Bézout identity for minor coprimeness can be interpreted similarly to the 2-D interpretation, namely a 1-D Bézout identity over n generalised rings $F(z_1,\ldots,z_{i-1},z_{i+1},\ldots,z_n)[z_i]$ for $i = 1,2,\ldots,n$, followed by renormalisation. However, it has not been established whether a type of Bézout identity exists for factor coprime matrices. Indeed there is evidence in Chapter 2 that if a Bézout identity does exist for factor coprime matrices it does not explicitly involve a unit matrix on the right-hand-side.

To close this preliminary chapter consider fractions of polynomial matrices and the definition of singularities. A ratio of polynomial matrices can not be defined simply

[3] *This theorem was originally proved by two authors working independently in 1976, Quillen [41] and Suslin [42] and applied specifically using purely matrix theoretic arguments in [43]. A new and simple proof is given to this result in Theorem 2.3 and also a generalisation to minor coprimeness in Theorem 2.4.*

as a ratio of scalar polynomials due to the non-commutivity of matrices. The denominator scalar polynomial of a ratio of scalar polynomials can be viewed as the multiplicative inverse of a scalar polynomial, i.e. $b^{-1}(z)$ in (12). A polynomial matrix denominator can be defined as the inverse of a polynomial matrix: thus defining a left denominator, by multiplication on the left, and a right denominator, by multiplication on the right. Suppose $G_{p\times q}(z)$ is a matrix containing rational polynomial elements (called a *rational polynomial matrix*), then $G(z)$ may be written as

$$G(z) = N_1(z)D_1^{-1}(z) \tag{16}$$

$$= D_2^{-1}(z)N_2(z) \tag{17}$$

where $N_1(z)$, $N_2(z)$ are $p \times q$ polynomial matrices, $D_1(z)$ is a $q \times q$ polynomial matrix and $D_2(z)$ is a $p \times p$ polynomial matrix. If, additionally, the pairs $D_1(z)$, $N_1(z)$ and $D_2(z)$, $N_2(z)$ do not have any common matrix divisors, respectively right and left, the matrix fractions may be viewed as being irreducible in some sense. In this case the following definitions are accepted as defining the non-essential singularities.

Definition 1.14: (*Matrix Non-Essential Singularities*)
The singularities of the irreducible matrix fraction (16) are said to be

(i) *non-essential singularities of the first kind* if $|D_1(z)| = 0$ and the matrix

$$\begin{pmatrix} D_1(z) \\ N_1(z) \end{pmatrix}$$

possesses full rank;

(ii) *non-essential singularities of the second kind* if $|D_1(z)| = 0$ and the matrix

$$\begin{pmatrix} D_1(z) \\ N_1(z) \end{pmatrix}$$

does not possess full rank. □

Note: Similar statements can be made about a left fraction $D_2(z)$, $N_2(z)$ by considering $(D_2 \quad N_2)$.

Matrix fractions as defined by (16) and (17) are defined more fully and results concerning the coprimeness conditions that exist between the pair of matrices is the subject of Chapter 2.

Chapter 2

Matrix Fraction Descriptions

2.1 Introduction

The purpose of this chapter is to consider the matrix fraction description (MFD) of two- and n-dimensional systems. The matrix fraction description has been extensively employed in the past as a simple way of realising a one dimensional (1-D) system, given its transfer function matrix [14]. The process of forming a matrix fraction description arises from a generalisation of the representation of scalar systems in which the transfer function is a ratio of polynomials in a single indeterminate, see Chapter 1. These 1-D transfer functions are said to be reducible or irreducible depending on whether or not the two polynomials have a common polynomial factor (i.e. whether or

not they are relatively prime). Thus a reducible transfer function can be transformed into an irreducible fraction of polynomials, by cancelling the common polynomial factor between the numerator and denominator polynomials. This process of forming an irreducible denominator and numerator may similarly be extended to a rational matrix to define a coprime matrix fraction description.

The first difficulty in generalising the polynomial fraction for 1-D scalar systems to 1-D matrix systems is the representation of a matrix ratio. The matrix equivalent of polynomial division is imperspicuous due to the non-commutivity of matrix multiplication, since division is the inverse of scalar multiplication. Thus the denominator of a matrix fraction description, expressed as the inverse of a polynomial matrix, may be multiplied on either the left or on the right of the numerator matrix: this defines two types of matrix fraction description, namely the left MFD and the right MFD, respectively. This representation overcomes the problem of matrix division but introduces a new difficulty of determining whether a particular matrix fraction description can be described as being reducible or irreducible. The solution to this problem lies in polynomial matrix theory and the notion of coprimeness (or relative primeness). For 1-D and 2-D matrix fraction descriptions the determination of coprimeness is well defined and relatively simple theoretical procedures exist, e.g. by computing the high-order minors of the augmented matrix comprising the numerator and denominator matrices or by computing the Hermite form. However, for matrices in more than two indeterminates the notion of factor coprimeness can not be determined in this way. It is theoretically possible to determine whether two matrices are factor coprime but practically it is complicated and difficult. This difficulty has resulted in a limited number of factor coprime examples being available to demonstrate results obtained.

Even if it is known that the numerator and denominator matrices are not coprime the process of computing the common matrix factor is not a trivial matter. For 1-D matrices elementary operations may be employed to deliver the required matrix factor (the *greatest common divisor*), but this is not the case for matrices in more than one indeterminate. This is due to the underlying ring structure of the matrix elements. The ring $\mathbb{R}[z_1]$ is Euclidean and thus a division algorithm is defined, and all unimodular matrices can be written as a product of elementary matrices. The ring $\mathbb{R}[z_1, z_2]$ is non-Euclidean resulting in non-elementary unimodular matrices, called secondary matrices [5], being required to form a basis for unimodular decomposition.

A complicated process does exist for the computation of the greatest common divisor of matrices with elements in two indeterminates [33], [34] and is the subject of Part

two of this thesis, but for matrices with elements in more than two indeterminates the process is simply not evident.

Finally, before discussing matrix fraction descriptions in detail, one further point should be made concerning the computational feasibility of such descriptions. In the previous paragraph the question of coprimeness (and its determination) between a numerator and denominator matrix has been briefly addressed but the computational process has not been discussed. The precise details regarding the computational procedure will be given later in the chapter. It suffices to say that for every rational matrix in many indeterminates not only does there exist a numerator and denominator matrix constituting a matrix fraction description, but there also exist matrix fraction descriptions with conditions of coprimeness holding between the numerator and denominator matrices. The types of coprimeness that exist between the matrices is a main consideration of this exposition and will be given particular attention. One other result worthy of mention at this stage is the generalisation of the so called 2-D MFD Theorem [34] (which establishes a relation between the determinants of the denominator matrices of two indeterminate matrix fraction descriptions); here it is shown that the relation not only extends to the invariant polynomials of the denominator matrices but also to the invariant polynomials of the numerator matrices. This result is also valid for a certain class of matrix fraction descriptions in more than two indeterminates.

Matrix fraction descriptions in more than one indeterminate are considered in two separate sections: the first considers the case of two indeterminates and the second more than two indeterminates. This division is made because the two cases are essentially different due to the splitting of the notion of coprimeness (Definition 1.13 and Theorem 1.3).

2.2 2-D Matrix Fraction Descriptions

Matrix fraction descriptions of 2-D rational matrices involve polynomial numerator and denominator matrices with elements that are polynomial in two indeterminates, i.e. $\mathbb{R}[z_1, z_2]$.

Definition 2.1: (*General MFD*)
A *general MFD* of a 2-D rational matrix $G_{p\times q}(z_1, z_2)$ is given by

$$G(z_1, z_2) = D_1^{-1}(z_1, z_2)N_1(z_1, z_2) \qquad \text{left MFD} \qquad (18)$$

$$= N_2(z_1, z_2)D_2^{-1}(z_1, z_2) \qquad \text{right MFD} \qquad (19)$$

where $N_1(z_1, z_2)$, $N_2(z_1, z_2)$, $D_1(z_1, z_2)$ and $D_2(z_1, z_2)$ are polynomial matrices. □

This definition of a matrix fraction description may be restricted further by insisting that the numerator and denominator matrices are coprime. However, for polynomial matrices in two indeterminates there is not only one definition of coprimeness, as indicated by Chapter 1, but two (minor and zero). This defines two types of coprime matrix fraction description.

Definition 2.2: (*Coprime MFD*)
The factorisation of the rational matrix $G_{p\times q}(z_1, z_2)$ given by

$$G(z_1, z_2) = D^{-1}(z_1, z_2)N(z_1, z_2),$$

where $D_{p\times p}(z_1, z_2)$ and $N_{p\times q}(z_1, z_2)$ are polynomial matrices, is described as a:

(i) *minor (left) coprime MFD* if $D(z_1, z_2)$ and $N(z_1, z_2)$ are minor (left) coprime matrices;

(ii) *zero (left) coprime MFD* if $D(z_1, z_2)$ and $N(z_1, z_2)$ are zero (left) coprime matrices.

Right factorisation may similarly be defined by transposition of $G(z_1, z_2)$. □

Note: The term *coprime MFD* will be used to denote a matrix fraction description in which the numerator and denominator matrices are either minor coprime or zero coprime.

The first question arising from the definition concerns the existence of such coprime matrix fraction descriptions.

Theorem 2.1: *For every rational matrix $G_{p\times q}(z_1, z_2)$, in two indeterminates, there exists a coprime MFD.*
Proof: The proof is constructive and is given as the following algorithm.

Algorithm 2.1: (*Coprime MFD Algorithm*)

Let $G_{p\times q}(z_1, z_2)$ be a rational matrix in the two indeterminates z_1, z_2.

1. Form $d_i(z_1, z_2)$, the least common denominator of the ith row of $G_{p\times q}(z_1, z_2)$, for $i = 1, 2, \ldots, p$.
2. Construct the diagonal matrix $D_{p\times p}(z_1, z_2)$,
$$D(z_1, z_2) = \begin{pmatrix} d_1(z_1, z_2) & 0 & \cdots & 0 \\ 0 & d_2(z_1, z_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_p(z_1, z_2) \end{pmatrix}.$$
3. Form the polynomial matrix $N_{p\times q}(z_1, z_2) = D_{p\times p}(z_1, z_2)G_{p\times q}(z_1, z_2)$.
4. Compute $Q(z_1, z_2)$, the greatest common (left) divisor of $D(z_1, z_2)$ and $N(z_1, z_2)$, i.e. compute $Q(z_1, z_2)$ such that
$$[\,D(z_1, z_2) \quad N(z_1, z_2)\,] = Q(z_1, z_2)\,[\,\bar{D}(z_1, z_2) \quad \bar{N}(z_1, z_2)\,]$$
where $\bar{D}(z_1, z_2)$ and $\bar{N}(z_1, z_2)$ are minor (left) coprime. Then
$$\begin{aligned} G(z_1, z_2) &= D^{-1}(z_1, z_2)N(z_1, z_2) \\ &= (\bar{D}(z_1, z_2)^{-1}Q^{-1}(z_1, z_2))(Q(z_1, z_2)\bar{N}(z_1, z_2)) \\ &= \bar{D}(z_1, z_2)^{-1}\bar{N}(z_1, z_2) \end{aligned}$$
is a minor (left) coprime MFD. Additionally $\bar{D}(z_1, z_2)$, $\bar{N}(z_1, z_2)$ may also be zero (left) coprime, depending on the nature of $G(z_1, z_2)$. □

Steps 1–3 are easily computed. However, as mentioned in the introduction, the calculation of the greatest common divisor of 2-D polynomial matrices is not a trivial matter. Part Two details a computer program to calculate automatically this greatest common divisor.

To illustrate the algorithm the left and right coprime matrix fraction descriptions of a 2-D rational matrix are given in the following example.

Example 2.1: Consider forming the left MFD of the rational matrix $G(z_1, z_2)$
$$G(z_1, z_2) = \begin{pmatrix} z_1 & z_2 + 1 \\ \dfrac{z_1 + z_2}{z_2} & \dfrac{z_2}{z_1} \\ \dfrac{z_1}{z_2} & 1 \end{pmatrix}.$$

By Step 1. $d_1 = 1$, $d_2 = z_1 z_2$, $d_3 = z_2$, thus

$$D(z_1, z_2) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & z_1 z_2 & 0 \\ 0 & 0 & z_2 \end{pmatrix}$$

By Step 2.

$$N(z_1, z_2) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & z_1 z_2 & 0 \\ 0 & 0 & z_2 \end{pmatrix} \begin{pmatrix} z_1 & z_2 + 1 \\ \frac{z_1 + z_2}{z_2} & \frac{z_2}{z_1} \\ \frac{z_1}{z_2} & 1 \end{pmatrix}$$
$$= \begin{pmatrix} z_1 & z_2 + 1 \\ z_1(z_1 + z_2) & z_2^2 \\ z_1 & z_2 \end{pmatrix}$$

The high-order minors of $M(z_1, z_2) = [D(z_1, z_2)\ N(z_1, z_2)]$ are

$$M_{1,2,3}^{1,2,3} = z_1 z_2^2 \qquad M_{1,2,4}^{1,2,3} = z_1^2 z_2$$
$$M_{1,2,5}^{1,2,3} = z_1 z_2^2 \qquad M_{1,3,4}^{1,2,3} = -z_1 z_2 (z_1 + z_2)$$
$$M_{1,3,5}^{1,2,3} = -z_2^3 \qquad M_{1,4,5}^{1,2,3} = z_1^2 z_2$$
$$M_{2,3,4}^{1,2,3} = z_1^2 z_2^2 \qquad M_{2,3,5}^{1,2,3} = z_1 z_2^2 (z_2 + 1)$$
$$M_{2,4,5}^{1,2,3} = z_1^2 z_2 \qquad M_{3,4,5}^{1,2,3} = -z_1 z_2 (z_1 z_2 + z_1 + z_2)$$

A greatest common divisor of the high-order minors is z_2, therefore M is not coprime. A greatest common divisor of $D(z_1, z_2)$, $N(z_1, z_2)$ is given by

$$Q(z_1, z_2) = \begin{pmatrix} 1 & 1 & 0 \\ 0 & z_1 & z_2 \\ 0 & 1 & 0 \end{pmatrix}$$

Thus the (left) coprime MFD is given by

$$G(z_1, z_2) = \bar{D}^{-1}(z_1, z_2)\bar{N}(z_1, z_2) = \begin{pmatrix} 1 & 0 & -z_2 \\ 0 & 0 & z_2 \\ 0 & z_1 & -z_1 \end{pmatrix}^{-1} \begin{pmatrix} 0 & 1 \\ z_1 & z_2 \\ z_1 & z_2 - z_1 \end{pmatrix}.$$

The high-order minors of $[\bar{D}(z_1, z_2)\ \bar{N}(z_1, z_2)]$ are given by the high-order minors of M divided by the determinant of $Q(z_1, z_2)$, i.e. $-z_2$. Therefore $\bar{D}(z_1, z_2)$, $\bar{N}(z_1, z_2)$ forms a minor (left) coprime matrix fraction description.

The right coprime MFD may be calculated by using the above procedure on $G^T(z_1,$

z_2). Thus

$$G^T(z_1,z_2) = \begin{pmatrix} z_1 & \frac{z_1+z_2}{z_2} & \frac{z_1}{z_2} \\ z_2+1 & \frac{z_2}{z_1} & 1 \end{pmatrix}$$

$$= D^{-1}(z_1,z_2)N(z_1,z_2) = \begin{pmatrix} z_2 & 0 \\ 0 & z_1 \end{pmatrix}^{-1} \begin{pmatrix} z_1z_2 & z_1+z_2 & z_1 \\ (z_2+1)z_1 & z_2 & z_1 \end{pmatrix}$$

The high-order minors of the compound matrix $M = [D\ N]$ are given by

$$M_{1,2}^{1,2} = z_1z_2 \qquad M_{1,3}^{1,2} = z_1z_2(z_2+1)$$

$$M_{1,4}^{1,2} = z_2^2 \qquad M_{1,5}^{1,2} = z_1z_2$$

$$M_{2,3}^{1,2} = -z_1^2z_2 \qquad M_{2,4}^{1,2} = -z_1(z_1+z_2)$$

$$M_{2,5}^{1,2} = -z_1^2 \qquad M_{3,4}^{1,2} = -z_1(z_1z_2+z_1+z_2)$$

$$M_{3,5}^{1,2} = -z_1^2 \qquad M_{4,5}^{1,2} = z_1^2$$

Therefore $D(z_1,z_2)$, $N(z_1,z_2)$ form a minor coprime pair and the right coprime MFD is given by

$$G(z_1,z_2) = \begin{pmatrix} z_1z_2 & (z_2+1)z_1 \\ z_1+z_2 & z_2 \\ z_1 & z_1 \end{pmatrix} \begin{pmatrix} z_2 & 0 \\ 0 & z_1 \end{pmatrix}^{-1} \qquad \square$$

Thus by using Algorithm 2.1 a left and a right minor coprime MFD has been formed. This is the type of coprimeness that is expected from the algorithm since only the greatest common divisor is removed from both the numerator and denominator matrices: thus constructing a minor coprime pair of matrices. Two interesting points arise from this example:

1. both the left and right MFD have the same type of coprimeness;
2. there does not exist a zero coprime MFD of the rational matrix $G(z_1,z_2)$ in Example 2.1 by using this algorithm. Only unimodular factors can be removed from a minor coprime pair, which does not alter the coprimeness of the two matrices.

Therefore two questions arising from these observations are

1. Is the type of coprimeness for all MFDs invariant?
2. Is there a process by which zero coprime MFDs can be formed for all 2-D rational matrices?

These questions will be addressed in the following section.

2.2.1 Coprime Invariance of 2-D MFDs

The answer to the first question posed above is provided by the following theorem and its corollaries.

Theorem 2.2: *If the 2-D rational matrix $G_{p\times q}(z_1, z_2)$ possesses one MFD which is zero coprime then every coprime MFD of $G(z_1, z_2)$ is zero coprime.*

Proof: Without loss of generality suppose $G(z_1, z_2)$ has a zero (left) coprime MFD. Let

$$G(z_1, z_2) = D_1^{-1}(z_1, z_2)N_1(z_1, z_2) \tag{20}$$

be the zero (left) coprime MFD, i.e. $D_1(z_1, z_2)$, $N_1(z_1, z_2)$ are zero (left) coprime. Suppose that $G(z_1, z_2)$ has a minor (right) coprime MFD, which is not additionally zero coprime, given by

$$G(z_1, z_2) = N_2(z_1, z_2)D_2^{-1}(z_1, z_2). \tag{21}$$

By the Bézout identities, Theorem 1.4, there exist polynomial matrices $X(z_1, z_2)$, $Y(z_1, z_2)$ such that

$$D_1X + N_1Y = I_p \tag{22}$$

and polynomial matrices $X_i(z_1, z_2)$, $Y_i(z_1, z_2)$ and polynomials $\psi_i(z_i)$ for $i = 1, 2$, such that

$$X_iN_2 + Y_iD_2 = \psi_i(z_i)I_q. \tag{23}$$

Thus (20)–(23) may be written as

$$\begin{pmatrix} D_1 & N_1 \\ -X_i & Y_i \end{pmatrix}\begin{pmatrix} X & -N_2 \\ Y & D_2 \end{pmatrix} = \begin{pmatrix} I_p & 0 \\ W_i & \psi_i I_q \end{pmatrix} \tag{24}$$

where $W_i = Y_iY - X_iX$. Let

$$U_i \equiv \begin{pmatrix} D_1 & N_1 \\ -X_i & Y_i \end{pmatrix} \qquad V \equiv \begin{pmatrix} X & -N_2 \\ Y & D_2 \end{pmatrix} \qquad i = 1, 2. \tag{25}$$

Then from (24)

$$|U_i||V| = \psi_i^q(z_i) \quad \text{for} \quad i = 1, 2. \tag{26}$$

Hence $|V| \neq 0$ and so $V(z_1, z_2)$ is non-singular. Further, the condition (26) for $i = 1$, indicates that $|V|$ is polynomial in z_1 alone, while the same equation for $i = 2$ shows that $|V|$ is polynomial in z_2 alone. Consequently $|V|$ is constant and so $V(z_1, z_2)$ is unimodular.

Now since $D_2(z_1, z_2)$, $N_2(z_1, z_2)$ are not zero (right) coprime by the initial premise, there exists (z_1^0, z_2^0) such that

$$V(z_1^0, z_2^0) = \begin{pmatrix} X(z_1^0, z_2^0) & -N_2(z_1^0, z_2^0) \\ Y(z_1^0, z_2^0) & D_2(z_1^0, z_2^0) \end{pmatrix}$$

has at least two linearly dependent columns, thus $|V(z_1^0, z_2^0)| = 0$ contradicting the unimodularity of V. Therefore $D_2(z_1, z_2)$, $N_2(z_1, z_2)$ are zero (right) coprime.

Thus if one left coprime MFD is zero coprime then all right coprime MFDs are zero coprime. The above arguments may then be repeated for one of these right coprime MFDs (which is necessarily zero coprime) to deduce that all left coprime MFDs are zero coprime.

Hence if one coprime MFD of $G(z_1, z_2)$ is zero coprime then every coprime MFD of $G(z_1, z_2)$ is zero coprime. □

Corollary 1: *If $G(z_1, z_2)$ has one MFD which is zero coprime then there does not exist a minor coprime MFD, which is not simultaneously zero coprime.*

Proof: This is a restatement of the theorem. □

Corollary 2: *If $G(z_1, z_2)$ has one MFD which is minor coprime but not zero coprime then there does not exist a zero coprime MFD.*

Proof: This follows from the theorem because if there exists a zero coprime MFD the above theorem ensures that all MFDs are zero coprime. □

The theorem and its corollaries indicate that the MFDs of a given $G(z_1, z_2)$ are all of the same coprimeness type. Thus either $G(z_1, z_2)$ has all its MFDs of the zero coprime type or else they are all of the minor coprime type, without being zero coprime. These findings are confirmed by the following example.

Example 2.2: Consider the left coprime MFD of

$$G(z_1, z_2) = \begin{pmatrix} \frac{z_2}{z_1} & 0 \\ 0 & \frac{z_1}{z_2} \end{pmatrix}$$

$$D^{-1}(z_1, z_2)N(z_1, z_2) = \begin{pmatrix} z_1 & 0 \\ 0 & z_2 \end{pmatrix}^{-1} \begin{pmatrix} z_2 & 0 \\ 0 & z_1 \end{pmatrix}$$

The high-order minors of $M = [D\ N]$ are given by

$$M_{1,2}^{1,2} = z_1 z_2 \qquad M_{1,3}^{1,2} = 0$$

$$M_{1,4}^{1,2} = z_1^2 \qquad M_{2,3}^{1,2} = -z_2^2$$

$$M_{2,4}^{1,2} = 0 \qquad M_{3,4}^{1,2} = z_1 z_2$$

Therefore $D^{-1}(z_1, z_2)N(z_1, z_2)$ is a minor (left) coprime MFD of $G(z_1, z_2)$. If it is additionally possible to form a zero (left) coprime MFD then there exists $Q(z_1, z_2)$ such that

$$[\, D(z_1, z_2) \quad N(z_1, z_2)\,] = Q(z_1, z_2)\,[\, \bar{D}(z_1, z_2) \quad \bar{N}(z_1, z_2)\,]$$

and $\bar{D}(z_1, z_2)$, $\bar{N}(z_1, z_2)$ are zero (left) coprime. However, $D(z_1, z_2)$, $N(z_1, z_2)$ are minor (left) coprime, therefore all left matrix divisors are unimodular, i.e. $Q(z_1, z_2)$ is unimodular. Thus $\bar{D}(z_1, z_2)$, $\bar{N}(z_1, z_2)$ possess the same notion of coprimeness as $D(z_1, z_2)$, $N(z_1, z_2)$, i.e minor (left) coprimeness. Hence there does not exist a zero (left) coprime MFD of $G(z_1, z_2)$. □

It has been previously noted, following Algorithm 2.1, that in constructing one MFD of a given rational matrix $G(z_1, z_2)$ only minor coprimeness can be ensured. Whether a MFD is, in addition, zero coprime is seen to be a property of $G(z_1, z_2)$ itself rather than of the particular representation of $G(z_1, z_2)$. Thus Algorithm 2.1, which in theory ensures minor coprime representations, practically delivers zero coprime representations when appropriate.

This theorem (2.2) may be employed to give a new and simple proof of the polynomial matrix form of the Completion Theorem [43] and [72].

Theorem 2.3: (*Completion Theorem*)
The two polynomial matrices $D_{p\times p}(z_1, z_2)$, $N_{p\times q}(z_1, z_2)$ may be incorporated as the first p rows of a $(p+q)\times(p+q)$ unimodular matrix if and only if $D(z_1, z_2)$, $N(z_1, z_2)$ are zero (left) coprime.

Proof: Firstly assume that $D(z_1, z_2)$, $N(z_1, z_2)$ are zero (left) coprime and without loss of generality $D(z_1, z_2)$ is invertible. By the above theorem there exist zero (right) coprime polynomial matrices $D'_{q\times q}(z_1, z_2)$, $N'_{p\times q}(z_1, z_2)$ such that

$$D^{-1}(z_1, z_2)N(z_1, z_2) = N'(z_1, z_2)D'^{-1}(z_1, z_2)$$

and by the Bézout identity

$$D(z_1, z_2)X(z_1, z_2) + N(z_1, z_2)Y(z_1, z_2) = I_p$$
$$X'(z_1, z_2)D'(z_1, z_2) + Y'(z_1, z_2)N'(z_1, z_2) = I_q.$$

Thus

$$\begin{pmatrix} D & N \\ -Y' & X' \end{pmatrix}\begin{pmatrix} X & -N' \\ Y & D' \end{pmatrix} = \begin{pmatrix} I_p & 0 \\ W & I_q \end{pmatrix}$$

where $W = X'Y - Y'X$. Define the polynomial matrices $U(z)$ and $V(z_1, z_2)$ by

$$U = \begin{pmatrix} D & N \\ -Y' & X' \end{pmatrix}, \qquad V = \begin{pmatrix} X & -N' \\ Y & D' \end{pmatrix}.$$

Then

$$|U||V| = 1$$

Hence $U(z_1, z_2)$, $V(z_1, z_2)$ are unimodular matrices.

Conversely, assume that there exist polynomial matrices $X(z_1, z_2)$, $Y(z_1, z_2)$ such that $U(z_1, z_2)$ is unimodular, where

$$U = \begin{pmatrix} D & N \\ X & Y \end{pmatrix}.$$

Now suppose that there exist $(z_1^{(0)}, z_2^{(0)})$ such that the high-order minors of

$$\left(D(z_1^{(0)}, z_2^{(0)}), N(z_1^{(0)}, z_2^{(0)})\right)$$

are all identically equal to zero. Therefore $|U(z_1^{(0)}, z_2^{(0)})| = 0$ and a contradiction is obtained. □

Thus a zero (left) coprime matrix pair may be characterised by row bordering to a unimodular matrix and conversely. This fact has already been noted in Chapter 1 during the discussion of Rosenbrock's relative primeness conditions, Theorem 1.2, and the relation to coprimeness in many indeterminates, Section 1.5. The matrix theoretic proof of this theorem has been previously provided by Zak *et al* [43] but the fact that if one MFD of a rational 2-D polynomial matrix is zero coprime then all MFDs are zero coprime is not made, which is central to the proof. Therefore the proof given here is more rigorous and complete than that given by Zak *et al.*

The natural extension to this is to consider the situation for a minor coprime pair of matrices. By comparison with the zero coprime case and the nature of the Bézout identity it is expected that more than one row bordering statement characterises minor (left) coprimeness. Thus the following statement can be proved.

Theorem 2.4: *The polynomial matrix pair $D_{p\times p}(z_1, z_2)$, $N_{p\times q}(z_1, z_2)$ are minor (left) coprime if and only if there exist polynomial matrices $X_i(z_1, z_2)$, $Y_i(z_1, z_2)$ for $i = 1, 2$ such that the determinants of the $(p+q) \times (p+q)$ matrices $U_1(z_1, z_2)$ and $U_2(z_1, z_2)$ are polynomial in z_1 alone and z_2 alone, respectively, where*

$$U_i(z_1, z_2) = \begin{pmatrix} D(z_1, z_2) & N(z_1, z_2) \\ X_i(z_1, z_2) & Y_i(z_1, z_2) \end{pmatrix} \qquad \text{for} \qquad i = 1, 2.$$

Proof: Firstly, assume that $D(z_1, z_2)$, $N(z_1, z_2)$ are minor (left) coprime and with out loss of generality $D(z_1, z_2)$ is non-singular. Therefore by the corollary to Theorem

2.2 there exist minor (right) coprime polynomial matrices $D'_{q\times q}(z_1,z_2)$, $N'_{p\times q}(z_1,z_2)$ such that

$$D^{-1}(z_1,z_2)N(z_1,z_2) = N'(z_1,z_2)D'^{-1}(z_1,z_2)$$

and by the Bézout identities

$$D(z_1,z_2)X_i(z_1,z_2) + N(z_1,z_2)Y_i(z_1,z_2) = \phi_i(z_i)I_p$$
$$W'_i(z_1,z_2)D'(z_1,z_2) + Z'_i(z_1,z_2)N'(z_1,z_2) = \psi_i(z_i)I_q.$$

for $i = 1,2$ and where $\psi_i(z_i)$, $\phi_i(z_i)$ are polynomials in z_i for $i = 1,2$. Thus

$$\begin{pmatrix} D & N \\ -Z'_i & W'_i \end{pmatrix}\begin{pmatrix} X_i & -N' \\ Y_i & D' \end{pmatrix} = \begin{pmatrix} \phi_i(z_i)I_p & 0 \\ J_i & \psi_i(z_i)I_q \end{pmatrix}$$

where $J_i = W'_iY_i - Z'_iX_i$ for $i = 1,2$. Define, for $i = 1,2$, the polynomial matrices $U_i(z_1,z_2)$, $V_i(z_1,z_2)$

$$U_i = \begin{pmatrix} D & N \\ -Z'_i & W'_i \end{pmatrix}, \qquad V_i = \begin{pmatrix} X_i & -N' \\ Y_i & D' \end{pmatrix}$$

Thus

$$|U_i(z_1,z_2)||V_i(z_1,z_2)| = \phi_i^p(z_i)\psi_i^q(z_i)$$

for $i = 1,2$. Therefore the determinants of $U_1(z_1,z_2)$ and $U_2(z_1,z_2)$ are polynomial in z_1 alone and z_2 alone, respectively.

Conversely, suppose that there exist polynomial matrices $X_i(z_1,z_2)$, $Y_i(z_1,z_2)$ for $i = 1,2$ such that the determinants of $|U_i(z_1,z_2)| = \alpha_i(z_i)$, where $\alpha_i(z_i)$ are polynomials in z_1 alone and z_2 alone, respectively, and

$$U_i = \begin{pmatrix} D & N \\ X_i & Y_i \end{pmatrix}.$$

Assume that $D(z_1,z_2)$, $N(z_1,z_2)$ are not minor (left) coprime therefore there exist polynomial matrices $Q(z_1,z_2)$, $D'(z_1,z_2)$ and $N'(z_1,z_2)$ such that

$$\begin{pmatrix} D & N \\ X_i & Y_i \end{pmatrix} = \begin{pmatrix} Q & 0 \\ 0 & I \end{pmatrix}\begin{pmatrix} D' & N' \\ X_i & Y_i \end{pmatrix}.$$

Taking determinants of this equation and using the premise yields

$$|Q|\begin{vmatrix} D' & N' \\ X_i & Y_i \end{vmatrix} = \alpha_i(z_i) \qquad \text{for} \qquad i = 1,2.$$

Therefore $|Q(z_1,z_2)|$ is a constant, i.e. $Q(z_1,z_2)$ is a unimodular matrix, contradicting the fact that $D(z_1,z_2)$, $N(z_1,z_2)$ are not minor (left) coprime. Hence the result is obtained. □

By comparison with the third characterisation of 1-D relative primeness (Theorem 1.2) Theorem 2.4 provides a generalised Completion Theorem for minor coprimeness.

Note: The characterisations of zero and minor coprimeness provided by Theorems 2.3 and 2.4 reflect the form of the Bézout identities. Thus reinforcing the theoretical importance of the Bézout identities, noted in Chapter 1.

2.2.2 2-D MFD Structure Theorem

In the previous section the external structure of matrix fraction descriptions has been investigated by considering the types of coprime MFD that exist for any 2-D rational matrix. Now the internal structure of the matrix fraction description is investigated.

It is well known [32]–[34], [44] that the determinants of the denominator matrices of matrix fraction descriptions, of the same 2-D rational matrix, are equal modulo a multiplicative constant: this is often known as the *2-D MFD Theorem.* In fact it is not generally recognised that this property is shared by all the invariant polynomials of not only the denominator matrices but also the numerator matrices: the proof of this statement is the main subject of this section and constitutes one of the main results of this thesis.

Before proving the this result recall:

Lemma 2.1: *(2-D MFD Theorem)*
Let $G_{p\times q}(z_1, z_2)$ be a rational matrix with MFDs defined by

$$G(z_1, z_2) = D_1^{-1} N_1 = N_2 D_2^{-1} \tag{27}$$

where $D_1(z_1, z_2)$, $N_1(z_1, z_2)$ are minor (left) coprime and $D_2(z_1, z_2)$, $N_2(z_1, z_2)$ are minor (right) coprime. Then

$$\det D_1 = \text{const} \times \det D_2 \tag{28}$$

Proof: The proof is contained in [33]. □

The following theorem describes more deeply the structure of not only the denominator matrices but also the numerator matrices between given matrix fraction descriptions.

Theorem 2.5: *(2-D MFD Structure Theorem)*
Let $G_{p\times q}(z_1, z_2)$ be a rational matrix and have a MFD defined by

$$\begin{aligned} G(z_1, z_2) &= N_1(z_1, z_2)D_1^{-1}(z_1, z_2) \\ &= D_2^{-1}(z_1, z_2)N_2(z_1, z_2) \end{aligned} \tag{29}$$

where $N_1(z_1, z_2)$, $D_1(z_1, z_2)$ are minor (right) coprime and $N_2(z_1, z_2)$, $D_2(z_1, z_2)$ are minor (left) coprime.

(i) Let $d_1^{[1]}(z_1, z_2), d_2^{[1]}(z_1, z_2), \ldots, d_q^{[1]}(z_1, z_2)$ denote the invariant polynomials of the $q \times q$ polynomial matrix $D_1(z_1, z_2)$ and $d_1^{[2]}(z_1, z_2), d_2^{[2]}(z_1, z_2), \ldots, d_p^{[2]}(z_1, z_2)$ denote the invariant polynomials of $p \times p$ polynomial matrix $D_2(z_1, z_2)$ then

$$d_{q-i}^{[1]} = c_i d_{p-i}^{[2]} \quad \text{for} \quad i = 0, 1, \ldots, \max(p-1, q-1)$$

where $d_j^{[1]} = 1$, $d_j^{[2]} = 1$ for $j < 1$ and $c_i \in \mathbb{R}\backslash\{0\}$.

(ii) The $p\times q$ polynomial matrices $N_1(z_1, z_2)$ and $N_2(z_1, z_2)$ have identical invariant polynomials, modulo a non-zero constant factor.

Proof: Since $N_1(z_1, z_2)$, $D_1(z_1, z_2)$ are minor (right) coprime and $N_2(z_1, z_2)$, $D_2(z_1, z_2)$ are minor (left) coprime there exist polynomial matrices $X_i(z_1, z_2)$, $Y_i(z_1, z_2)$, $W_i(z_1, z_2)$ and $Z_i(z_1, z_2)$ for $i = 1, 2$ of appropriate dimensions such that

$$\left.\begin{aligned} X_i D_1 + Y_i N_1 &= \psi_i(z_i) I_q \\ N_2 Z_i + D_2 W_i &= \phi_i(z_i) I_p \end{aligned}\right\} \quad \text{for} \quad i = 1, 2 \tag{30}$$

where $\psi_i(z_i)$ and $\phi_i(z_i)$ are polynomials. From (29) and (30) it follows that

$$\begin{bmatrix} X_i & Y_i \\ N_2 & -D_2 \end{bmatrix} \begin{bmatrix} D_1 & Z_i \\ N_1 & -W_i \end{bmatrix} = \begin{bmatrix} \psi_i(z_i) I_q & J_i \\ 0 & \phi_i(z_i) I_p \end{bmatrix} \tag{31}$$

where $J_i = X_i Z_i - Y_i W_i$. Now take $i = 1$ and replace $\begin{bmatrix} Z_i \\ -W_i \end{bmatrix}$ with $\begin{bmatrix} 0 \\ I \end{bmatrix}$. Then (31) gives

$$\begin{bmatrix} X_1 & Y_1 \\ N_2 & -D_2 \end{bmatrix} \begin{bmatrix} D_1 & 0 \\ N_1 & I_p \end{bmatrix} = \begin{bmatrix} \psi_1 I_q & Y_1 \\ 0 & -D_2 \end{bmatrix}. \tag{32}$$

For any matrix Q let $Q_{j_1,\ldots,j_h}^{i_1,\ldots,i_k}$ denote the $k \times h$ submatrix formed from rows $i_1, \ldots, i_k$ and columns $j_1, \ldots, j_h$. Consider the following $(q+k) \times (q+k)$ submatrix formed from (32)

$$\underbrace{\begin{bmatrix} X_1 & Y_1 \\ N_{2\ i_1,\ldots,i_k} & -D_{2\ i_1,\ldots,i_k} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} D_1 & 0_{q\times k} \\ N_1 & I_{p\ j_1,\ldots,j_k} \end{bmatrix}}_{B} = \begin{bmatrix} \psi_1 I_q & Y_{1\ j_1,\ldots,j_k} \\ 0_{k\times q} & -D_{2\ j_1,\ldots,j_k}^{i_1,\ldots,i_k} \end{bmatrix}. \tag{33}$$

Take determinants of both sides and use the Cauchy-Binet Theorem [28] and (33) to show that

$$\sum_{1\le l_1<\cdots<l_{q+k}\le q+p} |A^{1,\ldots,q+k}_{l_1,\ldots,l_{q+k}}|\,|B^{l_1,\ldots,l_{q+k}}_{1,\ldots,q+k}| = -\psi_1^q |D_2{}^{i_1,\ldots,i_k}_{j_1,\ldots,j_k}|. \tag{34}$$

Now the form of B indicates that any factor of B of the type occurring in the left-hand-side of (34), for which $\{q+j_i,\ldots q+j_k\}$ is not a subset of $\{l_1,\ldots,l_{q+k}\}$ is zero. Thus all the non-zero minors of B which occur in the left-hand-side of (34) contain the rows $q+j_1,\ldots,q+j_k$. Such a factor is then expressible via Laplace expansion in terms of minors of N_1 and D_1. The smallest minor of D_1 occurring in this Laplace expansion is of order $q+k-p$. Therefore if $g_i^{[1]}(z_1,z_2)$ for $i=1,\ldots,q$ denotes the greatest common divisor of the $i\times i$ minors D_1, it follows that

$$g^{[1]}_{q+k-p} | \psi_1^q | D_2{}^{i_1,\ldots,i_k}_{j_1,\ldots,j_k}| \tag{35}$$

where $g^{[1]}_{q+k-p}=1$ if $q+k-p\le 0$.

If then $g_i^{[2]}(z_1,z_2)$ for $i=1,\ldots,p$ denotes the greatest common divisor of the $i\times i$ minors D_2, it follows from (35) and the fact that $i_1,\ldots,i_k$ and $j_1,\ldots,j_k$ are arbitrary that

$$g^{[1]}_{q+k-p} | \psi_1^q g_k^{[2]}. \tag{36}$$

On the other hand if we take $i=2$ then the same argument shows that

$$g^{[1]}_{q+k-p} | \psi_2^q g_k^{[2]} \quad k=1,\ldots,p. \tag{37}$$

Statements (36) and (37) then imply, by Theorem 1.1, since ψ_1, ψ_2 are factor coprime

$$g^{[1]}_{q+k-p} | g_k^{[2]}, \quad k=1,\ldots,p$$

or, on writing $k=p-j$,

$$g^{[1]}_{q-j} | g^{[2]}_{p-j}, \quad j=0,\ldots,\max(p-1,q-1) \tag{38}$$

where, if necessary, $g_j^{[1]}=1$, $g_j^{[2]}=1$ for $j<1$.

Now in (31) replace $[X_i\ Y_i]$ with $[I_q\ 0]$ to give

$$\begin{bmatrix} I_q & 0 \\ N_2 & -D_2 \end{bmatrix} \begin{bmatrix} D_1 & Z_1 \\ N_1 & -W_1 \end{bmatrix} = \begin{bmatrix} D_1 & Z_1 \\ 0 & \phi_1 I_p \end{bmatrix}. \tag{39}$$

The same argument surrounding (32) may now be used in the case of (39) to show that

$$g^{[2]}_{p-j} | g^{[1]}_{q-j}, \quad j=0,\ldots,\max(p-1,q-1). \tag{40}$$

Statements (38) and (40) then yield, modulo a constant non-zero factor,

$$g^{[1]}_{q-j}(z_1,z_2) = g^{[2]}_{p-j}(z_1,z_2), \quad j=0,\ldots,\max(p-1,q-1) \tag{41}$$

Now $g_h^{[1]}(z_1, z_2)$, $g_k^{[2]}(z_1, z_2)$ are the determinantal divisors of $D_1(z_1, z_2)$, $D_2(z_1, z_2)$ respectively, and so from the relationship between the determinantal divisors and their invariant polynomials the result (i) follows.

In the case of the numerators $N_1(z_1, z_2)$ and $N_2(z_1, z_2)$ the argument presented above will carry through with some minor modifications. Specifically, in the case $p \leq q$ for example, the equation corresponding to (32), for $i = 1$, is

$$\begin{bmatrix} X_1 & Y_1 \\ N_2 & -D_2 \end{bmatrix} \begin{bmatrix} D_1 & E \\ N_1 & 0 \end{bmatrix} = \begin{bmatrix} \psi_1 I_q & X_{1\ l_1,\ldots,l_p} \\ 0 & N_{2\ l_1,\ldots,l_p} \end{bmatrix} \tag{42}$$

where the constant matrix $E_{q\times p}$ is the unit matrix I_p with $q - p$ zero rows to form a $q \times p$ matrix and $l_1, \ldots, l_p$ correspond to the columns of the matrices N_2 and X_1 selected by multiplication by E. The analogue of (33) is obtained by selecting rows $i_1, i_2, \ldots, i_k$ from the second block row and columns $j_1, j_2, \ldots, j_k$ from the second block column. By considering all combinations of the rows of $E_{j_1,j_2,\ldots,j_k}$ and the form of the second matrix on the right-hand-side of (42), i.e the only non-zero minors are those involving rows $j_1, j_2, \ldots, j_k$ of the first block row, it is seen by taking all $1 \leq i_1 < i_2 < \cdots < i_k \leq p$ and $1 \leq j_1 < j_2 < \cdots < j_k \leq q$ that

$$h_k^{[1]} | \psi_1^q h_k^{[2]}$$

where $h_k^{[1]}$, $h_k^{[2]}$ are the greatest common divisors of the $k \times k$ order minors of $N_1(z_1, z_2)$, $N_2(z_1, z_2)$ respectively. Also by considering $i = 2$

$$h_k^{[1]} | \psi_2^q h_k^{[2]}.$$

Therefore, by similar reasoning surrounding (38)

$$h_k^{[1]} | h_k^{[2]} \quad \text{for} \quad k = 1, \ldots, p.$$

Now the equation corresponding to (39) is

$$\begin{bmatrix} 0 & E \\ N_2 & -D_2 \end{bmatrix} \begin{bmatrix} D_1 & Z_1 \\ N_1 & -W_1 \end{bmatrix} = \begin{bmatrix} N_{1\ l_1,\ldots,l_p} & W_{1\ l_1,\ldots,l_p} \\ 0 & \phi_1 I_p \end{bmatrix}$$

where E and $l_1, \ldots, l_p$ are defined in (42). Thus the result

$$h_k^{[2]} | h_k^{[1]} \quad \text{for} \quad k = 1, \ldots, p$$

is obtained by a similar discussion to that following (42). Therefore

$$h_k^{[1]} = c_k h_k^{[2]} \quad \text{for} \quad k = 1, \ldots, p$$

where $c_k \in \mathbb{R}\backslash\{0\}$ for $k = 1, \ldots, p$ and (ii) is established. □

It is thus seen from the above that all minor coprime MFDs of a given rational matrix $G(z_1, z_2)$ are such that they have identical numerator invariant polynomials and identical denominator non-unit invariant polynomials.

The well known 2-D MFD Theorem (Lemma 2.1) may now be readily shown as a particular case of the 2-D Structure Theorem as follows.

Corollary 1: *The determinants of the denominator matrices in (29) are equal modulo a multiplicative constant.*

Proof: From the proof of the theorem it has been shown in (41) that the determinantal divisors of the denominator matrices are equal modulo a multiplicative constant, in particular

$$g_q^{[1]} = c_0 g_p^{[2]} \tag{43}$$

Since $D_1(z_1, z_2)$ and $D_2(z_1, z_2)$ are $q \times q$ and $p \times p$ polynomial matrices respectively the greatest common divisor of the qth and pth order minors, respectively, are the determinants of $D_1(z_1, z_2)$ and $D_2(z_1, z_2)$. □

This theorem also demonstrates the necessity part of Theorem 2.4.

Corollary 2: *Defining $G_{p\times q}(z_1, z_2)$ as in (29) with the Bézout identities (30) and also*

$$U_i = \begin{bmatrix} X_i & Y_i \\ N_2 & -D_2 \end{bmatrix} \quad , \qquad V_i = \begin{bmatrix} D_1 & Z_i \\ N_1 & -W_i \end{bmatrix} \quad \text{for} \quad i = 1, 2. \tag{44}$$

Then

$$|U_i| = k_i \psi_i^q \quad , \qquad |V_i| = l_i \phi_i^p \quad \text{where} \quad k_i, l_i \in \mathbb{R}\backslash\{0\} \quad \text{for} \quad i = 1, 2.$$

Proof: From the proof of the theorem

$$U_i(z_1, z_2) V_i(z_1, z_2) = \begin{bmatrix} \psi_i(z_i^c) I_q & J_i \\ 0 & \phi_i(z_i^c) I_p \end{bmatrix} \tag{45}$$

where $J_i = X_i Z_i - Y_i W_i$. Now replace $\begin{bmatrix} Z_i \\ -W_i \end{bmatrix}$ with $\begin{bmatrix} 0 \\ I_p \end{bmatrix}$ to give

$$\begin{bmatrix} X_i & Y_i \\ N_2 & -D_2 \end{bmatrix} \begin{bmatrix} D_1 & 0 \\ N_1 & I_p \end{bmatrix} = \begin{bmatrix} \psi_1 I_q & Y_1 \\ 0 & -D_2 \end{bmatrix}. \tag{46}$$

Now take determinants of both sides to give

$$|U_i||D_1| = -\psi_i^q |D_2|$$

but by Corollary 1 $|D_1| = k|D_2|$, thus $|U_i| = k_i \psi_i^q$ for some $k_i \in \mathbb{R}\backslash\{0\}$ and $i = 1, 2$.

Similarly $|V_i| = l_i \phi_i^p$ for some $l_i \in \mathbb{R}\backslash\{0\}$ and $i = 1, 2$. □

Corollary 3: *Suppose that two polynomial matrices* $P_1(z_1, z_2)$, $P_2(z_1, z_2)$, *with sizes* $p_1 \times q_1$ *and* $p_2 \times q_2$ *respectively and* $p_1 - q_1 = p_2 - q_2$, *are related by a polynomial equation of the form*

$$S_1(z_1, z_2)P_2(z_1, z_2) = P_1(z_1, z_2)S_2(z_1, z_2)$$

where $S_1(z_1, z_2)$, $S_2(z_1, z_2)$ *are* $p_1 \times p_2$, $q_1 \times q_2$ *polynomial matrices and* $S_1(z_1, z_2)$, $P_1(z_1, z_2)$ *are minor (left) coprime,* $S_2(z_1, z_2)$, $P_2(z_1, z_2)$ *are minor (right) coprime.*

(i) Let $d_1^{[1]}(z_1, z_2), d_2^{[1]}(z_1, z_2), \ldots, d_q^{[1]}(z_1, z_2)$, *where* $q = \min(p_1, q_1)$, *denote the invariant polynomials of the polynomial matrix* $P_1(z_1, z_2)$ *and* $d_1^{[2]}(z_1, z_2), d_2^{[2]}(z_1, z_2), \ldots, d_p^{[2]}(z_1, z_2)$, *where* $q = \min(p_2, q_2)$ *denote the invariant polynomials of the polynomial matrix* $P_2(z_1, z_2)$ *then*

$$d_{q-i}^{[1]} = c_i d_{p-i}^{[2]} \quad \text{for} \quad i = 0, 1, \ldots, \max(p-1, q-1)$$

where $d_j^{[1]} = 1$, $d_j^{[2]} = 1$ *for* $j < 1$ *and* $c_i \in \mathbb{R}\backslash\{0\}$.

(ii) Let $e_1^{[1]}(z_1, z_2), e_2^{[1]}(z_1, z_2), \ldots, e_r^{[1]}(z_1, z_2)$, *where* $r = \min(p_1, p_2)$, *denote the invariant polynomials of the polynomial matrix* $S_1(z_1, z_2)$ *and* $e_1^{[2]}(z_1, z_2), e_2^{[2]}(z_1, z_2), \ldots, e_t^{[2]}(z_1, z_2)$, *where* $t = \min(q_1, q_2)$ *denote the invariant polynomials of the polynomial matrix* $S_2(z_1, z_2)$ *then*

$$e_{r-i}^{[1]} = c_i e_{t-i}^{[2]} \quad \text{for} \quad i = 0, 1, \ldots, \max(r-1, t-1)$$

where $e_j^{[1]} = 1$, $e_j^{[2]} = 1$ *for* $j < 1$ *and* $c_i \in \mathbb{R}\backslash\{0\}$.

Proof: The proof follows analogously to part (ii) of the theorem. Thus by the Bézout identities there exist polynomial matrices $X_i(z_1, z_2)$, $Y_i(z_1, z_2)$, $W_i(z_1, z_2)$, $Z_i(z_1, z_2)$ for $i = 1, 2$ such that

$$\begin{aligned} S_1 X_i + P_1 Y_i &= \psi_i(z_i) I_{p_1} \\ W_i P_2 + Z_i S_2 &= \phi_i(z_i) I_{q_2} \end{aligned} \quad \text{for} \quad i = 1, 2$$

where $\psi_i(z_i)$ and $\phi_i(z_i)$ are polynomials. Thus

$$\begin{bmatrix} W_i & -Z_i \\ S_1 & P_1 \end{bmatrix} \begin{bmatrix} P_2 & X_i \\ -S_2 & Y_i \end{bmatrix} = \begin{bmatrix} \phi_i(z_i) I_{q_2} & J_i \\ 0 & \psi_i(z_i) I_{p_1} \end{bmatrix} \tag{47}$$

where $J_i = W_i X_i - Z_i Y_i$. The proof follows analogously to Part (ii) of the theorem by systematic substitution of $X_i(z_1, z_2)$, $Y_i(z_1, z_2)$, $W_i(z_1, z_2)$, $Z_i(z_1, z_2)$ by the zero matrix and E, as defined in the theorem, for $i = 1, 2$. □

The above theorem can also be interpreted in terms of the Smith forms of the numerator and denominator matrices. By definition the Smith form is a polynomial matrix with the invariant polynomials on the principle diagonal and zeros everywhere else.

Thus in (29) if $p \geq q$ and the $q \times q$ denominator $D_1(z_1, z_2)$ has invariant polynomials $d_1^{[1]}(z_1, z_2), d_2^{[1]}(z_1, z_2), \ldots, d_q^{[1]}(z_1, z_2)$ the Smith form is

$$\begin{pmatrix} d_1^{[1]}(z_1, z_2) & 0 & \cdots & 0 \\ 0 & d_2^{[1]}(z_1, z_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_q^{[1]}(z_1, z_2) \end{pmatrix} \tag{48}$$

and using the theorem the Smith form of the $p \times p$ denominator matrix $D_2(z_1, z_2)$ is

$$\begin{array}{c} \\ 1 \\ \\ p-q \\ \\ \\ \\ p \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & & & p-q & & & & p \end{array} \\ \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & c_1 d_1^{[1]}(z_1, z_2) & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & c_2 d_2^{[1]}(z_1, z_2) & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & c_q d_q^{[1]}(z_1, z_2) \end{pmatrix} \end{array} \tag{49}$$

where the first $p-q$ are unit invariant polynomials and $c_i \in \mathbb{R}\backslash\{0\}$ for $i = 1, 2, \ldots, q$. Similarly if the invariant polynomials of the $p \times q$ numerator matrix $N_1(z_1, z_2)$ are $n_1^{[1]}(z_1, z_2), n_2^{[1]}(z_1, z_2), \ldots, n_q^{[1]}(z_1, z_2)$ the Smith form of $N_2(z_1, z_2)$ is

$$\begin{pmatrix} k_1 n_1^{[1]}(z_1, z_2) & 0 & \cdots & 0 \\ 0 & k_2 n_2^{[1]}(z_1, z_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & k_q n_q^{[1]}(z_1, z_2) \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

where $k_i \in \mathbb{R}\backslash\{0\}$ for $i = 1, 2, \ldots, q$. To demonstrate this equivalence of Smith form consider the following example.

Example 2.3: Consider the rational matrix $G(z_1, z_2)$ given in Example 2.1

$$G(z_1, z_2) = \begin{pmatrix} z_1 & z_2 + 1 \\ \dfrac{z_1 + z_2}{z_2} & \dfrac{z_2}{z_1} \\ \dfrac{z_1}{z_2} & 1 \end{pmatrix}$$

with minor coprime MFDs given by

$$G(z_1,z_2) = N_1(z_1,z_2)D_1^{-1}(z_1,z_2) = \begin{pmatrix} z_1z_2 & (z_2+1)z_1 \\ z_1+z_2 & z_2 \\ z_1 & z_1 \end{pmatrix} \begin{pmatrix} z_2 & 0 \\ 0 & z_1 \end{pmatrix}^{-1} \tag{50}$$

$$= D_2^{-1}(z_1,z_2)N_2(z_1,z_2) = \begin{pmatrix} 1 & 0 & -z_2 \\ 0 & 0 & z_2 \\ 0 & z_1 & -z_1 \end{pmatrix}^{-1} \begin{pmatrix} 0 & 1 \\ z_1 & z_2 \\ z_1 & z_2-z_1 \end{pmatrix} \tag{51}$$

The Smith forms of $D_1(z_1,z_2)$, $D_2(z_1,z_2)$, $N_1(z_1,z_2)$, $N_2(z_1,z_2)$ (viz, $S_{D_1}(z_1,z_2)$, $S_{D_2}(z_1,z_2)$, $S_{N_1}(z_1,z_2)$, $S_{N_2}(z_1,z_2)$, respectively) are given by

$$S_{D_1}(z_1,z_2) = \begin{pmatrix} 1 & 0 \\ 0 & z_1z_2 \end{pmatrix} \qquad S_{D_2}(z_1,z_2) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -z_1z_2 \end{pmatrix}$$

$$S_{N_1}(z_1,z_2) = \begin{pmatrix} 1 & 0 \\ 0 & z_1 \\ 0 & 0 \end{pmatrix} \qquad S_{N_2}(z_1,z_2) = \begin{pmatrix} 1 & 0 \\ 0 & z_1 \\ 0 & 0 \end{pmatrix}.$$

Hence the theorem is demonstrated. □

Note: Theorem 2.5 only considers the case when both the left and right matrix fraction descriptions are minor coprime, but in light of the previous section if one coprime MFD is minor coprime then all coprime MFDs are minor coprime. Also the theorem does not explicitly state the result if the MFDs are zero coprime, however, zero coprimeness is a stronger statement of coprimeness than minor coprimeness thus it is contained in the definition of minor coprimeness. For these reasons the MFDs (29) of Theorem 2.5 could be described as any coprime MFDs and the theorem is still valid.

The MFD structural result, presented as Theorem 2.5, also holds for coprime 1-D matrix fraction descriptions, because all definitions of coprimeness are equivalent when $n = 1$. The result is restated here to demonstrate that the 1-D result is a specific case of the more general 2-D theory.

Theorem 2.6: (*1-D MFD Structure Theorem*)
Let $G_{p\times q}(z_1)$ be a rational matrix and have a MFD defined by

$$\begin{aligned} G(z_1) &= N_1(z_1)D_1^{-1}(z_1) \\ &= D_2^{-1}(z_1)N_2(z_1) \end{aligned} \tag{52}$$

where $N_1(z_1)$, $D_1(z_1)$ are 1-D (right) coprime and $N_2(z_1)$, $D_2(z_1)$ are 1-D (left) coprime.

(i) *Let* $d_1^{[1]}(z_1), d_2^{[1]}(z_1), \ldots, d_q^{[1]}(z_1)$ *denote the invariant polynomials of the* $q \times q$ *polynomial matrix* $D_1(z_1)$ *and* $d_1^{[2]}(z_1), d_2^{[2]}(z_1), \ldots, d_p^{[2]}(z_1)$ *denote the invariant polynomials of* $p \times p$ *polynomial matrix* $D_2(z_1)$ *then*

$$d_{q-i}^{[1]} = c_i d_{p-i}^{[2]} \quad \text{for} \quad i = 0, 1, \ldots, \max(p-1, q-1)$$

where $d_j^{[1]} = 1$, $d_j^{[2]} = 1$ *for* $j < 1$ *and* $c_i \in \mathbb{R} \backslash \{0\}$.

(ii) *The* $p \times q$ *polynomial matrices* $N_1(z_1)$ *and* $N_2(z_1)$ *have identical invariant polynomials, modulo a non-zero constant factor.* □

2.3 n-D Matrix Fraction Descriptions

It has been seen in the previous section how results for matrix fraction descriptions in one indeterminate are transformed into complicated and important results for 2-D MFDs. For example, the single definition of 1-D coprimeness renders Theorem 2.2 as an obvious statement, however, its importance within a 2-D framework is self evident: by eliminating the possibility of mixed coprimeness statements the coprimeness of one MFD determines the coprimeness of all other MFDs.

The situation of matrix fraction descriptions in an arbitrary number of indeterminates (generally assumed greater than two, unless stated) is more complicated. The 2-D results only partially carry over to the general case. These results are discussed below.

The definition of a general MFD is equivalent to Definition 2.1 with n indeterminates, i.e (z_1, z_2) is replaced by (z) where $(z) = (z_1, z_2, \ldots, z_n)$. However, for polynomial matrices in n indeterminates there are three definitions of coprimeness, thus defining three types of coprime MFD. Explicitly:

Definition 2.3: (*Coprime MFD*)
The factorisation of the rational matrix $G_{p\times q}(z)$ given by

$$G(z) = D^{-1}(z)N(z),$$

where $D_{p\times p}(z)$ and $N_{p\times q}(z)$ are polynomial matrices, is described as a:

(i) *factor (left) coprime MFD* if $D(z)$ and $N(z)$ are factor (left) coprime matrices;
(ii) *minor (left) coprime MFD* if $D(z)$ and $N(z)$ are minor (left) coprime matrices;
(iii) *zero (left) coprime MFD* if $D(z)$ and $N(z)$ are zero (left) coprime matrices.

Right factorisation may similarly be defined by transposition of $G(z)$. □

Note: The term *coprime MFD* will be used in its more general sense to mean a matrix fraction description with numerator and denominator that are factor, minor or zero coprime. In certain circumstances this term will be used with the qualification of the number of indeterminates, i.e. n-D coprime MFD.

Earlier it was proved (Theorem 2.1) that every 2-D rational matrix possesses a 2-D coprime MFD via a constructive algorithm. The generalisation of this theorem is trivial since the algorithm does not depend on the number of indeterminates, therefore the same algorithm may be employed to calculate n-D coprime MFDs. Unfortunately the practical implication of this algorithm is not so easily generalised. The formation of a matrix fraction description, that is not necessarily coprime (Steps 1 to 3), is straight

forward, however, the calculation of a greatest common divisor to form a coprime MFD is difficult; it is not even known whether an algorithm exists to systematically calculate this form.

2.3.1 Coprime Invariance of n-D MFDs

For matrix fraction descriptions in two indeterminates the type of coprimeness is a property of the rational matrix (Theorem 2.2) and does not depend on the MFD. The results for the general n-D case are split into two parts. Firstly it is shown that all left factorisations of a rational matrix possess the same type of coprimeness and consequently, by transposition, all right factorisations of a rational matrix possess the same type of coprimeness. Secondly it is shown that the type of coprimeness possessed by all left factorisations does not necessarily determine the coprimeness type of all right factorisations and vice versa.

Theorem 2.7: *Any two left coprime MFDs of the same rational matrix $G_{p\times q}(z)$ in n indeterminates have the same type of coprimeness.*

Proof: This theorem will be proved by showing that any two left coprime MFDs can not be of different coprimeness. Thus it is required to prove

(i) if a *zero (left) coprime* MFD exists, then does not exist either a *minor (left) coprime* or a *factor (left) coprime* MFD,

(ii) if a *minor (left) coprime* MFD exists, then a *factor (left) coprime* MFD does not exist.

(i) It is required to prove that *if there exists a zero (left) coprime MFD then there does not exist a minor (left) coprime MFD.* Suppose that

$$\begin{aligned} G_{p\times q}(z) &= D_1^{-1}(z)N_1(z) \\ &= D_2^{-1}(z)N_2(z) \end{aligned} \tag{53}$$

where $D_1(z)$, $N_1(z)$ are either minor (left) coprime or factor (left) coprime and $D_2(z)$, $N_2(z)$ are zero (left) coprime. Now from the Bézout identities (Theorem 1.4) there exist polynomial matrices $X(z)$, $Y(z)$ such that

$$D_2X + N_2Y = I_p. \tag{54}$$

Now substitute $N_2 = D_2D_1^{-1}N_1$ from (53) into (54) to obtain

$$\begin{aligned} D_2X + D_2D_1^{-1}N_1Y &= I \\ D_1X + N_1Y &= D_1D_2^{-1}. \end{aligned} \tag{55}$$

The left-hand-side of (55) is polynomial, therefore the right-hand-side of (55) is also polynomial, i.e. $D_1 D_2^{-1} = E(z)$, a polynomial matrix. Thus

$$N_1 = EN_2$$

But D_1, N_1 are either minor (left) coprime or factor (left) coprime, therefore E is unimodular. But (55) gives a zero coprime Bézout identity

$$D_1 X E^{-1} + N_1 Y E^{-1} = I_p$$

hence D_1, N_1 are zero (left) coprime.

Thus (i) is proved.

(ii) It is required to prove that *if there exists a minor (left) coprime MFD then there does not exist a factor (left) coprime MFD.* Suppose that

$$\begin{aligned} G_{p\times q}(z) &= D_1^{-1}(z) N_1(z) \\ &= D_2^{-1}(z) N_2(z) \end{aligned} \tag{56}$$

where $D_1(z)$, $N_1(z)$ are factor (left) coprime and $D_2(z)$, $N_2(z)$ are minor (left) coprime. Now from the Bézout identities (Theorem 1.4) there exist polynomial matrices $X_i(z)$, $Y_i(z)$ for $i = 1, 2, \ldots, n$ such that

$$D_2 X_i + N_2 Y_i = \psi_i(z_i^c) I_p, \quad i = 1, 2, \ldots, n \tag{57}$$

where $\psi_i(z_i^c)$ is a polynomial in the $n-1$ indeterminates $z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n$. Now substitute $N_2 = D_2 D_1^{-1} N_1$ from (56) into (57) to obtain for $i = 1, 2, \ldots, n$

$$D_2 X_i + D_2 D_1^{-1} N_1 Y_i = \psi_i I \tag{58}$$

$$\Rightarrow \quad D_1 X_i + N_1 Y_i = \psi_i D_1 D_2^{-1} \tag{59}$$

Now the elements of $D_1 D_2^{-1}$ are not rational, as the elements of the matrices of the left-hand-side of (59) are not rational and the ψ_i are a factor coprime set for $i = 1, 2, \ldots, n$, i.e. if one element of $D_1 D_2^{-1}$ is rational, $n(z_1, z_2)/d(z_1, z_2)$ say, then the $d | \psi_i$ for $i = 1, 2, \ldots, n$, thus by Theorem 1.1 $d(z_1, z_2)$ is a constant.

Thus there are two possibilities for the form of $D_1 D_2^{-1}$.

(a) $D_1 D_2^{-1} = E_1(z)$, a non-unimodular polynomial matrix or

(b) $D_1 D_2^{-1} = U(z)$, a unimodular matrix.

If (a) holds then $D_1 = E_1 D_2$ and from (56) $N_1 = E_1 D_2 D_2^{-1} N_2 = E_1 N_2$ hence D_1, N_1 are not factor (left) coprime as they both have a polynomial factor E_1.

If (b) holds then $D_1 D_2^{-1} = U(z)$ and D_1, N_1 possess a minor coprime Bézout identity

$$D_1 X_i U^{-1} + N_1 Y_i U^{-1} = \psi_i I$$

and therefore can not be factor (left) coprime.

Thus (ii) is proved. □

Corollary 1: *If one left MFD of the rational matrix $G_{p\times q}(z)$ is*

(i) zero (left) coprime then all coprime left MFDs are zero (left) coprime,

(ii) minor (left) coprime then all coprime left MFDs are minor (left) coprime,

(iii) factor (left) coprime then all coprime left MFDs are factor (left) coprime.

Proof: This is a restatement of the above theorem. □

The same statements can be made for right factorisations.

Corollary 2: *Suppose that $D_1^{-1}(z)N_1(z)$, $D_2^{-1}(z)N_2(z)$ are minor (left) coprime MFDs of $G_{p\times q}(z)$. Then there exists a unimodular matrix $U(z)$ such that*

$$D_1(z) = U(z)D_2(z), \qquad N_1(z) = U(z)N_2(z). \tag{60}$$

Proof: From part (i) (a) and (b) of the theorem it has been deduced that $D_1D_2^{-1}$ is unimodular. Therefore there exists a unimodular matrix $U(z) = D_1D_2^{-1}$ such that $D_1(z) = U(z)D_2(z)$ and $N_1(z) = U(z)D_2(z)$. □

This corollary also holds for zero coprime factorisations due to the inclusion of zero coprimeness in minor coprimeness (see Theorem 1.3). However, it is not true that all factor (left) coprime MFDs are linked by a unimodular matrix as in Corollary 2 of Theorem 2.7. To prove this consider the following theorem by Lin [45].

Theorem 2.8: *Suppose that $G_{p\times q}(z)$ has the following MFDs*

$$\begin{aligned} G(z) &= N_1(z)D_1^{-1}(z) \\ &= D_2^{-1}(z)N_2(z) \end{aligned} \tag{61}$$

where $D_1(z)$, $N_1(z)$ are factor (right) coprime and $D_2(z)$, $N_2(z)$ are minor (left) coprime. Then there exists a right MFD of $G(z) = \bar{N}(z)\bar{D}^{-1}(z)$ such that $\bar{D} \neq D_1U$ and $\bar{N} \neq N_1U$ for any polynomial matrix, not just unimodular, $U(z)$.

Proof: The proof is detailed in [45]. □

This theorem provides the first indication that the type of coprime MFD possessed by a n-D rational matrix is not unique. A further indication is provided by the following example due to Lévy [34].

Example 2.4: Let

$$G(z_1, z_2, z_3) = \begin{pmatrix} \frac{z_2}{z_1} & \frac{z_3}{z_1} \end{pmatrix}$$

then

$$G(z_1, z_2, z_3) = (z_1)^{-1} \begin{pmatrix} z_2 & z_3 \end{pmatrix} = \begin{pmatrix} z_2 & z_3 \end{pmatrix} \begin{pmatrix} z_1 & 0 \\ 0 & z_1 \end{pmatrix}^{-1}.$$

The high-order minors of the left MFD are

$$z_1, \quad z_2, \quad z_3$$

and the high-order minors of the right MFD are

$$-z_1 z_3, \quad z_1 z_2, \quad z_1^2.$$

Thus the left MFD is minor (left) coprime. However, it is shown by Lévy [34] that all polynomial matrix right divisors are unimodular; therefore the right MFD is factor (right) coprime.

By Theorem 2.7 all left factorisations of $G(z_1, z_2)$ are necessarily minor (left) coprime and all right factorisations are necessarily factor (right) coprime. Therefore a minor (right) coprime MFD does not exist. □

The case of a rational matrix possessing a zero (left) coprime MFD and a factor (right) coprime MFD is theoretically feasible since zero coprimeness is contained in minor coprimeness but an example has proved elusive to find. Further theoretical evidence for the existence of such a combination of coprimeness types is provided later by the introduction of the notion of generating polynomials. These are discussed after the following theorem relating the two remaining types of coprimeness for opposite sided MFDs of a rational matrix.

Theorem 2.9: *If the rational matrix $G(z)$ has one minor coprime factorisation then there does not exist a zero coprime factorisation and conversely if $G(z)$ has one zero coprime factorisation then there does not exist a minor coprime factorisation.*

Proof: The proof of this theorem follows in a manner analogous to that of Theorem 2.2 and the ensuing corollaries. Therefore only a sketch of the proof is given here.

Firstly suppose that $G(z)$ possesses a zero (left) coprime MFD defined by

$$G(z) = D_1^{-1}(z) N_1(z) \tag{62}$$

and suppose that there exists a minor (right) coprime MFD defined by

$$G(z) = N_2(z) D_2^{-1}(z) \tag{63}$$

The Bézout identities now give

$$D_1X + N_1Y = I_p \tag{64}$$

$$X_iD_2 + Y_iN_2 = \psi_i(z_i^c)I_q \quad \text{for} \quad i = 1,2,\ldots,n \tag{65}$$

where $X(z)$, $Y(z)$ and $X_i(z)$, $Y_i(z)$ for $i = 1,2,\ldots,n$ are polynomial matrices and $\psi_i(z_i^c)$ are polynomials in the $n-1$ indeterminates $z_1,\ldots,z_{i-1},z_{i+1},\ldots,z_n$.

From these Bézout identities (64), (65) and equations (62), (63) a matrix equation corresponding to (24) of Theorem 2.2 may be set up and the discussion following this equation establishes that if one MFD of $G(z)$ is zero coprime then there does not exist a minor coprime MFD of $G(z)$. The converse is proved by Corollary 2 of Theorem 2.2. □

In light of these results it is not possible to prove the Completion Theorem for n-D polynomial matrices using matrix theoretic arguments similar to Theorems 2.3 and 2.4. This is due to the coprimeness type of, say, a right MFD given the coprimeness of a left MFD, i.e. if a left MFD of a rational n-D polynomial matrix $G(z)$ is zero (left) coprime then a right MFD may be either zero or factor (right) coprime. However, it is possible to provide sufficient conditions for zero and minor coprimeness similar to those derived for 2-D polynomial matrices (Theorems 2.3 and 2.4).

Lemma 2.2: *Suppose that two polynomial matrices $D_{p\times p}(z)$ and $N_{p\times q}(z)$ can be incorporated as the first p rows of a unimodular matrix $U(z)$. Then $D(z)$, $N(z)$ are zero (left) coprime.*

Proof: The proof follows in an analogous way to the second part of Theorem 2.3. □

Considering the sufficiency proof of Theorem 2.4, i.e. all polynomial matrix factors are unimodular, it might be expected that a sufficiency characterisation for factor coprimeness can be derived. However, on closer inspection the result is seen to hold for minor coprimeness by using Laplace expansion. Therefore the following result is obtained.

Lemma 2.3: *Let $D_{p\times p}(z)$ and $N_{p\times q}(z)$ be polynomial matrices and suppose that there exist polynomial matrices $X_i(z)$, $Y_i(z)$ for $i = 1,2,\ldots,n$ such that*

$$U_i(z) = \begin{pmatrix} D(z) & N(z) \\ X_i(z) & Y_i(z) \end{pmatrix} \qquad \text{for} \qquad i = 1,2,\ldots,n$$

and $|U_i(z)| = \alpha_i(z_i^c)$ for $i = 1,2,\ldots,n$, where $\alpha_i(z_i^c)$ are polynomials in the $n-1$ indeterminates $z_1,\ldots,z_{i-1},z_{i+1},\ldots,z_n$. Then $D(z)$, $N(z)$ form a minor (left) coprime pair.

Proof: Suppose that there exist polynomial matrices $X_i(z)$, $Y_i(z)$ for $i = 1, 2, \ldots, n$ such that the determinants of $U_i(z)$ possess the properties defined by the theorem hypothesis. Using Laplace expansion on the first p rows of $|U_i|$ for $i = 1, 2, \ldots, n$ the following equations are obtained.

$$\sum_{j=1}^{m} u_j(z)\bar{u}_j(z) = \alpha_i(z_i^c) \qquad \text{for} \qquad i = 1, 2, \ldots, n$$

where $u_i(z)$, $i = 1, \ldots, m$, are the high-order minors of $(D\ N)$ and $\bar{u}_j^{(i)}(z)$ are the minors obtained from $U_i(z)$ for $i = 1, 2, \ldots, n$ by deleting the rows and columns used in forming the minor $u_i(z)$. Let $g(z)$ be the greatest common divisor of the high-order minors of $D(z)$, $N(z)$. Then

$$g(z)|\alpha_i(z_i^c) \qquad \text{for} \qquad i = 1, 2, \ldots, n$$

Since the $\alpha_i(z_i^c)$ form a factor coprime set of polynomials, Theorem 1.1 proves that $g(z)$ is a constant, i.e. the high-order minors of $D(z)$, $N(z)$ are factor coprime. Therefore $D(z)$, $N(z)$ form a minor (left) coprime pair of matrices. □

The concept of generating polynomials is now introduced. These were first defined by Lin [45] to investigate the structure of n-D matrix fraction descriptions.

Definition 2.4: (*Generating Polynomials*)
Let $A_{p\times q}(z)$ be a full rank polynomial matrix in the n indeterminates $z_1, z_2, \ldots, z_n$. Suppose that $r = \min(p, q)$; then denote the determinants of the $r \times r$ minors by

$$a_1(z), a_2(z), \ldots, a_\beta(z) \qquad \text{where} \quad \beta = \binom{\max(p,q)}{\min(p,q)} \tag{66}$$

i.e. the number of combinations of $\max(p, q)$ taken $\min(p, q)$ at a time. Denote the greatest common divisor of these polynomials by $g(z)$, then

$$a_i(z) = g(z)b_i(z), \qquad i = 1, 2, \ldots, \beta. \tag{67}$$

The $b_1(z), \ldots, b_\beta(z)$ are called the *generating polynomials* of $A(z)$. □

Suppose that a rational matrix $G_{p\times q}(z)$ has MFDs, not necessarily coprime, defined by

$$\begin{aligned} G(z) &= N_1(z)D_1^{-1}(z) \\ &= D_2^{-1}(z)N_2(z) \end{aligned} \tag{68}$$

Then the generating polynomials of the right MFD are defined to be the generating polynomials of the matrix $A_1(z)$

$$A_1(z) = \begin{bmatrix} D_1(z) \\ N_1(z) \end{bmatrix} \tag{69}$$

and the generating polynomials of the left MFD are defined to be the generating polynomials of the matrix $A_2(z)$

$$A_2(z) = [\, D_2(z) \quad N_2(z) \,] \tag{70}$$

If a matrix fraction description is minor or zero coprime, by definition the high-order minors form a factor coprime set, and so the generating polynomials may be taken to be these polynomials.

The following two theorems relate the generating polynomials of two, not necessarily coprime, MFD of the same rational matrix.

Theorem 2.10: *Let $G_{p\times q}(z)$ be a rational matrix with two left MFDs defined by*

$$\begin{aligned} G(z) &= D_1^{-1}(z)N_1(z) \\ &= D_2^{-1}(z)N_2(z). \end{aligned} \tag{71}$$

Denote the generating polynomials of $D_1^{-1}(z)N_1(z)$ by $b_1^{[1]}, b_2^{[1]}, \ldots, b_\beta^{[1]}$ and the generating polynomials of $D_2^{-1}(z)N_2(z)$ by $b_1^{[2]}, b_2^{[2]}, \ldots, b_\beta^{[2]}$, where $\beta = \binom{\max(p,q)}{\min(p,q)}$. Then

$$b_i^{[1]} = k b_i^{[2]}, \qquad i = 1, 2, \ldots, \beta, \quad \text{for some} \quad k \in \mathbb{R}. \tag{72}$$

Proof: The proof is detailed in [45]. □

Thus left MFDs have essentially unique generating polynomials. The same theorem may be applied to right MFDs with the same result being deduced. But can the generating polynomials of a left MFD and a right MFD, of the same rational matrix, be linked in a similar manner? The following theorem gives the answer.

Theorem 2.11: *Let $G_{p\times q}(z)$ be an n-D rational matrix with a general left and right MFD defined by*

$$\begin{aligned} G(z) &= N_1(z)D_1^{-1}(z) \\ &= D_2^{-1}(z)N_2(z). \end{aligned} \tag{73}$$

Denote the generating polynomials of $N_1(z)D_1^{-1}(z)$ by $b_1^{[1]}, b_2^{[1]}, \ldots, b_\beta^{[1]}$ and the generating polynomials of $D_2^{-1}(z)N_2(z)$ by $b_1^{[2]}, b_2^{[2]}, \ldots, b_\beta^{[2]}$, where $\beta = \binom{\max(p,q)}{\min(p,q)}$. Then

$$b_i^{[1]} = k \bar{b}_i^{[2]}, \qquad i = 1, 2, \ldots, \beta \tag{74}$$

where $\bar{b}_i^{[2]}$ are obtained by reordering $b_1^{[2]}, b_2^{[2]}, \ldots, b_\beta^{[2]}$, with $b_1^{[2]} = \bar{b}_1^{[2]}$, and $k \in \mathbb{R}$.

Proof: The proof is detailed in [45]. □

To illustrate the equivalence of the generating polynomials of matrix fraction descriptions consider the following example.

Example 2.5: From Example 2.1 the rational matrix $G(z_1, z_2)$

$$\begin{pmatrix} z_1 & z_2+1 \\ \frac{z_1+z_2}{z_2} & \frac{z_2}{z_1} \\ \frac{z_1}{z_2} & 1 \end{pmatrix}$$

admits three matrix fraction descriptions, one minor (right) coprime, one minor (left) coprime and one non-coprime respectively given by (75), (76) and (77).

$$N_1(z_1,z_2)D_1^{-1}(z_1,z_2) = \begin{pmatrix} z_1z_2 & (z_2+1)z_1 \\ z_1+z_2 & z_2 \\ z_1 & z_1 \end{pmatrix} \begin{pmatrix} z_2 & 0 \\ 0 & z_1 \end{pmatrix}^{-1} \tag{75}$$

$$D_2^{-1}(z_1,z_2)N_2(z_1,z_2) = \begin{pmatrix} 1 & 0 & -z_2 \\ 0 & 0 & z_2 \\ 0 & z_1 & -z_1 \end{pmatrix}^{-1} \begin{pmatrix} 0 & 1 \\ z_1 & z_2 \\ z_1 & z_2-z_1 \end{pmatrix} \tag{76}$$

$$\bar{D}_2^{-1}(z_1,z_2)\bar{N}_2(z_1,z_2) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & z_1z_2 & 0 \\ 0 & 0 & z_2 \end{pmatrix}^{-1} \begin{pmatrix} z_1 & z_2+1 \\ z_1(z_1+z_2) & z_2^2 \\ z_1 & z_2 \end{pmatrix} \tag{77}$$

The high-order minors of the augmented matrices

$$M_1(z_1,z_2) = [D_1^T(z_1,z_2)\ N_1^T(z_1,z_2)]^T$$
$$M_2(z_1,z_2) = [D_2(z_1,z_2)\ N_2(z_1,z_2)]$$
$$\bar{M}_2(z_1,z_2) = [\bar{D}_2(z_1,z_2)\ \bar{N}_2(z_1,z_2)]$$

are given by:

For the minor (right) coprime MFD

$$M_1\ {}^{1,2}_{1,2} = z_1z_2 \qquad M_1\ {}^{1,3}_{1,2} = z_1z_2(z_2+1)$$
$$M_1\ {}^{1,4}_{1,2} = z_2^2 \qquad M_1\ {}^{1,5}_{1,2} = z_1z_2$$
$$M_1\ {}^{2,3}_{1,2} = -z_1^2z_2 \qquad M_1\ {}^{2,4}_{1,2} = -z_1(z_1+z_2)$$
$$M_1\ {}^{2,5}_{1,2} = -z_1^2 \qquad M_1\ {}^{3,4}_{1,2} = -z_1(z_1z_2+z_1+z_2)$$
$$M_1\ {}^{3,5}_{1,2} = -z_1^2 \qquad M_1\ {}^{4,5}_{1,2} = z_1^2$$

These polynomials are factor coprime and may therefore be taken to be the generating polynomials.

For the minor (left) coprime MFD

$$M_2\ {}^{1,2,3}_{1,2,3} = z_1 z_2 \qquad M_2\ {}^{1,2,3}_{1,2,4} = z_1^2$$
$$M_2\ {}^{1,2,3}_{1,2,5} = z_1 z_2 \qquad M_2\ {}^{1,2,3}_{1,3,4} = -z_1(z_1 + z_2)$$
$$M_2\ {}^{1,2,3}_{1,3,5} = -z_2^2 \qquad M_2\ {}^{1,2,3}_{1,4,5} = z_1^2$$
$$M_2\ {}^{1,2,3}_{2,3,4} = z_1^2 z_2 \qquad M_2\ {}^{1,2,3}_{2,3,5} = z_1 z_2(z_2 + 1)$$
$$M_2\ {}^{1,2,3}_{2,4,5} = z_1^2 \qquad M_2\ {}^{1,2,3}_{3,4,5} = -z_1(z_1 z_2 + z_1 + z_2)$$

Again these polynomials are factor coprime and may therefore be taken to be the generating polynomials.

For the left non-coprime MFD

$$\bar{M}_2\ {}^{1,2,3}_{1,2,3} = z_1 z_2^2 \qquad \bar{M}_2\ {}^{1,2,3}_{1,2,4} = z_1^2 z_2$$
$$\bar{M}_2\ {}^{1,2,3}_{1,2,5} = z_1 z_2^2 \qquad \bar{M}_2\ {}^{1,2,3}_{1,3,4} = -z_1 z_2(z_1 + z_2)$$
$$\bar{M}_2\ {}^{1,2,3}_{1,3,5} = -z_2^3 \qquad \bar{M}_2\ {}^{1,2,3}_{1,4,5} = z_1^2 z_2$$
$$\bar{M}_2\ {}^{1,2,3}_{2,3,4} = z_1^2 z_2^2 \qquad \bar{M}_2\ {}^{1,2,3}_{2,3,5} = z_1 z_2^2(z_2 + 1)$$
$$\bar{M}_2\ {}^{1,2,3}_{2,4,5} = z_1^2 z_2 \qquad \bar{M}_2\ {}^{1,2,3}_{3,4,5} = -z_1 z_2(z_1 z_2 + z_1 + z_2)$$

These polynomial have greatest common polynomial divisor z_2, therefore the generating polynomials may be taken to be the high-order minors divided by z_2 to give identical generating polynomials to $M_2(z_1, z_2)$. Thus Theorem 2.10 is demonstrated with all k's identically equal to one.

To illustrate Theorem 2.11 consider the generating polynomials of $M_1(z_1, z_2)$ and $M_2(z_1, z_2)$. Thus

$$M_2\ {}^{1,2,3}_{1,2,3} = M_1\ {}^{1,2}_{1,2} \qquad M_2\ {}^{1,2,3}_{1,2,4} = M_1\ {}^{2,5}_{1,2}$$
$$M_2\ {}^{1,2,3}_{1,2,5} = M_1\ {}^{1,5}_{1,2} \qquad M_2\ {}^{1,2,3}_{1,3,4} = M_1\ {}^{2,4}_{1,2}$$
$$M_2\ {}^{1,2,3}_{1,3,5} = -M_1\ {}^{1,4}_{1,2} \qquad M_2\ {}^{1,2,3}_{1,4,5} = -M_1\ {}^{3,5}_{1,2}$$
$$M_2\ {}^{1,2,3}_{2,3,4} = -M_1\ {}^{2,3}_{1,2} \qquad M_2\ {}^{1,2,3}_{2,3,5} = M_1\ {}^{1,3}_{1,2}$$
$$M_2\ {}^{1,2,3}_{2,4,5} = M_1\ {}^{4,5}_{1,2} \qquad M_2\ {}^{1,2,3}_{3,4,5} = M_1\ {}^{3,4}_{1,2}$$

□

Applying Theorem 2.11 to a 2-D rational matrix Theorem 2.2 may be derived in the following manner.

Suppose that the 2-D rational matrix $G_{p\times q}(z_1, z_2)$ has a left coprime MFD defined by

$$G(z_1, z_2) = D(z_1, z_2)^{-1} N(z_1, z_2) \tag{78}$$

Then $D(z_1, z_2)$, $N(z_1, z_2)$ are either minor (left) coprime or zero (left) coprime, hence the generating polynomials may be taken as the high-order minors of $A(z_1, z_2) = [D(z_1, z_2)\ N(z_1, z_2)]$. By Theorems 2.10 and 2.11 these generating polynomials define the generating polynomials of all other MFDs. Thus all coprime MFDs possess the same zero set and hence possess the same type of coprimeness.

The proof of Theorem 2.11 requires polynomial matrices with elements in $\mathbb{R}[z_1, z_2, \ldots, z_n]$ to be considered as elements of the more generalised ring $\mathbb{R}(z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n)[z_i]$ and the argument repeated for $i = 1, 2, \ldots, n$. On the other hand the proof of Theorem 2.2 uses only polynomial arguments in the ring $\mathbb{R}[z_1, z_2, \ldots, z_n]$ and the Bézout identities (Theorem 1.4). Thus the proof of Theorem 2.2 is seen to be a more direct and simple argument.

In the same way Theorem 2.9 can also be derived for n-D MFDs using Theorem 2.11, but once again the proof of Theorem 2.9 is more direct and simple. Also Theorems 2.10 and 2.11 do not reveal the full invariance of same sided MFDs proved in Theorem 2.7 for the following reasons.

Consider a factor (left) coprime MFD then the generating polynomials are defined to be a set of coprime polynomials derived from the high-order minors. In general it is not possible to recreate the high-order minors with only the knowledge of the generating polynomials, thus it is not possible to prove that all other left coprime MFDs are also factor coprime as demonstrated by Theorem 2.7.

As mentioned earlier an example for a zero (left) coprime MFD and a factor (right) coprime MFD of the same n-D rational matrix has proved elusive. A theoretical justification for the existence of such a combination of coprime MFDs can be derived using the notion of generating polynomials. Suppose that there exists a factor (right) coprime MFD with generating polynomials $a_1(z), a_2(z), \ldots, a_\beta(z)$. It has already been seen that if these generating polynomials are factor coprime, but not zero coprime, a minor (left) coprime MFD can be formed as in Example 2.5. Thus if these generating polynomials are additionally zero coprime a zero (left) coprime MFD is theoretically feasible.

2.3.2 Coprime n-D MFD Examples

Below are detailed four examples of coprime matrix fraction descriptions to demonstrate the results obtained in the previous section. The following table summarises these results by division into three categories

		Right MFD		
		Zero	Minor	Factor
Left MFD	Zero	✓	✘	?*
	Minor	✘	✓	✓
	Factor	?*	✓	✓

✓ Factorisations of these types are possible for a specific rational matrix and an example has been presented.

✘ Factorisations of these types are theoretically impossible: thus no example exists.

*? Factorisations of this type are theoretically possible but no example is given.

Figure 2.1 Summary of Examples

Example 2.6: (Factor Right/Minor Left) From Example 2.4 the rational matrix

$$G(z_1, z_2, z_3) = \begin{pmatrix} \frac{z_2}{z_1} & \frac{z_3}{z_1} \end{pmatrix}$$

has a factor (right) coprime MFD and a minor (left) coprime MFD given by

$$\begin{aligned} G(z_1, z_2, z_3) &= \begin{pmatrix} z_2 & z_3 \end{pmatrix} \begin{pmatrix} z_1 & 0 \\ 0 & z_1 \end{pmatrix}^{-1} \\ &= (z_1)^{-1} \begin{pmatrix} z_2 & z_3 \end{pmatrix}. \end{aligned}$$

□

Example 2.7: (Factor Right/Factor Left) This example and subsequent reasoning is due to Youla and Gnavi [40]. Let

$$\begin{aligned} G &= \begin{pmatrix} \frac{z_2 z_3 - z_1^3}{z_1} & \frac{z_2^2 - z_1 z_3}{z_1} \\ \frac{z_3^2 - z_1^2 z_2}{z_1} & \frac{z_2 z_3 - z_1^3}{z_1} \end{pmatrix} \\ &= \begin{pmatrix} z_1 & 0 \\ 0 & z_1 \end{pmatrix}^{-1} \begin{pmatrix} z_2 z_3 - z_1^3 & z_2^2 - z_1 z_3 \\ z_3^2 - z_1^2 z_2 & z_2 z_3 - z_1^3 \end{pmatrix} \\ &= \begin{pmatrix} z_2 z_3 - z_1^3 & z_2^2 - z_1 z_3 \\ z_3^2 - z_1^2 z_2 & z_2 z_3 - z_1^3 \end{pmatrix} \begin{pmatrix} z_1 & 0 \\ 0 & z_1 \end{pmatrix}^{-1} \end{aligned}$$

The high-order minors of the two compound matrices are

* It has been proved that these types of MFD can not simultaneously exist.

$$M_1 = \begin{pmatrix} z_1 & 0 & f_2 & f_1 \\ 0 & z_1 & f_3 & f_2 \end{pmatrix} \quad \Rightarrow \quad z_1^2, z_1 f_3, z_1 f_2, -z_1 f_2, -z_1 f_1, f_2^2 - f_1 f_3$$

$$M_2 = \begin{pmatrix} z_1 & 0 \\ 0 & z_1 \\ f_2 & f_1 \\ f_3 & f_2 \end{pmatrix} \quad \Rightarrow \quad z_1^2, z_1 f_1, z_1 f_2, -z_1 f_2, -z_1 f_3, f_2^2 - f_1 f_3$$

where

$$\begin{aligned} f_1 &= z_2^2 - z_1 z_3 \\ f_2 &= z_2 z_3 - z_1^3 \\ f_3 &= z_3^2 - z_1^2 z_2 \\ \Rightarrow \quad f_2^2 - f_1 f_3 &= z_1 \underbrace{(z_1^5 - 3 z_1^2 z_2 z_3 + z_1 z_2^3 + z_3^3)}_{d} \end{aligned}$$

The matrix $F(z_1, z_2, z_3)$ defined by

$$F = \begin{pmatrix} f_2 & f_1 \\ f_3 & f_2 \end{pmatrix}$$

admits no polynomial decomposition $F = F_1 F_2$ such that neither $F_1(z_1, z_2, z_3)$ nor $F_2(z_1, z_2, z_3)$ are non-unimodular, i.e. for any polynomial factorisation $F = F_1 F_2$ either F_1 or F_2 is a unimodular matrix.

Now suppose that M_2 is not factor (right) coprime, i.e. there exists a polynomial matrix $D(z_1, z_2, z_3)$ such that

$$\begin{bmatrix} F(z_1, z_2, z_3) \\ z_1 I_2 \end{bmatrix} = \begin{bmatrix} A(z_1, z_2, z_3) \\ B(z_1, z_2, z_3) \end{bmatrix} D(z_1, z_2, z_3)$$

where $A(z_1, z_2, z_3)$, $B(z_1, z_2, z_3)$ are polynomial matrices. Thus $F = AD$ and by the previous paragraph either A or D is unimodular, but D can not be unimodular otherwise M_2 is factor (right) coprime, therefore suppose that A is unimodular. Thus $\det D = z_1 d$ where d is as defined earlier.

Now

$$\begin{aligned} \det(z_1 I_2) &= \det B \det D \\ z_1^2 &= b \times z_1 d \\ \Rightarrow \quad z_1 &= bd \\ \Rightarrow \quad d &| z_1 \end{aligned}$$

This is obviously false hence M_2 is factor (right) coprime.

By similar argument M_1 is also factor (left) coprime. Thus showing that a rational matrix can have a factor (left) coprime MFD and a factor (right) coprime MFD. □

Example 2.8: (Minor Right/Minor Left) Let

$$\begin{aligned} G &= \begin{pmatrix} \frac{z_2}{z_1} & \frac{1}{z_1} \\ \frac{1}{z_1} & \frac{z_3}{z_1} \end{pmatrix} \\ &= \begin{pmatrix} z_1 & 0 \\ 0 & z_1 \end{pmatrix}^{-1} \begin{pmatrix} z_2 & 1 \\ 1 & z_3 \end{pmatrix} \\ &= \begin{pmatrix} z_2 & 1 \\ 1 & z_3 \end{pmatrix} \begin{pmatrix} z_1 & 0 \\ 0 & z_1 \end{pmatrix}^{-1} \end{aligned}$$

The high-order minors of the compound matrices

$$M_1 = \begin{pmatrix} z_1 & 0 & z_2 & 1 \\ 0 & z_1 & 1 & z_3 \end{pmatrix} \qquad M_2 = \begin{pmatrix} z_1 & 0 \\ 0 & z_1 \\ z_2 & 1 \\ 1 & z_3 \end{pmatrix}$$

are given by

$$\begin{array}{llll} M_1{}_{1,2}^{1,2} = z_1^2 & M_1{}_{1,3}^{1,2} = z_1 & M_2{}_{1,2}^{1,2} = z_1^2 & M_2{}_{1,2}^{1,3} = z_1 \\ M_1{}_{1,4}^{1,2} = z_1 z_3 & M_1{}_{2,3}^{1,2} = -z_1 z_2 & M_2{}_{1,2}^{1,4} = z_1 z_3 & M_2{}_{1,2}^{2,3} = -z_1 z_2 \\ M_1{}_{2,4}^{1,2} = -z_1 & M_1{}_{3,4}^{1,2} = z_2 z_3 - 1 & M_2{}_{1,2}^{2,4} = -z_1 & M_2{}_{1,2}^{3,4} = z_2 z_3 - 1 \end{array}$$

Thus both M_1 and M_2 are minor coprime, as the high-order minors do not have a factor but if $(z_1, z_2, z_3) = (0, 1, 1)$ all the high-order minors are zero.

Therefore the left MFD is minor (left) coprime and the right MFD is minor (right) coprime. □

Example 2.9: (Zero Left/Zero Right) Let

$$\begin{aligned} G &= \begin{pmatrix} \frac{z_2}{z_1} & \frac{1}{z_1} \\ \frac{1}{z_1} & 0 \end{pmatrix} \\ &= \begin{pmatrix} z_1 & 0 \\ 0 & z_1 \end{pmatrix}^{-1} \begin{pmatrix} z_2 & 1 \\ 1 & 0 \end{pmatrix} \\ &= \begin{pmatrix} z_2 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} z_1 & 0 \\ 0 & z_1 \end{pmatrix}^{-1} \end{aligned}$$

The high-order minors of the compound matrices

$$M_1 = \begin{pmatrix} z_1 & 0 & z_2 & 1 \\ 0 & z_1 & 1 & 0 \end{pmatrix} \qquad M_2 = \begin{pmatrix} z_1 & 0 \\ 0 & z_1 \\ z_2 & 1 \\ 1 & 0 \end{pmatrix}$$

are given by

$$\begin{array}{llll} M_1{}^{1,2}_{1,2} = z_1^2 & M_1{}^{1,2}_{1,3} = z_1 & M_2{}^{1,2}_{1,2} = z_1^2 & M_2{}^{1,3}_{1,2} = z_1 \\ M_1{}^{1,2}_{1,4} = 0 & M_1{}^{1,2}_{2,3} = -z_1 z_2 & M_2{}^{1,4}_{1,2} = 0 & M_2{}^{2,3}_{1,2} = -z_1 z_2 \\ M_1{}^{1,2}_{2,4} = -z_1 & M_1{}^{1,2}_{3,4} = -1 & M_2{}^{2,4}_{1,2} = -z_1 & M_2{}^{3,4}_{1,2} = -1 \end{array}$$

Thus both M_1 and M_2 are zero coprime, as one of the minors is a constant.

Therefore the left MFD is zero (left) coprime and the right MFD is zero (right) coprime. □

2.3.3 n-D MFD Structure Theorem

Recall the equivalence possessed by the invariant polynomials of all coprime matrix fraction descriptions of a 2-D rational matrix, Theorem 2.5. In light of the results for coprime invariance of n-D MFDs it would seem improbable that the 2-D Structure Theorem (Theorem 2.5) holds for all n-D coprime MFDs. This fact is supported by the following example.

Example 2.10: Recall Example 2.4 with MFDs of the rational matrix $G(z_1, z_2, z_3)$ defined by

$$G(z_1, z_2, z_3) = \begin{pmatrix} \frac{z_2}{z_1} & \frac{z_3}{z_1} \end{pmatrix} = (z_1)^{-1} \begin{pmatrix} z_2 & z_3 \end{pmatrix} = \begin{pmatrix} z_2 & z_3 \end{pmatrix} \begin{pmatrix} z_1 & 0 \\ 0 & z_1 \end{pmatrix}^{-1}.$$

The two denominator matrices have determinants z_1 and z_1^2; thus demonstrating Corollary 1 of Theorem 2.5 does not hold for n-D factor coprime MFDs. □

However, by examining the proof of Theorem 2.5 it would seem to suggest that if both the left and right coprime MFDs possess some type of Bézout identity the theorem holds. In fact, if both MFDs are either minor or zero coprime the 2-D MFD Structure Theorem holds for n-D MFDs.

Theorem 2.12: *(n-D MFD Structure Theorem)*
Let $G_{p\times q}(z)$ be a n-D rational matrix and have a left and right MFD defined by

$$\begin{aligned} G(z) &= N_1(z) D_1^{-1}(z) \\ &= D_2^{-1}(z) N_2(z) \end{aligned} \tag{79}$$

where $N_1(z)$, $D_1(z)$ are minor (right) coprime and $N_2(z)$, $D_2(z)$ are minor (left) coprime.

(i) Let $d_1^{[1]}(z), d_2^{[1]}(z), \ldots, d_q^{[1]}(z)$ *denote the invariant polynomials of the* $q \times q$ *polynomial matrix* $D_1(z)$ *and* $d_1^{[2]}(z), d_2^{[2]}(z), \ldots, d_p^{[2]}(z)$ *denote the invariant polynomials of* $p \times p$ *polynomial matrix* $D_2(z)$ *then*

$$d_{q-i}^{[1]} = c_i d_{p-i}^{[2]} \quad \text{for} \quad i = 0, 1, \ldots, \max(p-1, q-1)$$

where $d_j^{[1]} = 1$, $d_j^{[2]} = 1$ *for* $j < 1$ *and* $c_i \in \mathbb{R}\backslash\{0\}$.

(ii) The $p \times q$ *polynomial matrices* $N_1(z)$ *and* $N_2(z)$ *have identical invariant polynomials, modulo a non-zero constant factor.*

Proof: The proof is analogous to that for two indeterminates. Thus it suffices to set up the equation (81), below, from which a similar argument following (30) of Theorem 2.5 completes the proof.

From the Bézout identities there exist polynomial matrices $X_i(z)$, $Y_i(z)$, $W_i(z)$ and $Z_i(z)$ for $i = 1, 2, \ldots, n$ of appropriate dimensions such that

$$\left.\begin{aligned} X_i D_1 + Y_i N_1 &= \psi_i(z_i^c) I_q \\ N_2 Z_i + D_2 W_i &= \phi_i(z_i^c) I_p \end{aligned}\right\} \quad \text{for} \quad i = 1, 2, \ldots, n. \tag{80}$$

where $\psi_i(z_i^c)$ and $\phi_i(z_i^c)$ are polynomials in the $n-1$ indeterminates $z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n$. From (79) and (80) it follows that

$$\begin{bmatrix} X_i & Y_i \\ N_2 & -D_2 \end{bmatrix} \begin{bmatrix} D_1 & Z_i \\ N_1 & -W_i \end{bmatrix} = \begin{bmatrix} \psi_i(z_i^c) I_q & J_i \\ 0 & \phi_i(z_i^c) I_p \end{bmatrix} \tag{81}$$

where $J_i = X_i Z_i - Y_i W_i$. □

The following corollaries follow directly from the proofs of the 2-D MFD Structure Theorem corollaries (Theorem 2.5), therefore the proofs are not given.

Corollary 1: *The determinant of the denominator matrices in (79) are equal modulo a multiplicative constant.* □

Corollary 2: *Let* $N_1(z)$, $D_1(z)$ *be minor (right) coprime and* $N_2(z)$, $D_2(z)$ *be minor (left) coprime then define* $G_{p\times q}(z)$ *as in (79) and*

$$U_i = \begin{bmatrix} X_i & Y_i \\ N_2 & -D_2 \end{bmatrix} \quad , \quad V_i = \begin{bmatrix} D_1 & Z_i \\ N_1 & -W_i \end{bmatrix} \quad \text{for} \quad i = 1, 2, \ldots, n. \tag{82}$$

Then

$$|U_i| = k_i \psi_i^q \quad , \quad |V_i| = l_i \phi_i^p \quad \text{where} \quad k_i, l_i \in \mathbb{R}\backslash\{0\} \quad \text{for} \quad i = 1, 2, \ldots, n. \quad \square$$

Corollary 3: *Express $G_{p\times q}(z)$ as in (79). Let $S_{N_1}(z)$, $S_{N_2}(z)$, $S_{D_1}(z)$, $S_{D_2}(z)$ be the Smith forms of $N_1(z)$, $N_2(z)$, $D_1(z)$, $D_2(z)$, respectively. Then*

(a) the Smith forms of the numerators are related by

$$S_{N_1} = CS_{N_2}$$

where $C_{p\times p} = \text{diag}(c_1, c_2, \ldots, c_p)$ with $c_i \in \mathbb{R}\backslash\{0\}$ for $i = 1, 2, \ldots, p$.

(b) the Smith forms of the denominators are related by

$$S_{D_1}F_1 = F_2S_{D_2}$$

where

(i) if $p > q$ $\qquad F_1 = [0_{q\times(p-q)} \quad I_q] \qquad F_2 = [0_{q\times(p-q)} \quad E_{q\times q}]$

where $E_{q\times q} = \text{diag}(e_1, e_2, \ldots, e_q)$ with $e_i \in \mathbb{R}\backslash\{0\}$ for $i = 1, 2, \ldots, q$;

(ii) if $p \leq q$ $\qquad F_1 = \begin{bmatrix} 0_{(q-p)\times p} \\ I_p \end{bmatrix} \qquad F_2 = \begin{bmatrix} 0_{(q-p)\times p} \\ E_{p\times p} \end{bmatrix}$

where $E_{p\times p} = \text{diag}(e_1, e_2, \ldots, e_p)$ with $e_i \in \mathbb{R}\backslash\{0\}$ for $i = 1, 2, \ldots, p$. □

Corollary 4: *If the n-D Structure Theorem does not hold then $G(z)$ has at least a one-sided factor coprime MFD.*

Proof: By the generalisation of Theorem 2.1 to n-D rational matrices there exist a left and a right coprime MFD for any rational matrix. In particular, let $G(z)$ possess a left coprime MFD $D_1^{-1}N_1$ and a right coprime MFD $N_2D_2^{-1}$. If $N_1(z)$, $D_1(z)$ are and $N_2(z)$, $D_2(z)$ are not factor (right) coprime then the n-D Structure Theorem holds. Therefore, either $N_1(z)$, $D_1(z)$ are factor (left) coprime or $N_2(z)$, $D_2(z)$ are factor (right) coprime. □

The proof of Theorem 2.12 and Example 2.10 provide some information about the possible form of a Bézout identity for n-D factor coprime matrices. Theorem 2.10 does not necessarily hold if either the left or right MFD is factor coprime, as demonstrated by Example 2.10. However, this is not necessarily true for all factor coprime MFDs, as demonstrated by considering the numerator and denominator matrices in Example 2.7, i.e. the denominators both have Smith form given by

$$\begin{pmatrix} z_1 & 0 \\ 0 & z_1 \end{pmatrix}$$

and the numerators both have Smith form given by

$$\begin{pmatrix} 1 & 0 \\ 0 & z_1(z_1^5 - 3z_1^2 z_2 z_3 + z_1 z_2^3 + z_3^3) \end{pmatrix}$$

Theorem 2.12 is dependent on the coprimeness of the determinants of the right-hand-side of the Bézout identity. The right-hand-side of the minor coprime Bézout identity, (Theorem 1.4(ii))

$$A(z)X_i(z) + B(z)Y_i(z) = \psi_i(z_i^c)I_p \qquad \text{for} \qquad i = 1, 2, \ldots, n$$

has determinants given by $\psi_i^p(z_i^c)$ for $i = 1, 2, \ldots, n$ and since each $\psi_i^p(z_i^c)$ is independent of z_i these polynomials are factor coprime. Thus if a Bézout identity exists for factor coprime matrices the determinants of the right-hand-side can not be coprime, otherwise Theorem 2.12 can be proved.

Two further restrictions on a possible form of the factor coprime Bézout identity are provided by Theorem 1.3. Firstly for $n = 1$ the Bézout identity must deliver the 1-D Bézout identity, i.e.

$$A(z)X(z) + B(z)Y(z) = I$$

and secondly it must contain, as a special case, the Bézout identities for n-D minor coprime and zero coprime matrices. Thus n equations, at least, are necessary to define a Bézout identity for factor coprimeness.

For any pair of matrices with the same number of rows $A_{p\times q}(z)$, $B_{p\times r}(z)$ it is possible to form the Smith form over the ring $\mathbb{R}(z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n)[z_i]$ for $i = 1, 2, \ldots, n$. This defines n matrices. Clearly, if $A(z)$, $B(z)$ are minor (left) coprime each of the Smith forms is the identity matrix with extra columns of zeros to form a matrix with the same size as $M(z) = [A(z)\ B(z)]$. The ring of polynomials $\mathbb{R}(z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n)[z_i]$ contains a division algorithm, therefore there exist $U_i(z)$, $V_i(z)$, unimodular matrices in this ring, with elements rational in $z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n$ and polynomial in z_i and sizes $p \times p$ and $(q + r) \times (q + r)$ such that

$$U_i[A\ B]V_i = [I_p\ 0] \quad \text{for} \quad i = 1, 2, \ldots, n$$

Thus

$$[A\ B]V_i \begin{bmatrix} I_p \\ 0 \end{bmatrix} U_i^{-1} = I_p \quad \text{for} \quad i = 1, 2, \ldots, n$$

or equivalently

$$AX_i + BY_i = I \quad \text{for} \quad i = 1, 2, \ldots, n$$

Now by multiplying through by a polynomial in $z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n$ $\psi_i(z_i^c)$, X_i and Y_i possess no rational elements. Thus the Bézout identity is formed.

Thus for any matrix pair $A_{p\times q}(z)$, $B_{p\times r}(z)$ it is possible to form

$$[A\ B]V_i = \psi_i(z_i^c)U_i^{-1}S_i \quad \text{for} \quad i = 1,2,\ldots,n \tag{83}$$

where $V_i(z)$ is a $(q+r)\times p$ polynomial matrix over $\mathbb{R}[z_1,z_2,\ldots,z_n]$, U_i is a polynomial matrix over $\mathbb{R}(z_1,\ldots,z_{i-1},z_{i+1},\ldots,z_n)[z_i]$, $\psi_i(z_i^c)$ is a polynomial over $\mathbb{R}[z_1,\ldots,z_{i-1},z_{i+1},\ldots,z_n]$ and $S_i(z)$ is a diagonal square matrix with the invariant polynomials of $[A\ B]$ over $\mathbb{R}(z_1,\ldots,z_{i-1},z_{i+1},\ldots,z_n)[z_i]$. The ultimate question now is "under what necessary and sufficient conditions does the equation (83) define factor (left) coprimeness?". This is an open research problem.

It may be conjectured that necessary and sufficient conditions for factor (left) coprimeness is that there exist polynomial matrices $X_i(z)$, $Y_i(z)$ with elements in $\mathbb{R}[z_1,z_2,\ldots,z_n]$ such that

$$A(z)X_i(z) + B(z)Y_i(z) = \psi_i(z_i^c)S_i(z) \quad \text{for} \quad i = 1,2,\ldots,n \tag{84}$$

where $\psi_i(z_i^c) \in \mathbb{R}[z_1,\ldots,z_{i-1},z_{i+1},\ldots,z_n]$ and $S_i(z)$ is defined as in (83). However, this conjecture is false, as demonstrated by the following example.

Example 2.11: The factor (left) coprime matrix (Example 2.5)

$$\begin{pmatrix} z_1 & 0 & z_2 \\ 0 & z_1 & z_3 \end{pmatrix}$$

has Smith forms

$$S_1 = \begin{pmatrix} 1 & 0 \\ 0 & z_1 \end{pmatrix}, \quad S_2 = S_3 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

over $\mathbb{R}(z_2,z_3)[z_1]$, $\mathbb{R}(z_1,z_3)[z_2]$ and $\mathbb{R}(z_1,z_2)[z_3]$ respectively. It is possible to form (84) over $\mathbb{R}(z_1,z_3)[z_2]$ and $\mathbb{R}(z_1,z_2)[z_3]$, i.e.

$$\begin{pmatrix} z_1 & 0 & z_2 \\ 0 & z_1 & z_3 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} = z_1 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

However, it is not possible to form (84) over $\mathbb{R}(z_2,z_3)[z_1]$ as indicated by the following discussion.

Suppose that there exists a polynomial matrix

$$X(z) = \begin{pmatrix} a & b \\ c & d \\ e & f \end{pmatrix}$$

such that

$$\begin{pmatrix} z_1 & 0 & z_2 \\ 0 & z_1 & z_3 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \\ e & f \end{pmatrix} = \psi_1(z_2, z_3) \begin{pmatrix} 1 & 0 \\ 0 & z_1 \end{pmatrix} \tag{85}$$

where $a, b, c, d, e, f \in \mathbb{R}[z_1, z_2, z_3]$. The constituent equations of (85) are

$$az_1 + ez_2 = \psi_1 \tag{86}$$

$$bz_1 + fz_2 = 0 \tag{87}$$

$$cz_1 + ez_3 = 0 \tag{88}$$

$$dz_1 + fz_3 = z_1\psi_1 \tag{89}$$

Now from equations (86) and (88), i.e. $z_3(86) - z_2(88)$

$$z_1(az_3 - cz_2) = \psi_i z_3$$

thus $z_1 | \psi_i(z_2, z_3)$, which is a contradiction. □

2.4 Conclusions

In this chapter the matrix fraction descriptions of 2-D and n-D rational matrices have been considered. The more general results for n-D matrices have complicated theorem statements due to the complexity of polynomial matrices in more than two indeterminates. However, when these theorems are considered in the less general case of $n = 2$ they become more compact.

In particular, the MFD Structure Theorem (Theorem 2.12) only holds for minor or zero coprime MFDs and not for factor coprime MFDs. When this theorem is considered for the less general case of $n = 2$ the definitions of factor and minor coprimeness are coincident, thus the theorem holds for every type of coprimeness. The same situation arises when considering the types of coprimeness possessed by opposite sided MFDs: for $n \geq 3$ it is possible for a rational matrix to possess different types of coprimeness for a left and a right MFD; this is not the case for $n = 2$.

Chapter 3

Equivalence of Polynomial Matrices

3.1 Introduction

This chapter will investigate the properties of certain types of equivalence that exist between classes of matrices. Two of the most fundamental requirements for the equivalence of two system-representations is the same input-output behaviour and the invariance of the system matrix zero structure.

The invariance of the input-output behaviour is guaranteed by the equality of the transfer function matrices. This type of equivalence has been extensively researched for 1-D systems. Rosenbrock [12] introduces two types of equivalence, namely strict

system equivalence and system equivalence. The latter is an algorithmically based necessary and sufficient condition for invariant input-output behaviour, whilst the former is a sufficient closed matrix form condition, which may be shown to be a special case of system equivalence. Subsequently, Pugh *et al* [46] have provided a closed matrix form representation of system equivalence known as extended strict system equivalence which is based on rational rectangular transforming matrices. The special case of strict system equivalence is based on square polynomial transforming matrices. Further studies of this transformation are contained in [47]–[52]. The purpose of this work is to consider some of the 1-D ideas in a more general n-D framework and to propose generalised definitions that will facilitate a deeper understanding of the structure of n-D system matrices.

The important notion of least order for 1-D systems, defined as the relative primeness of the T, U and T, V matrix pairs, is fundamental to the existence of strict system equivalence between two system matrices. The evidence from the previous chapter provides an indication of the nature of the problem inherent in defining an equivalent definition for least order n-D system matrices and the subsequent analogue of strict system equivalence. This will be addressed later.

The second fundamental requirement for equivalence, mentioned above, is the invariance of the system matrix zero structure. The problem of determining the conditions under which the finite zeros remain invariant has been much studied and many conditions have been provided, including the one presented here. The generalised theory arises from a desire to analyse the point at infinity on the same basis as the finite points, this has been studied in [51], [53]–[59]. To establish the invariance of finite and infinite zero structure under the recently defined full equivalence [60] a complicated lemma is required; here a simplified proof is given and a link made to conditions proposed by Zhang [61] which guarantee the non-existence of infinite zeros.

The chapter is composed of four main sections all linked by the general theme of equivalence of polynomial matrices. In the first, Section 3.2, the invariance of finite and infinite zeros [62], [63] is investigated and a link made with recent work which guarantees the absence of infinite zeros.

Polynomial transformations of 2-D systems are the subject of Section 3.3; addressing the problem of invariant properties of system matrices. A generalised definition of least order, based on coprimeness, is considered and a 2-D analogue of extended strict

system equivalence is proposed. In Section 3.4 polynomial transformations of n-D systems matrices are considered. In particular three types of polynomial transformation can be immediately formulated and the extent to which these transformations define equivalence relations is investigated. In the final section, 3.5, a specific type of 2-D system is considered in terms of the existence of zeros and a least order realisation, as defined for general 2-D systems in Section 3.3.

3.2 Some Aspects of Equivalence of 1-D Systems

3.2.1 Some Conditions Guaranteeing Identical Zero Structure

In the conventional theory of linear systems and in the context of polynomial models, the following equivalence transformation plays a fundamental role [50], [52].

Consider the set $\mathfrak{P}(m,\ell)$ of $(r+m)\times(r+\ell)$ polynomial matrices where r is any integer greater than $\max(-m,-\ell)$.

Definition 3.1: (*Extended Unimodular Equivalence*)
Let $T_i(s)\in\mathfrak{P}(m,\ell)$ for $i=1,2$ be polynomial matrices in the single indeterminate s, i.e. $r_i\geq\max(-m,-\ell)$ for $i=1,2$. Then $T_1(s)$, $T_2(s)$ are said to be *extended unimodular equivalent* if there exist polynomial matrices $M(s)$, $N(s)$ such that

$$M(s)T_1(s)=T_2(s)N(s) \tag{90}$$

or equivalently

$$[M(s)\quad T_2(s)]\begin{bmatrix}T_1(s)\\-N(s)\end{bmatrix}=0 \tag{91}$$

where the compound matrices

$$[M(s)\quad T_2(s)]\quad;\qquad\begin{bmatrix}T_1(s)\\-N(s)\end{bmatrix} \tag{92}$$

have full normal rank and no finite zeros. □

Note: The rank and zero condition on the compound matrices (92) is equivalent to the matrices in the first compound matrix being relatively (left) prime and those in the second matrix being relatively (right) prime.

The basic result concerning this transformation and the reason for its importance is the following.

Lemma 3.1: *Let $T_i(s)\in\mathfrak{P}(m,\ell)$ for $i=1,2$ then $T_1(s)$, $T_2(s)$ are extended unimodular equivalent if and only if they have the same finite zero structure.*

Proof: See [50], [52]. □

In the generalised theory of linear systems it is necessary that the point at infinity is treated on the same basis as the finite points of the complex plane $\mathbb{C}$. It is clear therefore that in this context, extended unimodular equivalence will not play such a fundamental role since its complete and independent invariants relate solely to the finite zero structure. Accordingly some further restriction of the action of extended unimodular equivalence will be required in order to obtain a transformation relevant to the generalised theory. The following definition has been proposed [56], [60] [64], [65].

Definition 3.2: (*Full Equivalence*)
Let $T_i(s) \in \mathfrak{P}(m,\ell)$, for $i = 1,2$. Then $T_1(s)$, $T_2(s)$ are said to be *fully equivalent* if there exist polynomial matrices $M(s), N(s)$ such that

$$[M(s) \quad T_2(s)] \begin{bmatrix} T_1(s) \\ \\ -N(s) \end{bmatrix} = 0 \tag{93}$$

where the compound matrices (92)

(i) have full normal rank;
(ii) have no finite nor infinite zeros;
(iii) satisfy the McMillan degree conditions given by

$$\delta([M(s) \quad T_2(s)]) = \delta(T_2(s)) \quad ; \qquad \delta\left(\begin{bmatrix} T_1(s) \\ \\ -N(s) \end{bmatrix}\right) = \delta(T_1(s)) \tag{94}$$

where $\delta(\cdot)$ denotes the McMillan Degree of $(\cdot)$ [12], [66]. □

The unexpected thing about this definition is the requirement for a condition such as (iii). However, it is quite easy to construct examples to demonstrate that in order to simultaneously preserve both the finite and infinite zero structures the conditions (i) and (ii) are not sufficient. The following example from [64] demonstrates this.

Example 3.1: Consider

$$\underbrace{\begin{pmatrix} s-1 & s+1 \\ s & 0 \end{pmatrix}}_{M(s)} \underbrace{\begin{pmatrix} 1 & s \\ 0 & 1 \end{pmatrix}}_{T_1(s)} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{T_2(s)} \underbrace{\begin{pmatrix} s-1 & s^2+1 \\ s & s^2 \end{pmatrix}}_{N(s)} \tag{95}$$

Consider first the compound matrix $[M(s)\ T_2(s)]$, i.e.

$$\begin{pmatrix} s-1 & s+1 & 1 & 0 \\ s & 0 & 0 & 1 \end{pmatrix}. \tag{96}$$

The Smith form is given by

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

therefore (96) has full normal rank and no finite zeros. The Smith-McMillan form [12] of (96) with $s = 1/w$ is

$$\begin{pmatrix} \frac{1}{w} & 0 & 0 & 0 \\ 0 & \frac{1}{w} & 0 & 0 \end{pmatrix}$$

which demonstrates the absence of infinite zeros. Secondly, the compound matrix

$$\begin{bmatrix} T_1(s) \\ -N(s) \end{bmatrix} = \begin{pmatrix} 1 & s \\ 0 & 1 \\ s-1 & s^2+1 \\ s & s^2 \end{pmatrix} \tag{97}$$

has Smith form

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

and Smith-McMillan form with $w = 1/s$

$$\begin{pmatrix} \frac{1}{w^2} & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

Demonstrating that (97) has full normal rank, no finite zeros and no infinite zeros.

Therefore (96) and (97) satisfy conditions (i) and (ii). However, $T_1(s)$, $T_2(s)$ do not have the same infinite zero structure, as demonstrated by the following discussion.

The infinite zeros of $T_1(s)$, $T_2(s)$ are given by considering the Smith-McMillan forms of $T_1(\frac{1}{w})$, $T_2(\frac{1}{w})$. These are respectively given by

$$S_{T_1} = \begin{pmatrix} \frac{1}{w} & 0 \\ 0 & w \end{pmatrix}, \qquad S_{T_2} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Therefore $T_1(s)$ has an infinite zero of degree 1 and $T_2(s)$ does not possess any infinite zeros.

Notice that

$$\delta([\, M(s) \quad T_2(s)\,]) = 2 \neq 0 = \delta(T_2(s))$$
$$\delta(\begin{bmatrix} T_1(s) \\ -N(s) \end{bmatrix}) = 2 \neq 1 = \delta(T_1(s))$$

and so the McMillan degree conditions (iii) are not satisfied. □

The sufficiency of the three conditions (i) - (iii) in this respect has been established in [64], although the proof is somewhat dependent on a complicated lemma. A previously unpublished simpler proof is given here reproduced from [57]. Recall that the least order, $\nu(G)$, of a rational matrix $G(s)$ is its total number of finite poles, while the McMillan degree of $G(s)$, denoted $\delta(G)$, is its total number of finite and infinite poles.

Theorem 3.1: *If $T_i(s) \in \mathfrak{P}(m,\ell)$, for $i = 1,2$ are fully equivalent, then they possess identical finite and infinite zero structures.*

Proof: Suppose $T_i(s)$ $(i = 1,2)$ are fully equivalent, then from conditions (i) and (ii) of Definition 3.2, they are clearly extended unimodular equivalent and so by Lemma 3.1 have identical finite zero structures.

Let

$$\left.\begin{aligned} T_2\left(\frac{1}{w}\right) &= \tilde{D}_2^{-1}(w)\,\tilde{N}_2(w) \\ T_1\left(\frac{1}{w}\right) &= \tilde{N}_1(w)\,\tilde{D}_1^{-1}(w) \end{aligned}\right\} \tag{98}$$

be two relatively prime factorisations. Since $T_2(s)$ is polynomial it has no finite poles, only infinite ones. Thus $T_2\left(\frac{1}{w}\right)$ only has poles at $w = 0$, and so

$$\delta(T_2(s)) = \nu\left(T_2\left(\frac{1}{w}\right)\right) \tag{99}$$

Further since (98) are relatively prime factorisations it follows that

$$\delta\left(\mid \tilde{D}_2(w) \mid\right) = \nu\left(T_2\left(\frac{1}{w}\right)\right). \tag{100}$$

Now suppose that

$$\left.\begin{aligned} \left[M\left(\frac{1}{w}\right) \quad T_2\left(\frac{1}{w}\right)\right] &= D_2^{-1}(w)[N_{21}(w) \quad N_{22}(w)] \\ \begin{bmatrix} T_1\left(\frac{1}{w}\right) \\ -N\left(\frac{1}{w}\right) \end{bmatrix} &= \begin{bmatrix} N_{11}(w) \\ N_{12}(w) \end{bmatrix} D_1^{-1}(w) \end{aligned}\right\} \tag{101}$$

are relatively prime factorisations, then

$$\begin{aligned} \delta(\mid D_2(w) \mid) &= \nu\left(\left[M\left(\frac{1}{w}\right) \quad T_2\left(\frac{1}{w}\right)\right]\right) \\ &= \delta\left([M(s) \quad T_2(s)]\right) \\ &= \delta(T_2(s)) \end{aligned} \tag{102}$$

by condition (iii) of Definition 3.2. It follows therefore from (99) and (102) that

$$\delta\left(\mid D_2(w) \mid\right) = \nu\left(T_2\left(\frac{1}{w}\right)\right) \tag{103}$$

and hence from (101) that

$$D_2^{-1}(w)\,N_{22}(w) = T_2\left(\frac{1}{w}\right) \tag{104}$$

is a prime factorisation of $T_2\left(\frac{1}{w}\right)$. Thus $N_{22}(w)$ is a numerator of $T_2\left(\frac{1}{w}\right)$, and its zero structure at $w = 0$ is the infinite zero structure of $T_2(s)$.

In an entirely analogous manner it may be established that $N_{11}(w)$ of (101) is a numerator of $T_1\left(\frac{1}{w}\right)$, and that its zero structure at $w = 0$ represents the infinite zero structure of $T_1(s)$.

Substitute (101) into (93) to give, by pre- and post-multiplication by $D_2(w)$, $D_1(w)$ respectively,

$$[N_{21}(w) \quad N_{22}(w)] \begin{bmatrix} N_{11}(w) \\ \\ N_{12}(w) \end{bmatrix} = 0 \tag{105}$$

and by condition (ii) of Definition 2, the compound matrices (92) have no infinite zeros. Accordingly the compound matrices in (105) have full rank at $w = 0$. Additionally they also have full rank for every other finite value of w, since the compound matrices (92) have no finite zeros. Consequently in (105)

$$[N_{21}(w) \quad N_{22}(w)]; \qquad \begin{bmatrix} N_{11}(w) \\ \\ N_{12}(w) \end{bmatrix} \tag{106}$$

have full normal rank and no finite zeros. It thus follows that (105) is a statement of extended unimodular equivalence between $N_{22}(w)$ and $N_{11}(w)$, which therefore have identical finite zero structures, and specifically have identical zero structures at $w = 0$. Hence $T_1(s), T_2(s)$ have identical infinite zero structures. □

Thus from this theorem the matrices $T_1(s)$, $T_2(s)$ in Example 3.1 can not be connected by a transformation of full equivalence, since they do not have the same infinite zero structure. However, if the roles of the matrices in Example 3.1 are interchanged, i.e. $T_1(s)$, $T_2(s)$ are the transforming matrices and $M(s)$, $N(s)$ are the transformed matrices, all three conditions of Definition 3.2 are satisfied, since $\delta(M(s)) = 2 = \delta(N(s))$ and the conditions (i) and (ii) are satisfied by the discussion above. Thus ensuring the same finite and infinite zero structure of $M(s)$ and $N(s)$. This is now directly established.

The matrices $M(s)$, $N(s)$ have Smith form

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

therefore they have the same finite zero structure. The matrices $M(s)$, $N(s)$ for $s = 1/w$ have Smith-McMillan form S_M, S_N (respectively) given by

$$S_M = \begin{pmatrix} \frac{1}{w} & 0 \\ 0 & \frac{1+w}{w} \end{pmatrix} \qquad S_N = \begin{pmatrix} \frac{1}{w^2} & 0 \\ 0 & w-1 \end{pmatrix}$$

Therefore the infinite zero structure is also the same, i.e. they do not have any infinite zeros.

The McMillan degree seems an unexpected condition in the definition of full equivalence but as has already been seen, Example 3.1 and Theorem 3.1, that it is sufficient to guarantee the equivalence of the finite and infinite zero structure. It is thus instructive to note the precise interpretation of this condition in terms of mappings recently derived by Pugh *et. al.* [67]. This is as follows.

The McMillan degree condition

$$\delta\left(\begin{bmatrix} T_1(s) \\ \\ N(s) \end{bmatrix}\right) = \delta(T_1(s)) \tag{107}$$

may be interpreted as the necessary and sufficient conditions which guarantee that

$$\xi_1(t) = N(\rho)\xi(t) \tag{108}$$

is a formal mapping (in the sense of being a many to one relation) from the set χ_u of all solutions $\xi(t)$ of the differential equation

$$T_1(\rho)\xi(t) = 0$$

corresponding to all possible initial conditions of $\xi(t)$ and its $k-1$ derivatives, onto another set $\bar{\chi}_u$.

Thus if the matrix

$$\begin{bmatrix} T_1(s) \\ \\ N(s) \end{bmatrix} \tag{109}$$

satisfies the McMillan degree condition

$$\delta\left(\begin{bmatrix} T_1(s) \\ \\ N(s) \end{bmatrix}\right) = \delta(T_1(s))$$

and

(a) has no finite zeros then the strictly proper part of the inverse map of (108) is uniquely determined;

(b) has no infinite zeros then the polynomial part of the inverse map of (108) is uniquely determined.

Therefore the mapping (108) is injective if the matrix (109) has no infinite and no finite zeros.

3.2.2 Conditions For The Absence Of Infinite Zeros.

In a recent paper, Zhang [61] has developed a necessary and sufficient condition for the absence of infinite zeros in a polynomial matrix. The condition is interesting in the context of the previous section not least for the fact that it has a clear connection with the McMillan degree conditions which arise in the definition of full equivalence.

It is clear from the proof of Theorem 3.1 that the McMillan degree conditions ensure that the matrix relationship (91), through which the finite zero structures are explicitly related, has contained within itself a readily derivable relationship (106) connecting the infinite zero structures. In these terms it is not surprising therefore that a form of the McMillan degree conditions (iii) of Definition 3.2 arise quite naturally in the work of Zhang [61] concerning the absence of infinite zeros in a polynomial matrix. To establish this connection the following lemma is required.

Lemma 3.2: *Let $T(s) \in \mathfrak{P}(m,\ell)$ have full normal rank and let $p = \min(m,\ell)$. Then $T(s)$ has no finite (respectively infinite) zeros if and only if there is a $p \times p$ minor of $T(s)$ with degree zero (respectively $\delta(T(s))$).*

Proof: The case of the finite zeros is of course extremely well-known, while the case of the infinite zeros was originally established in [62]. □

To establish this connection let $T(s) \in \mathfrak{P}(m,\ell)$, with $m \leq \ell$ and let $V(s)$ be any $m \times m$ unimodular matrix such that

$$\bar{T}(s) = V(s)T(s), \tag{110}$$

where $\bar{T}(s)$ is row proper [13]. Let $\delta_{r_i}(\cdot)$ denote the i-th row degree of the indicated matrix then:

Theorem 3.2: *With the above notation write* (110) *in the form*

$$[\bar{T}(s) \quad V(s)] \begin{bmatrix} -I \\ \\ T(s) \end{bmatrix} = 0 \tag{111}$$

Then (111) *defines a transformation of full equivalence if and only if*

$$\delta_{r_i}(V(s)) \leq \delta_{r_i}(\bar{T}(s)) \tag{112}$$

Proof: Assume that (111) is indeed a transformation of full equivalence. In particular the McMillan degree conditions which apply to this transformation give, in relation to the compound matrix $[\bar{T}(s) \quad V(s)]$,

$$\delta([\bar{T}(s) \quad V(s)]) = \delta(\bar{T}(s)). \tag{113}$$

Since $\bar{T}(s)$ is row proper it follows that

$$\delta_{r_i}(V(s)) \leq \delta_{r_i}(\bar{T}(s)). \tag{114}$$

Conversely assume that (112) holds and consider the compound matrix

$$\begin{bmatrix} -I \\ T(s) \end{bmatrix}. \tag{115}$$

Clearly this has full normal rank and no finite zeros because of the unit matrix I. Further the McMillan degree condition in respect of this matrix is obviously satisfied. Finally any minor of $T(s)$ with degree $\delta(T(s))$ can be enlarged to an $\ell \times \ell$ minor of the same degree by including complementary rows from the unit matrix I. Hence, since the McMillan degree of (115) is $\delta(T(s))$ and there exists an $\ell \times \ell$ minor of this degree in (115), it follows, by Lemma 3.2, that the compound matrix (115) has no infinite zeros.

Consider now the compound matrix

$$[\bar{T}(s) \quad V(s)]. \tag{116}$$

Clearly this has full normal rank and no finite zeros since $V(s)$ is unimodular.

Now the condition (112) implies that

$$\delta_{r_i}([\bar{T}(s) \quad V(s)]) = \delta_{r_i}(\bar{T}(s)) \tag{117}$$

and since $\bar{T}(s)$ is row proper it follows therefore that $[\bar{T}(s) \quad V(s)]$ is row proper. Thus

$$\delta([\bar{T}(s) \quad V(s)]) = \sum_{i=1}^{m} \delta_{r_i}([\bar{T}(s) \quad V(s)]) \tag{118}$$

and

$$\delta(\bar{T}(s)) = \sum_{i=1}^{m} \delta_{r_i}(\bar{T}(s)). \tag{119}$$

It follows immediately from (117), (118), (119) that $[\bar{T}(s) \quad V(s)]$ satisfies the required McMillan degree condition i.e.

$$\delta([\bar{T}(s) \quad V(s)]) = \delta(\bar{T}(s)). \tag{120}$$

Further since $[\bar{T}(s) \quad V(s)]$ is row proper, it possesses an $m \times m$ minor with degree $\delta([\bar{T}(s) \quad V(s)])$, and so (116) has no infinite zeros by Lemma 3.2. □

An immediate consequence of the above result is

Corollary: *In the notation of Theorem 3.2, $T(s)$ has no infinite zeros if* (112) *holds.*

Proof: Clearly if (112) holds then by Theorem 3.2, (111) is a transformation of full equivalence between $T(s)$ and $\bar{T}(s)$. Hence by Theorem 3.1, $T(s)$ and $\bar{T}(s)$ have identical finite and infinite zero structures. However, $\bar{T}(s)$ is row proper and so possesses no infinite zeros. Hence $T(s)$ has no infinite zeros. □

It is clear from Theorem 3.2 and its Corollary that the row degree condition (112) is simply the McMillan degree condition of full equivalence in the context of the specific relationship (110). It follows therefore that Zhang's condition (112) is the mechanism which establishes that the given polynomial matrix $T(s)$ is fully equivalent to a row proper matrix $\bar{T}(s)$. Since such a matrix possesses no infinite zeros, it follows that $T(s)$ has this property. In this manner, therefore, it is seen that the problem of determining conditions under which a polynomial matrix has no infinite zeros may be viewed as a special case of the more general problem answered by Theorem 3.2.

One other point to be noted is that the condition (112) is not only sufficient for the absence of infinite zeros as the Corollary has shown, but also necessary, as Zhang [61] has established. The necessity of (112) for the absence of infinite zeros does not emerge in the Corollary because it has not as yet proved possible to establish full equivalence as a necessary condition in Theorem 3.1.

The requirement that a polynomial matrix possesses no infinite zeros is important in linear systems theory in areas such as composite system studies [68], system invertibility and minimality of system descriptions, besides its specific relevance to the transformation of full equivalence. It is therefore appropriate here to note that several equivalent conditions guaranteeing the absence of infinite zeros have been established. Thus for example it has been shown in [55] that $T(s)$ will possess no infinite zeros if and only if it has at least one $p \times p$ minor with degree $\delta(T(s))$. On the other hand it has been noted in [69] that an equivalent condition based on the ranks of two Toeplitz matrices formed from $T(s)$ (viewed as a matrix polynomial). Accordingly let

$$T(s) = T_n s^n + T_{n-1} s^{n-1} + \cdots + T_1 s + T_0 \tag{121}$$

be an $m \times \ell$ polynomial matrix of rank p and define the Toeplitz matrices T_{-i}^{∞}, with constant elements as

$$T_{-i}^{\infty} = \begin{bmatrix} T_n & T_{n-1} & \cdots & T_{i+1} & T_i \\ 0 & T_n & \cdots & T_{i+2} & T_{i+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & T_n & T_{n-1} \\ 0 & 0 & \cdots & 0 & T_n \end{bmatrix} \tag{122}$$

Theorem 3.3: *The polynomial matrix $T(s)$ of (121) possesses no infinite zeros if and only if one of the following equivalent conditions is satisfied:*

(i) there exists a $p \times p$ minor of $T(s)$ with degree $\delta(T(s))$;

(ii) the rank of T_0^{∞} is equal to the rank of $T_{-1}^{\infty} + p$; (123)

Additionally if $p = m$ then the following are mutually equivalent to (i) and (ii),

(iii) for any unimodular matrix $V(s)$ such that $\bar{T}(s)$ in (110) is row proper

$$\delta_{r_i}(V(s)) \leq \delta_{r_i}(\bar{T}(s)); \tag{124}$$

(iv) $T(s)$ has a proper right inverse.

Proof: The condition (i) is a consequence of the degree structure of the minors of $T(s)$ described in [55], while condition (ii) has recently been established in [69]. The conditions (iii) and (iv) are derived in [61]. □

The above conditions are each necessary and sufficient for the absence of infinite zeros in the polynomial matrix $T(s)$, however from a practical point of view some will be simpler to check than others. In this respect it has been noted in [61] that the condition (iii) is the simplest to evaluate. However, in view of the simplicity of forming the matrix polynomial (121) associated with $T(s)$, and the associated Toeplitz matrices $T_0^{\infty}(T)$, $T_{-1}^{\infty}(T)$, the condition (123) must be computationally simpler still, as has been noted in [55].

3.2.3 1-D System Matrices

A particular case of the set of matrices $\mathfrak{P}(m, \ell)$ is the Rosenbrock system matrix, $P_{(r+\ell)\times(r+m)}(s)$

$$P(s) = \begin{pmatrix} T(s) & U(s) \\ -V(s) & W(s) \end{pmatrix}$$

where $T_{r\times r}(s)$, $U_{r\times \ell}(s)$, $V_{m\times r}(s)$ and $W_{m\times \ell}(s)$ are polynomial matrices, arising from a transfer function matrix, $G(s)$, representation of

$$G(s) = V(s)T^{-1}(s)U(s) + W(s).$$

A basic requirement for the equivalence of two system matrices is that they possess the same input-output behaviour. This requirement translates to mean that the two system matrices possess the same transfer function matrix and is called *input-output equivalence.*

Many types of equivalence transformations have been defined for 1-D systems, for example, [38], [40], [42], [70], [71]. Two basic types of equivalence were originally defined by Rosenbrock [12] for system matrices. The first, strict system equivalence, is characterised by a closed matrix form, whereby two system realisations $P_1(s)$, $P_2(s)$ are said to be strictly system equivalent if they are related by an equation of the form

$$\begin{pmatrix} M(s) & 0 \\ X(s) & I_m \end{pmatrix} \underbrace{\begin{pmatrix} T_2(s) & U_2(s) \\ -V_2(s) & W_2(s) \end{pmatrix}}_{P_2(z_1,z_2)} \begin{pmatrix} N(s) & Y(s) \\ 0 & I_\ell \end{pmatrix} = \underbrace{\begin{pmatrix} T_1(s) & U_1(s) \\ -V_1(s) & W_1(s) \end{pmatrix}}_{P_1(z_1,z_2)} \quad (125)$$

where the two system matrices have sizes $(r+m)\times(r+\ell)$ and $M_{r\times r}(s)$, $N_{r\times r}(s)$ are unimodular polynomial matrices and $X_{m\times r}(s)$, $Y_{r\times \ell}(s)$ are polynomial matrices. This is only a sufficient condition for input-output equivalence because two system matrices may have the same transfer function matrix but have different sizes. For example, the transfer function matrix $G(s) = 1/s$ may be realised as

$$P_1(s) = \left(\begin{array}{c|c} s & 1 \\ \hline -1 & 0 \end{array}\right) \qquad P_2(s) = \left(\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & s & 1 \\ \hline 0 & -1 & 0 \end{array}\right).$$

However, strict system equivalence is a necessary and sufficient condition for two least order system matrices to be input-output equivalent. (Note a least order system matrix is defined to have no decoupling zeros or equivalently the matrices $T(s)$, $U(s)$ are relatively (left) prime and $T(s)$, $V(s)$ are relatively (right) prime.)

The second type, system equivalence, is a necessary and sufficient condition for input-output equivalence based on elementary operations over the field $F(s)$, i.e. rational functions in s. Two system matrices $P_1(s)$, $P_2(s)$ are said to be system equivalent if one can be obtained from the other by performing the following operations any number of (finite) times and in any order.

(a) Multiply any one of the first r rows (respectively columns) by a rational function not identically zero;

(b) Add a multiple, by a rational function, of any one of the first r rows (respectively columns) to any other row (respectively column);

(c) Interchange any two among the first r rows (respectively columns);

(d) Add a row and column to P to form

$$\begin{pmatrix} 1 & 0_{1\times(r+\ell)} \\ 0_{(r+m)\times 1} & P(s) \end{pmatrix} \tag{126}$$

or if the matrix has the form (126) delete the first row and first column.

The operation (d) allows system matrices of different sizes to be equivalent, thus overcoming the problem encountered with strict system equivalence. However, the transformation is no-longer polynomial. Thus the closed matrix form [46] of system equivalence is based on rational transforming matrices, i.e. two system matrices $P_1(s)$, $P_2(s)$ are realisations of the same transfer function matrix $G(s)$ if and only if there exist rational matrices $Q_1(s)$, $Q_2(s)$, $R_1(s)$, $R_2(s)$ such that

$$\begin{pmatrix} Q_1(s) & 0 \\ R_1(s) & I \end{pmatrix} \underbrace{\begin{pmatrix} T_2(s) & U_2(s) \\ -V_2(s) & W_2(s) \end{pmatrix}}_{P_2(s)} = \underbrace{\begin{pmatrix} T_1(s) & U_1(s) \\ -V_1(s) & W_1(s) \end{pmatrix}}_{P_1(s)} \begin{pmatrix} Q_2(s) & R_2(s) \\ 0 & I \end{pmatrix}$$

It has already been noted that Rosenbrock system matrices are contained in the set $\mathfrak{P}(m,\ell)$ necessary for the invariance of the finite and infinite zero structure under full equivalence, Definition 3.2. Thus the polynomial transformation of full equivalence between system matrices with the same transfer function $G(s)$ can be formulated as:

Definition 3.3: (*Full System Equivalence*)
Let $P_i(s)$ for $i = 1,2$ be two system matrices with transfer function matrix $G(s)$. Then $P_1(s)$, $P_2(s)$ are said to be *fully system equivalent* if there exist polynomial matrices $Q_1(s)$, $Q_2(s)$, $R_1(s)$, $R_2(s)$ such that

$$\underbrace{\begin{pmatrix} Q_1(s) & 0 \\ R_1(s) & I \end{pmatrix}}_{S_1(s)} \underbrace{\begin{pmatrix} T_2(s) & U_2(s) \\ -V_2(s) & W_2(s) \end{pmatrix}}_{P_2(s)} = \underbrace{\begin{pmatrix} T_1(s) & U_1(s) \\ -V_1(s) & W_1(s) \end{pmatrix}}_{P_1(s)} \underbrace{\begin{pmatrix} Q_2(s) & R_2(s) \\ 0 & I \end{pmatrix}}_{P_2(s)}$$

where the compound matrices

$$[S_1(s) \quad P_1(s)], \quad \begin{bmatrix} P_2(s) \\ -S_2(s) \end{bmatrix}$$

(i) have full normal rank

(ii) have no finite nor infinite zeros

(iii) satisfy the McMillan degree conditions

$$\delta([S_2(s) \quad P_2(s)]) = \delta(P_2(s)) \quad ; \qquad \delta\left(\begin{bmatrix} P_1(s) \\ -S_1(s) \end{bmatrix}\right) = \delta(P_1(s))$$

where $\delta(\cdot)$ denotes the McMillan Degree of $(\cdot)$. □

Thus by starting with the most basic form of equivalence for system matrices, namely input-output equivalence, further restrictions may be placed on the constituent matrices of the closed matrix form to preserve various properties of the system matrices. The following two sections, Sections 3.3 and 3.4, investigate transformations of many-indeterminate polynomial matrices and the properties of the system matrices that remain invariant. In particular the notion of least order is investigated using a generalisation of the relative prime definition for 1-D least order system matrices.

3.3 Equivalence of 2-D Polynomial Matrices

As noted above, a fundamental requirement for the equivalence of two system realisations is that they possess the same input-output behaviour, i.e. the two realisations have the same transfer function matrix. The Rosenbrock system matrix (RSM or system matrix) for 2-D systems can be defined in an analogous manner to 1-D systems:

$$P(z_1, z_2) = \begin{pmatrix} T(z_1, z_2) & U(z_1, z_2) \\ -V(z_1, z_2) & W(z_1, z_2) \end{pmatrix} \tag{127}$$

where $T_{r\times r}(z_1, z_2)$, $U_{r\times \ell}(z_1, z_2)$, $V_{m\times r}(z_1, z_2)$ and $W_{m\times \ell}(z_1, z_2)$ are polynomial matrices, arising from a transfer function matrix, $G(z_1, z_2)$, representation of

$$G(z_1, z_2) = V(z_1, z_2)T^{-1}(z_1, z_2)U(z_1, z_2) + W(z_1, z_2). \tag{128}$$

Thus the notion of equivalent input-output behaviour may be defined as:

Definition 3.4: (*Input-output Equivalence*)
Two system matrices $P_1(z_1, z_2)$, $P_2(z_1, z_2)$ with sizes $(r_1 + m) \times (r_1 + \ell)$ and $(r_2 + m) \times (r_2 + \ell)$ respectively are said to be input-output equivalent if

$$G_1(z_1, z_2) = G_2(z_1, z_2) \tag{129}$$

i.e. if their transfer function matrices, defined by (128), are equal. □

By analogy with 1-D systems the transformations of strict system equivalence and system equivalence may be defined. Now the transforming matrices are polynomial in two indeterminates. The initial point of investigation is the necessary and sufficient conditions for input-output equivalence. In 1-D system theory an exact characterisation is provided, in a closed matrix form, by Pugh [46]. This characterisation also holds for 2-D systems theory, as the following result demonstrates.

Theorem 3.4: *The system matrices*

$$P_i(z_1, z_2) = \begin{pmatrix} T_i(z_1, z_2) & U_i(z_1, z_2) \\ -V_i(z_1, z_2) & W_i(z_1, z_2) \end{pmatrix} \tag{130}$$

are realisations of the same transfer function matrix $G(z_1, z_2)$ if and only if there exist rational matrices $Q_1(z_1, z_2)$, $Q_2(z_1, z_2)$, $R_1(z_1, z_2)$, $R_2(z_1, z_2)$ such that

$$\begin{aligned} &\begin{pmatrix} Q_1(z_1, z_2) & 0 \\ R_1(z_1, z_2) & I_m \end{pmatrix} \begin{pmatrix} T_2(z_1, z_2) & U_2(z_1, z_2) \\ -V_2(z_1, z_2) & W_2(z_1, z_2) \end{pmatrix} \\ &= \begin{pmatrix} T_1(z_1, z_2) & U_1(z_1, z_2) \\ -V_1(z_1, z_2) & W_1(z_1, z_2) \end{pmatrix} \begin{pmatrix} Q_2(z_1, z_2) & R_2(z_1, z_2) \\ 0 & I_\ell \end{pmatrix} \end{aligned} \tag{131}$$

Proof: Suppose there exist rational matrices $Q_1(z_1, z_2)$, $Q_2(z_1, z_2)$, $R_1(z_1, z_2)$, $R_2(z_1, z_2)$ such that (131) holds. Note that the constituent equations of (131) are

$$Q_1 T_2 = T_1 Q_2 \tag{132}$$

$$Q_1 U_2 = T_1 R_2 + U_1 \tag{133}$$

$$R_1 T_2 - V_2 = -V_1 Q_2 \tag{134}$$

$$R_1 U_2 + W_2 = -V_1 R_2 + W_1 \tag{135}$$

Now consider the transfer function matrices associated with $P_1(z_1, z_2)$, $P_2(z_1, z_2)$

$$\begin{aligned} G_2(z_1, z_2) &= V_2 T_2^{-1} U_2 + W_2 \\ &= (R_1 T_2 + V_1 Q_2) T_2^{-1} U_2 + W_2 && \text{from (134)} \\ &= R_1 U_2 + W_2 + V_1 Q_2 T_2^{-1} U_2 \\ &= -V_1 R_2 + W_1 + V_1 Q_2 T_2^{-1} U_2 && \text{from (135)} \\ &= -V_1 R_2 + W_1 + V_1 T_1^{-1} Q_1 U_2 && \text{from (132)} \\ &= -V_1 R_2 + W_1 + V_1 T_1^{-1} (T_1 R_2 + U_1) && \text{from (133)} \\ &= W_1 + V_1 T_1^{-1} U_1 = G_1(z_1, z_2). \end{aligned}$$

Thus $P_1(z_1, z_2)$, $P_2(z_1, z_2)$ are input-output equivalent.

Conversely suppose that $P_1(z_1, z_2)$, $P_2(z_1, z_2)$ are input-output equivalent then,

$$\begin{aligned} & G_1(z_1, z_2) = G_2(z_1, z_2) \\ \Rightarrow \quad & V_1 T_1^{-1} U_1 + W_1 = V_2 T_2^{-1} U_2 + W_2. \end{aligned} \tag{136}$$

Hence for an arbitrary $r_2 \times r_1$ rational matrix $Q_1(z_1, z_2)$ the following is true

$$\begin{pmatrix} Q_1 & 0 & I_{r_2} & 0 \\ 0 & I_m & 0 & V_1 T_1^{-1} U_1 + W_1 \end{pmatrix} \begin{pmatrix} I_{r_1} & 0 \\ 0 & V_2 T_2^{-1} U_2 + W_2 \\ -Q_1 & 0 \\ 0 & -I_\ell \end{pmatrix} = 0. \tag{137}$$

Now the matrix

$$\begin{pmatrix} T_2^{-1} & 0 & 0 & 0 \\ V_2 T_2^{-1} & I_m & 0 & 0 \\ 0 & 0 & T_1 & U_1 \\ 0 & 0 & 0 & I_\ell \end{pmatrix} \tag{138}$$

is an invertible rational matrix with inverse

$$\begin{pmatrix} T_2 & 0 & 0 & 0 \\ -V_2 & I_m & 0 & 0 \\ 0 & 0 & T_1^{-1} & -T_1^{-1} U_1 \\ 0 & 0 & 0 & I_\ell \end{pmatrix}. \tag{139}$$

Thus post-multiplying the first matrix in (137) by (138), and pre-multiplying the second matrix in (137) by (139), (137) reduces to the form

$$\begin{pmatrix} Q_1T_2^{-1} & 0 & T_1 & U_1 \\ V_2T_2^{-1} & I_m & 0 & V_1T_1^{-1}U_1+W_1 \end{pmatrix} \begin{pmatrix} T_2 & 0 \\ -V_2 & V_2T_2^{-1}U_2+W_2 \\ -T_1^{-1}Q_1 & T_1^{-1}U_1 \\ 0 & -I_\ell \end{pmatrix} = 0. \quad (140)$$

Finally pre-multiplying (140) by

$$\begin{pmatrix} I & 0 \\ -V_1T_1^{-1} & I \end{pmatrix}$$

and post-multiplying (140) by

$$\begin{pmatrix} I & T_2^{-1}U_1 \\ 0 & I \end{pmatrix}$$

reduces (140) to the form

$$\begin{pmatrix} Q_1T_2^{-1} & 0 & T_1 & U_1 \\ V_1T_1^{-1}-V_1T_1^{-1}Q_1T_2^{-1} & I_m & -V_1 & W_1 \end{pmatrix} \begin{pmatrix} T_2 & U_2 \\ -V_2 & W_2 \\ -T_1^{-1}Q_1 & T_1^{-1}U_1-T_1^{-1}Q_1T_2^{-1}U_2 \\ 0 & -I \end{pmatrix} = 0$$

$$\text{i.e.} \quad \begin{pmatrix} Q_1T_2^{-1} & 0 \\ V_1T_1^{-1}-V_1T_1^{-1}Q_1T_2^{-1} & I_m \end{pmatrix} \begin{pmatrix} T_1 & U_1 \\ -V_1 & W_1 \end{pmatrix} = \begin{pmatrix} T_2 & U_2 \\ -V_2 & W_2 \end{pmatrix} \begin{pmatrix} T_1^{-1}Q_1 & T_1^{-1}Q_1T_1^{-1}U_2-T_1^{-1}U_1 \\ 0 & I \end{pmatrix} \quad (141)$$

which is of the form (131), as required. □

It should be noted that the transformation (131) is based on rational transforming matrices and describes the basic connection between two system matrices which give rise to the same transfer function matrix. The further conditions which should be imposed on these system matrices to ensure that the transformation (131) is based on polynomial matrices are not known, and in fact are not even known in the case of polynomial matrices in a single indeterminate. Whatever conditions are necessary to ensure a polynomial connection of the form (131) they are likely to be much more restrictive than their counterparts in single indeterminate polynomial matrix theory.

3.3.1 Least Order

Recall (Rosenbrock [12]) that in conventional multivariable systems theory a principal result is that two least order polynomial realisations of a given transfer function matrix are related by strict system equivalence (equation (131)), where least order is defined by the absence of input and output decoupling zeros or, equivalently, the pair $T(s)$, $U(s)$ are relatively (left) prime and the pair $T(s)$, $V(s)$ are relatively (right) prime.

In two dimensional systems theory the term "least order" might most reasonably be attributed to

$$P(z_1,z_2)=\begin{pmatrix} T(z_1,z_2) & U(z_1,z_2) \\ -V(z_1,z_2) & W(z_1,z_2) \end{pmatrix} \tag{142}$$

where $T(z_1,z_2)$ and $U(z_1,z_2)$ are minor (left) coprime and $T(z_1,z_2)$ and $V(z_1,z_2)$ are minor (right) coprime but as will be seen this turns out to be unsatisfactory in the definition of least order (in the sense of having similar properties to 1-D least order, namely a necessary and sufficient condition for a polynomial equivalence relation to exist between two system matrices of the same system). Later an alternative definition for least order will be introduced; thus to distinguish the two types of least order the type defined by (142) and the minor coprime conditions on the pairs $T(z_1,z_2)$, $U(z_1,z_2)$ and $T(z_1,z_2)$, $V(z_1,z_2)$ will be called *(minor) least order.* The term least order will be reserved for later use. Clearly it would be inappropriate to define least order using zero coprimeness as this would exclude a large class of system matrices that contain non-essential singularities of the second kind. For example $G(z_1,z_2)=z_1/z_2$ always possesses the non-essential singularity of the second kind $(0,0)$ and any system matrix realisation of $G(z_1,z_2)$ will need to reflect this. However, all systems possess a "(minor) least order" realisation: consider a system matrix realisation of the transfer function matrix $G(z_1,z_2)$

$$P(z_1,z_2)=\begin{pmatrix} T(z_1,z_2) & U(z_1,z_2) \\ -V(z_1,z_2) & W(z_1,z_2) \end{pmatrix} \tag{143}$$

and compute matrices $Q_1(z_1,z_2)$ and $Q_2(z_1,z_2)$, using the greatest common divisor algorithm of Part Two, such that

$$P(z_1,z_2)=\begin{pmatrix} Q_1(z_1,z_2) & 0 \\ 0 & I \end{pmatrix}\begin{pmatrix} \bar{T}(z_1,z_2) & \bar{U}(z_1,z_2) \\ -\bar{V}(z_1,z_2) & \bar{W}(z_1,z_2) \end{pmatrix}\begin{pmatrix} Q_1(z_1,z_2) & 0 \\ 0 & I \end{pmatrix} \tag{144}$$

where $\bar{T}(z_1,z_2)$, $\bar{U}(z_1,z_2)$ are minor (left) coprime and $\bar{T}(z_1,z_2)$, $\bar{V}(z_1,z_2)$ are minor (right) coprime; thus defining a (minor) least order system matrix. Consider then the following example.

Example 3.2: The system matrices

$$\left.\begin{aligned} P_1(z_1,z_2) &= \begin{pmatrix} z_2^2 & z_1 \\ -(z_1-1) & 0 \end{pmatrix} \\ P_2(z_1,z_2) &= \begin{pmatrix} z_2^2 & (z_1-1) \\ -z_1 & 0 \end{pmatrix} \end{aligned}\right\} \tag{145}$$

are both (minor) least order system matrices corresponding to the transfer function

$$G(z_1,z_2) = \frac{z_1(z_1-1)}{z_2^2}. \tag{146}$$

By Theorem 3.4, $P_1(z_1,z_2)$ and $P_2(z_1,z_2)$ will be related as in (131) through rational transforming matrices. Thus

$$\begin{pmatrix} q_1(z_1,z_2) & 0 \\ r_1(z_1,z_2) & 1 \end{pmatrix} \underbrace{\begin{pmatrix} z_2^2 & z_1 \\ -(z_1-1) & 0 \end{pmatrix}}_{P_1(z_1,z_2)} = \underbrace{\begin{pmatrix} z_2^2 & (z_1-1) \\ -z_1 & 0 \end{pmatrix}}_{P_2(z_1,z_2)} \begin{pmatrix} q_2(z_1,z_2) & r_2(z_1,z_2) \\ 0 & 1 \end{pmatrix} \tag{147}$$

for rational functions $q_i(z_1,z_2)$ and $r_i(z_1,z_2)$ for $i = 1,2$. Now the equation (1,2) of (147) is

$$q_1(z_1,z_2)z_1 - r_2(z_1,z_2)z_2^2 = z_1 - 1. \tag{148}$$

It follows from Hilbert Nullstellensatz, Lemma 1.2, that (148) does not have a polynomial solution for $q_1(z_1,z_2)$ and $r_1(z_1,z_2)$. Thus a polynomial connection does not exist between $P_1(z_1,z_2)$ and $P_2(z_1,z_2)$ of the form (147).

Interchanging the roles of $P_1(z_1,z_2)$ and $P_2(z_1,z_2)$ in (147), the transformation (131) becomes

$$\begin{pmatrix} q_1(z_1,z_2) & 0 \\ r_1(z_1,z_2) & 1 \end{pmatrix} \underbrace{\begin{pmatrix} z_2^2 & (z_1-1) \\ -z_1 & 0 \end{pmatrix}}_{P_2(z_1,z_2)} = \underbrace{\begin{pmatrix} z_2^2 & z_1 \\ -(z_1-1) & 0 \end{pmatrix}}_{P_1(z_1,z_2)} \begin{pmatrix} q_2(z_1,z_2) & r_2(z_1,z_2) \\ 0 & 1 \end{pmatrix}$$

and the (1,2) equation

$$q_1(z_1-1) = z_2^2 + z_1$$

demonstrates as before that there does not exist a polynomial transformation of the type (131).

Thus although $P_1(z_1,z_2)$ and $P_2(z_1,z_2)$ are both (minor) least order realisations of the same transfer function matrix, there is no polynomial based transformation connecting them. In particular, there does not exist constants q_1, q_2 such that a polynomial transformation of the form (131) required for strict system equivalence. □

It follows from the above example that the classical 1-D multivariable result, that least order (in the 1-D sense) realisations are strictly system equivalent, will not generalise

to two dimensional (minor) least order systems. Moreover, there does not necessarily even exist a polynomial transformation of the form (131). However, it is possible to establish the nature of this polynomial transformation when it does exist.

Theorem 3.5: *If two (minor) least order system matrices $P_i(z_1, z_2)$ for $i = 1, 2$ are related by a polynomial form of the relationship*

$$\begin{pmatrix} Q_1(z_1,z_2) & 0 \\ R_1(z_1,z_2) & I_m \end{pmatrix} \begin{pmatrix} T_2(z_1,z_2) & U_2(z_1,z_2) \\ -V_2(z_1,z_2) & W_2(z_1,z_2) \end{pmatrix} = \begin{pmatrix} T_1(z_1,z_2) & U_1(z_1,z_2) \\ -V_1(z_1,z_2) & W_1(z_1,z_2) \end{pmatrix} \begin{pmatrix} Q_2(z_1,z_2) & R_2(z_1,z_2) \\ 0 & I_\ell \end{pmatrix} \tag{149}$$

then $Q_1(z_1, z_2)$, $T_1(z_1, z_2)$ are minor (left) coprime and $Q_2(z_1, z_2)$, $T_2(z_1, z_2)$ are minor (right) coprime.

Proof: The relationship (149) may be written in the form

$$\begin{pmatrix} Q_1 & 0 & T_1 & U_1 \\ R_1 & I & -V_1 & W_1 \end{pmatrix} \begin{pmatrix} T_2 & U_2 \\ -V_2 & W_2 \\ -Q_2 & -R_2 \\ 0 & -I \end{pmatrix} = 0. \tag{150}$$

The (1,2) block equation yields

$$U_1 = Q_1 U_2 - T_1 R_2 \tag{151}$$

in which T_1, U_1 are minor (left) coprime. Therefore by Bézout identities, Theorem 1.4, there exist $\bar{X}_i$, $\bar{Y}_i$ for $i = 1, 2$ such that

$$T_1 \bar{X}_i + U_1 \bar{Y}_i = \psi_i(z_i) I \quad \text{for} \quad i = 1, 2$$

for scalar polynomials $\psi_i(z_i)$. This Bézout identity and equation (151) yield the equation

$$T_1(X_i - R_2 Y_i) + Q_1(U_2 Y_i) = \psi_i(z_i) I \quad \text{for} \quad i = 1, 2$$

demonstrating that T_1, Q_1 are minor (left) coprime.
The (2,1) block equation of (150) yields

$$V_2 = R_1 T_2 + V_1 Q_2. \tag{152}$$

Now V_2, T_2 are minor (right) coprime and so by the Bézout identities, Theorem 1.4, there exist polynomial matrices $X_i(z_1, z_2)$, $Y_i(z_1, z_2)$ for $i = 1, 2$ such that

$$X_i T_2 + Y_i V_2 = \phi_i(z_i) I \quad \text{for} \quad i = 1, 2. \tag{153}$$

Pre-multiplying (152) by Y_i and substituting from (153) for $Y_i V_2$ gives

$$\phi_i(z_i) I - X_i T_2 = Y_i R_1 T_2 + Y_i V_1 Q_2$$

$$\Rightarrow \quad (Y_i R_1 + X_i)T_2 + Y_i V_1 Q_2 = \phi_i(z_i)I \quad \text{for} \quad i = 1, 2.$$

Thus T_2, Q_2 are minor (right) coprime. □

Note: Some conditions of (minor) least order are redundant in the proof of Theorem 3.5. Namely, the minor (right) coprime condition on T_1, V_1 and the minor (left) coprime condition on T_2, U_2.

Consider now an alternative form of "least order" by imposing a stronger form of coprimeness on the T, V part of the system matrix. However, this form will be seen to be sufficiently general that every transfer function matrix can be realised in such a way.

Definition 3.5: (*Least Order*)
A two dimensional polynomial system matrix $P(z_1, z_2)$ of the form (142) will be said to be *least order* if $T(z_1, z_2)$ and $U(z_1, z_2)$ are minor (left) coprime and $T(z_1, z_2)$ and $V(z_1, z_2)$ are zero (right) coprime. □

The definition expresses an intention to realise the non-essential singularities of the second kind of the transfer function matrix so that they always occur in the $[T(z_1, z_2)\ U(z_1, z_2)]$ part of the system matrix. Note that $G(z_1, z_2)$ always possesses least order realisations of this type since any left (necessarily minor) coprime MFD gives rise to a system matrix which is least order in the above sense. For example suppose that $G(z_1, z_2)$ possess a minor (left) coprime MFD defined by $G(z_1, z_2) = D^{-1}(z_1, z_2)N(z_1, z_2)$ then $G(z_1, z_2)$ may be realised as the system matrix given by

$$G(z_1, z_2) = \begin{pmatrix} D(z_1, z_2) & N(z_1, z_2) \\ I & 0 \end{pmatrix} \tag{154}$$

thus the pair $D(z_1, z_2)$, $N(z_1, z_2)$ are minor (left) coprime by construction and the pair $D(z_1, z_2)$, I are necessarily zero coprime ($\det I = 1$).

There is, of course, no particular necessity to handle the non-essential singularities of the second kind in this manner; they could equally be taken to be realised in the $[T(z_1, z_2)^{\mathrm{T}} \ -V(z_1, z_2)^{\mathrm{T}}]^{\mathrm{T}}$ part of the system matrix. The important point at issue, seems to be that it is necessary to agree at the outset of the investigation exactly how these singularities of $G(z_1, z_2)$ are to be realised within the system matrix and Definition 3.5 takes one such view.

Despite this, it has still not proved possible to establish that any two least order realisations of $G(z_1, z_2)$ are related by a polynomial form of the relationship (141). However, it is possible to establish the nature of this polynomial connection when it exists.

Theorem 3.6: *If two least order system matrices $P_i(z_1, z_2)$ for $i = 1, 2$ are related by a polynomial form of the relationship*

$$\begin{pmatrix} Q_1(z_1,z_2) & 0 \\ R_1(z_1,z_2) & I_m \end{pmatrix} \begin{pmatrix} T_2(z_1,z_2) & U_2(z_1,z_2) \\ -V_2(z_1,z_2) & W_2(z_1,z_2) \end{pmatrix}$$
$$= \begin{pmatrix} T_1(z_1,z_2) & U_1(z_1,z_2) \\ -V_1(z_1,z_2) & W_1(z_1,z_2) \end{pmatrix} \begin{pmatrix} Q_2(z_1,z_2) & R_2(z_1,z_2) \\ 0 & I_\ell \end{pmatrix} \tag{155}$$

then $Q_1(z_1, z_2)$, $T_1(z_1, z_2)$ are zero (left) coprime and $Q_2(z_1, z_2)$, $T_2(z_1, z_2)$ are zero (right) coprime.

Proof: The relationship (155) may be written in the form

$$\begin{pmatrix} Q_1 & 0 & T_1 & U_1 \\ R_1 & I & -V_1 & W_1 \end{pmatrix} \begin{pmatrix} T_2 & U_2 \\ -V_2 & W_2 \\ -Q_2 & -R_2 \\ 0 & -I \end{pmatrix} = 0. \tag{156}$$

The (1,2) block equation yields

$$U_1 = Q_1 U_2 - T_1 R_2 \tag{157}$$

in which T_1, U_1 are minor (left) coprime. Therefore by Bézout identities, Theorem 1.4, there exist $\bar{X}_i$, $\bar{Y}_i$ for $i = 1, 2$ such that

$$T_1 \bar{X}_i + U_1 \bar{Y}_i = \psi_i(z_i) I \quad \text{for} \quad i = 1, 2$$

for scalar polynomials $\psi_i(z_i)$. This Bézout identity and equation (157) yield the equation

$$T_1(\bar{X}_i - R_2 \bar{Y}_i) + Q_1(U_2 \bar{Y}_i) = \psi_i(z_i) I \quad \text{for} \quad i = 1, 2$$

demonstrating the T_1, Q_1 are minor (left) coprime.
The (2,1) block equation yields

$$V_2 = R_1 T_2 + V_1 Q_2. \tag{158}$$

Now V_2, T_2 are zero (right) coprime and so by the Bézout identity there exist polynomial matrices $X(z_1, z_2)$, $Y(z_1, z_2)$ such that

$$XT_2 + YV_2 = I. \tag{159}$$

Pre-multiplying (158) by Y and substituting from (159) for YV_2 gives

$$I - XT_2 = YR_1T_2 + YV_1Q_2$$

$$\Rightarrow \quad (YR_1 + X)T_2 + YV_1Q_2 = I.$$

Thus T_2, Q_2 are zero (right) coprime.
Now from the (1,1) equation of (156)

$$Q_1 T_2 = T_1 Q_2$$

and so

$$T_1^{-1} Q_1 = Q_2 T_2^{-1} \tag{160}$$

are coprime MFDs. By Theorem 2.2 the fractions in (160) must be of the same coprimeness type and since T_2, Q_2 are zero (right) coprime, it follows that T_1, Q_1 are zero (left) coprime. □

The following theorem relates some coprimeness interconnections between the constituent matrices of (155).

Theorem 3.7: *If two system matrices $P_i(z_1, z_2)$ for $i = 1, 2$ are related by a polynomial form of the relationship*

$$\begin{pmatrix} Q_1(z_1,z_2) & 0 \\ R_1(z_1,z_2) & I_m \end{pmatrix} \begin{pmatrix} T_2(z_1,z_2) & U_2(z_1,z_2) \\ -V_2(z_1,z_2) & W_2(z_1,z_2) \end{pmatrix}$$
$$= \begin{pmatrix} T_1(z_1,z_2) & U_1(z_1,z_2) \\ -V_1(z_1,z_2) & W_1(z_1,z_2) \end{pmatrix} \begin{pmatrix} Q_2(z_1,z_2) & R_2(z_1,z_2) \\ 0 & I_\ell \end{pmatrix} \tag{161}$$

then

(i) $Q_1(z_1, z_2)$, $T_1(z_1, z_2)$ *are minor (left) coprime if* $T_1(z_1, z_2)$, $U_1(z_1, z_2)$ *are minor (left) coprime;*

(ii) $Q_1(z_1, z_2)$, $T_1(z_1, z_2)$ *are zero (left) coprime if* $T_1(z_1, z_2)$, $U_1(z_1, z_2)$ *are zero (left) coprime;*

(iii) $Q_2(z_1, z_2)$, $T_2(z_1, z_2)$ *are minor (right) coprime if* $T_2(z_1, z_2)$, $V_2(z_1, z_2)$ *are minor (right) coprime;*

(iv) $Q_2(z_1, z_2)$, $T_2(z_1, z_2)$ *are zero (right) coprime if* $T_2(z_1, z_2)$, $V_2(z_1, z_2)$ *are zero (right) coprime.*

Proof: (i) Suppose that T_1, U_1 are minor (left) coprime. Then by the Bézout identities there exist polynomial matrices $X_i(z_1, z_2)$, $Y_i(z_1, z_2)$ and scalar polynomials $\psi_i(z_i)$ for $i = 1, 2$ such that

$$T_1 X_i + U_1 Y_i = \psi_i I \quad \text{for} \quad i = 1, 2$$

and using the (1,2) equation of (161)

$$T_1(X_i - T_1 R_2 Y_i) + Q_1(U_2 Y_i) = \psi_i I \quad \text{for} \quad i = 1, 2.$$

Thus Q_1, T_1 are minor (left) coprime.

(ii) The proof follows analogously to (i) by considering $\psi_i(z_i) \equiv 1$.

(iii) T_2, V_2 are minor (right) coprime. Then by the Bézout identities there exist polynomial matrices $W_i(z_1, z_2)$, $Y_i(z_1, z_2)$ and scalar polynomials $\phi_i(z_i)$ for $i = 1, 2$ such that

$$W_i T_2 + Z_i V_2 = \phi_i I$$

and using the (2,1) equation of (161)

$$(W_i + Z_i R_1) T_2 + (Z_i V_1) Q_2 = \phi_i I \quad \text{for} \quad i = 1, 2.$$

Thus the result is established.

(iv) The proof follows analogously to (iii) by considering $\phi_i(z_i) \equiv 1$. □

Polynomial transformations of the type (161) are considered more fully in the following section.

3.3.2 Polynomial Transformations

In the previous section it was established that any two system matrix realisations of the same system are related by a closed form matrix relationship (131) in which the transforming matrices are, in general, rational. This was seen to be a necessary and sufficient condition for the system matrices to be input-output equivalent. Moreover if this relationship was polynomial and the system matrices were least order, in the sense of Definition 3.5, the T-blocks were shown to possess a zero coprimeness condition with the Q-blocks.

In this section the above form of relationship will be further developed for system matrices that are not necessarily least order. In fact it will be shown that under certain coprimeness conditions the polynomial form of the relationship (155) is an equivalence relation. Before undertaking this task some new terminology needs to be introduced.

Definition 3.6: (*Zero Equivalence*)
Let $T_1(z_1, z_2), T_2(z_1, z_2)$ be two polynomial matrices. If an equation of the form

$$Q_1(z_1, z_2) T_2(z_1, z_2) = T_1(z_1, z_2) Q_2(z_1, z_2) \tag{162}$$

exists where $T_1(z_1, z_2)$, $Q_1(z_1, z_2)$ are zero (left) coprime and $T_2(z_1, z_2)$, $Q_2(z_1, z_2)$ are zero (right) coprime then $T_2(z_1, z_2)$ is said to be *zero equivalent* to $T_1(z_1, z_2)$.

Furthermore if two polynomial system matrices $P_1(z_1, z_2)$, $P_2(z_1, z_2)$ are related by an equation of the form

$$\overbrace{\begin{pmatrix} Q_1(z_1,z_2) & 0 \\ R_1(z_1,z_2) & I_m \end{pmatrix}}^{S_1(z_1,z_2)} \overbrace{\begin{pmatrix} T_2(z_1,z_2) & U_2(z_1,z_2) \\ -V_2(z_1,z_2) & W_2(z_1,z_2) \end{pmatrix}}^{P_2(z_1,z_2)} = \underbrace{\begin{pmatrix} T_1(z_1,z_2) & U_1(z_1,z_2) \\ -V_1(z_1,z_2) & W_1(z_1,z_2) \end{pmatrix}}_{P_1(z_1,z_2)} \underbrace{\begin{pmatrix} Q_2(z_1,z_2) & R_2(z_1,z_2) \\ 0 & I_\ell \end{pmatrix}}_{S_2(z_1,z_2)} \tag{163}$$

where $S_1(z_1, z_2)$, $P_1(z_1, z_2)$ are zero (left) coprime and $S_2(z_1, z_2)$, $P_2(z_1, z_2)$ are zero (right) coprime then $P_2(z_1, z_2)$ is said to be *zero system equivalent* to $P_1(z_1, z_2)$. □

Definition 3.7: (*Minor Equivalence*)
Let $T_1(z_1, z_2), T_2(z_1, z_2)$ be two polynomial matrices. If an equation of the form

$$Q_1(z_1, z_2)T_2(z_1, z_2) = T_1(z_1, z_2)Q_2(z_1, z_2) \tag{164}$$

exists where $T_1(z_1, z_2)$, $Q_1(z_1, z_2)$ are minor (left) coprime and $T_2(z_1, z_2)$, $Q_2(z_1, z_2)$ are minor (right) coprime then $T_2(z_1, z_2)$ is said to be *minor equivalent* to $T_1(z_1, z_2)$. Furthermore if two polynomial system matrices $P_1(z_1, z_2)$, $P_2(z_1, z_2)$ are related by an equation of the form

$$\overbrace{\begin{pmatrix} Q_1(z_1,z_2) & 0 \\ R_1(z_1,z_2) & I_m \end{pmatrix}}^{S_1(z_1,z_2)} \overbrace{\begin{pmatrix} T_2(z_1,z_2) & U_2(z_1,z_2) \\ -V_2(z_1,z_2) & W_2(z_1,z_2) \end{pmatrix}}^{P_2(z_1,z_2)} = \underbrace{\begin{pmatrix} T_1(z_1,z_2) & U_1(z_1,z_2) \\ -V_1(z_1,z_2) & W_1(z_1,z_2) \end{pmatrix}}_{P_1(z_1,z_2)} \underbrace{\begin{pmatrix} Q_2(z_1,z_2) & R_2(z_1,z_2) \\ 0 & I_\ell \end{pmatrix}}_{S_2(z_1,z_2)} \tag{165}$$

where $S_1(z_1, z_2)$, $P_1(z_1, z_2)$ are minor (left) coprime and $S_2(z_1, z_2)$, $P_2(z_1, z_2)$ are minor (right) coprime then $P_2(z_1, z_2)$ is said to be *minor system equivalent* to $P_1(z_1, z_2)$. □

The first result is an interesting property relating system equivalence and the equivalence between the T-blocks of the system matrices.

Theorem 3.8: *Let the two system matrices $P_1(z_1, z_2), P_2(z_1, z_2)$ be related by a transformation of the form*

$$S_1(z_1, z_2)P_2(z_1, z_2) = P_1(z_1, z_2)S_2(z_1, z_2) \tag{166}$$

where

$$S_1(z_1, z_2) = \begin{pmatrix} Q_1(z_1,z_2) & 0 \\ R_1(z_1,z_2) & I_m \end{pmatrix} \quad S_2 = \begin{pmatrix} Q_2(z_1,z_2) & R_2(z_1,z_2) \\ 0 & I_\ell \end{pmatrix}.$$

Then

(a) *$P_1(z_1,z_2)$, $S_1(z_1,z_2)$ are minor (left) coprime if and only if $T_1(z_1,z_2)$, $Q_1(z_1,z_2)$ are minor (left) coprime.*

(b) *$P_1(z_1,z_2)$, $S_1(z_1,z_2)$ are zero (left) coprime if and only if $T_1(z_1,z_2)$, $Q_1(z_1,z_2)$ are zero (left) coprime.*

Similar statements may be formulated for the pairs $P_2(z_1,z_2)$, $S_2(z_1,z_2)$ and $T_2(z_1,z_2)$, $Q_2(z_1,z_2)$.

Proof:

(a) Firstly assume that $Q_1(z_1,z_2)$ and $T_1(z_1,z_2)$ are minor (left) coprime. Thus by the Bézout identities there exist polynomial matrices $X(z_1,z_2)$, $Y(z_1,z_2)$ such that

$$Q_1(z_1,z_2)X(z_1,z_2)+T_1(z_1,z_2)Y(z_1,z_2)=\phi_i(z_i)I_{r_1} \quad \text{for} \quad i=1,2 \tag{167}$$

where $\phi_i(z_i)$ is a polynomial in z_i. Consider the matrix equation

$$\begin{pmatrix} Q_1 & 0 \\ R_1 & I_m \end{pmatrix}\begin{pmatrix} X & 0 \\ V_1Y-R_1X & \phi_iI_m \end{pmatrix}+\begin{pmatrix} T_1 & U_1 \\ -V_1 & W_1 \end{pmatrix}\begin{pmatrix} Y & 0 \\ 0 & 0 \end{pmatrix}=\phi_iI_{r_1+m} \tag{168}$$

for $i=1,2$ which is a Bézout identity for minor (left) coprimeness and guarantees that S_1 and P_1 are minor (left) coprime. Hence the "if" part is proved.

Conversely, assume that S_1 and P_1 are minor (left) coprime. Thus there exist polynomial matrices $X_{ij}(z_1,z_2)$, $Y_{ij}(z_1,z_2)$ for $i,j=1,2$ such that

$$\begin{pmatrix} Q_1 & 0 \\ R_1 & I_m \end{pmatrix}\begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix}+\begin{pmatrix} T_1 & U_1 \\ -V_1 & W_1 \end{pmatrix}\begin{pmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{pmatrix}=\phi_iI_{r_1+m} \tag{169}$$

for $i=1,2$. Now the (1,1) equation gives

$$Q_1X_{11}+T_1Y_{11}+U_1Y_{21}=\phi_iI_{r_1} \tag{170}$$

and from the (1,2) equation of (165) $Q_1U_2=T_1R_2+U_1$ which gives

$$Q_1X_{11}+T_1Y_{11}+(Q_1U_2-T_1R_2)Y_{21}=\phi_iI_{r_1} \tag{171}$$

$$Q_1(X_{11}+U_2Y_{21})+T_1(Y_{11}-R_2Y_{21})=\phi_iI_{r_1}. \tag{172}$$

Thus $Q_1(z_1,z_2)$ and $T_1(z_1,z_2)$ are minor (left) coprime. Hence the "only if" part is proved.

(b) By setting $\phi_i\equiv 1$ for $i=1,2$ in equations (167)-(172) the result is proved. □

Thus the equivalence of the system matrix transformation induces the same equivalence transformation on the T-blocks of the system matrix and vice versa.

Theorem 3.9: *The relation of zero equivalence (162) is an equivalence relation.*

Proof: It is required to prove that the relation (162) is reflexive, transitive and symmetric.

Let $T_1(z_1, z_2)$ and $T_2(z_1, z_2)$ be two polynomial matrices, respectively of sizes $p \times q$ and $r \times s$ with $p - q = r - s$. Let $Q_1(z_1, z_2)$ and $Q_2(z_1, z_2)$ be two polynomial matrices such that

$$Q_2 T_1 = T_2 Q_1 \tag{173}$$

with Q_1, T_1 zero (right) coprime and Q_2, T_2 zero (left) coprime.

1. *Reflexivity*: Let $T_1 \equiv T_2$ in (173). Then $p = r$ and $q = s$. If $Q_1 = I_q$ and $Q_2 = I_r$ then

   $$I_r T_1 = T_2 I_\ell$$

   and since $\det I = 1$ T_1, I_r are zero (left) coprime and T_2, I_q are zero (right) coprime reflexivity is proved.

2. *Transitivity*: Suppose that

   $$Q_2 T_1 = T_2 Q_1 \tag{174}$$

   $$\bar{Q}_2 T_2 = T_3 \bar{Q}_1 \tag{175}$$

   with the usual coprime conditions. From (174)

   $$\begin{aligned} \bar{Q}_2(Q_2 T_1) &= \bar{Q}_2(T_2 Q_1) \\ &= T_3 \bar{Q}_1 Q_1. \end{aligned}$$

   Thus it is required to prove that $\bar{Q}_2 Q_2$, T_3 are zero (left) coprime and that $\bar{Q}_1 Q_1$, T_1 are zero (right) coprime. From the coprime conditions and the Bézout identities there exist X_1, X_2, Y_1 and Y_2 such that

   $$Q_2 X_1 + T_2 Y_1 = I \tag{176}$$

   $$\bar{Q}_2 X_2 + T_3 Y_2 = I. \tag{177}$$

   Pre-multiplying (176) by $\bar{Q}_2$ and using (175) gives

   $$\begin{aligned} \bar{Q}_2 Q_2 X_1 + \bar{Q}_2 T_2 Y_1 &= \bar{Q}_2 \\ \bar{Q}_2 Q_2 X_1 + T_3 \bar{Q}_1 Y_1 &= \bar{Q}_2. \end{aligned}$$

   Now post-multiply by X_2 and use (177) to give

   $$\bar{Q}_2 Q_2 X_1 X_2 + T_3 \bar{Q}_1 Y_1 X_2 = I - T_3 Y_2$$

   and rearrange to give

   $$\bar{Q}_2 Q_2 (X_1 X_2) + T_3(\bar{Q}_1 Y_1 X_2 + Y_2) = I.$$

Thus $\bar{Q}_2 Q_2$ and T_3 are zero (left) coprime. In the same way it can be proved that $\bar{Q}_1 Q_1$ and T_2 are zero (right) coprime.

3. *Symmetry*: Write the coprimeness conditions as

$$(\bar{Y}_1 \quad \bar{X}_1)\begin{pmatrix} T_1 \\ -Q_1 \end{pmatrix} = I_q. \tag{178}$$

$$(Q_2 \quad T_2)\begin{pmatrix} X_2 \\ Y_2 \end{pmatrix} = I_r. \tag{179}$$

Also (173) may be written as

$$(Q_2 \quad T_2)\begin{pmatrix} T_1 \\ -Q_1 \end{pmatrix} = 0. \tag{180}$$

Then

$$\begin{pmatrix} Q_2 & T_2 \\ \bar{Y}_1 & \bar{X}_1 \end{pmatrix}\begin{pmatrix} X_2 & T_1 \\ Y_2 & -Q_1 \end{pmatrix} = \begin{pmatrix} I_r & 0 \\ J & I_\ell \end{pmatrix} \tag{181}$$

where $J = \bar{Y}_1 X_2 + \bar{X}_1 Y_2$. Now pre-multiply (181) by $\begin{pmatrix} I_r & 0 \\ -J & I_\ell \end{pmatrix}$ so that

$$\underbrace{\begin{pmatrix} Q_2 & T_2 \\ Y_1 & X_1 \end{pmatrix}}_{A}\underbrace{\begin{pmatrix} X_2 & T_1 \\ Y_2 & -Q_1 \end{pmatrix}}_{B} = \begin{pmatrix} I_r & 0 \\ 0 & I_\ell \end{pmatrix} \tag{182}$$

where $X_1 = \bar{X}_1 - JT_2$ and $Y_1 = \bar{Y}_1 - JQ_2$. Since $A_{(r+q)\times(p+s)}$ and $B_{(p+s)\times(r+q)}$ are square, the same size and are polynomial inverses of each other they must be unimodular and so commute, to give

$$\underbrace{\begin{pmatrix} X_2 & T_1 \\ Y_2 & -Q_1 \end{pmatrix}}_{B}\underbrace{\begin{pmatrix} Q_2 & T_2 \\ Y_1 & X_1 \end{pmatrix}}_{A} = \begin{pmatrix} I_m & 0 \\ 0 & I_s \end{pmatrix}. \tag{183}$$

The constituent equations are

$$X_2 Q_2 + T_1 Y_1 = I_m \tag{184}$$

$$X_2 T_2 + T_1 X_1 = 0 \tag{185}$$

$$Y_2 Q_2 + (-Q_1) Y_1 = 0 \tag{186}$$

$$Y_2 T_2 + (-Q_1) X_1 = I_s. \tag{187}$$

Thus from (184) X_2, T_1 are zero (left) coprime and from (187) $-X_1$, T_2 are zero (right) coprime. Also from (185)

$$X_2 T_2 = T_1(-X_1)$$

thus the relation is symmetrical.

Hence the relation is reflexive, symmetric and transitive, i.e. an equivalence relation, and the theorem is proved. □

It has been proved in Theorem 3.8 that the coprimeness on the T-blocks is synonymous with the coprimeness on the system matrices. Thus, in light of the previous theorem, if two system matrices are zero system equivalent the T-blocks are related by a true equivalence relation, the natural extension to consider is whether zero system equivalence also defines a true equivalence relation. The answer is provided by the following theorem. Notice that the previous theorem guarantees the coprimeness conditions for each of the three relations (reflexivity, transitivity and symmetry). In addition the specific structure of the transforming matrices of zero system equivalence must be preserved.

Theorem 3.10: *Zero system equivalence is an equivalence relation.*

Proof: Let $P_1(z_1, z_2)$ and $P_2(z_1, z_2)$ be two polynomial system matrices of sizes $(r_i + m) \times (r_i + \ell)$ for $i = 1, 2$. Also let $S_1(z_1, z_2)$ and $S_2(z_1, z_2)$ be polynomial matrices of the form given above so that

$$S_1(z_1, z_2)P_2(z_1, z_2) = P_1(z_1, z_2)S_2(z_1, z_2) \tag{188}$$

where $S_1(z_1, z_2)$, $P_1(z_1, z_2)$ are zero (left) coprime and $S_2(z_1, z_2)$, $P_2(z_1, z_2)$ are zero (right) coprime.

1. Reflexivity: Let $P_1 \equiv P_2$ and let $Q_1 = I_{r_1}$, $Q_2 = I_{r_2 l}$, $R_1 = 0$, $R_2 = 0$ then $P_1 = P_2$ and the coprime conditions hold since $\det I = 1$.
2. Transitivity: Suppose that P_2 is zero system equivalent to P_1 and that P_1 is zero system equivalent to P_3, i.e.

$$S_1 P_2 = P_1 S_2 \tag{189}$$

$$\bar{S}_1 P_1 = P_3 \bar{S}_2 \tag{190}$$

with the usual coprime conditions. Then from Theorem 3.9 it is known that P_3 and P_2 are connected by an equation of the form

$$\bar{S}_1 S_1 P_2 = P_3 \bar{S}_2 S_2$$

with the zero equivalence coprime conditions. To show that the relation, in systems terms, is also transitive the transforming matrices, $\bar{S}_1 S_1$ and $\bar{S}_2 S_2$, must have the correct structure. Let

$$\bar{S}_1 = \begin{pmatrix} \bar{Q}_1 & 0 \\ \bar{R}_1 & I_m \end{pmatrix} \quad \text{and} \quad \bar{S}_2 = \begin{pmatrix} \bar{Q}_2 & \bar{R}_2 \\ 0 & I_\ell \end{pmatrix}$$

then

$$\bar{S}_1 S_1 = \begin{pmatrix} \bar{Q}_1 Q_1 & 0 \\ \bar{R}_1 Q_1 + R_1 & I_m \end{pmatrix} \quad \text{and} \quad \bar{S}_2 S_2 = \begin{pmatrix} \bar{Q}_2 Q_2 & \bar{Q}_2 R_2 + \bar{R}_2 \\ 0 & I_\ell \end{pmatrix}.$$

Thus the relation is transitive.

3. Symmetry: Write equation (188) in the form

$$\underbrace{\begin{pmatrix} Q_1 & 0 \\ R_1 & I_m \end{pmatrix}}_{S_1} \underbrace{\begin{pmatrix} T_2 & U_2 \\ -V_2 & W_2 \end{pmatrix}}_{P_2} = \underbrace{\begin{pmatrix} T_1 & U_1 \\ -V_1 & W_1 \end{pmatrix}}_{P_1} \underbrace{\begin{pmatrix} Q_2 & R_2 \\ 0 & I_m \end{pmatrix}}_{S_2} \tag{191}$$

and the constituent equations are

$$Q_1 T_2 = T_1 Q_2 \tag{192}$$

$$Q_1 U_2 = T_1 R_2 + U_1 \tag{193}$$

$$R_1 T_2 - V_2 = -V_1 Q_2 \tag{194}$$

$$R_1 U_2 + W_2 = -V_1 R_2 + W_1. \tag{195}$$

Now since S_1, P_1 are zero (left) coprime then Q_1, T_1 are also zero (left) coprime, similarly zero (right) coprime conditions for the pairs S_2, P_2 and Q_2, T_2. Thus there exist X_1, X_2, Y_1 and Y_2 such that

$$Q_1 X_1 + T_1 Y_1 = I_{r_1} \tag{196}$$

$$X_2 Q_2 + Y_2 T_2 = I_{r_2} \tag{197}$$

and from these two Bézout identities the Bézout identities for the system matrices may be deduced

$$\underbrace{\begin{pmatrix} Q_1 & 0 \\ R_1 & I_m \end{pmatrix}}_{S_1} \underbrace{\begin{pmatrix} X_1 & 0 \\ V_1 Y_1 - R_1 X_1 & I_m \end{pmatrix}}_{\bar{X}_1} + \underbrace{\begin{pmatrix} T_1 & U_1 \\ -V_1 & W_1 \end{pmatrix}}_{P_1} \underbrace{\begin{pmatrix} Y_1 & 0 \\ 0 & 0 \end{pmatrix}}_{\bar{Y}_1} = \begin{pmatrix} I_{r_1} & 0 \\ 0 & I_m \end{pmatrix} \tag{198}$$

$$\underbrace{\begin{pmatrix} X_2 & -X_2 R_2 - Y_2 U_2 \\ 0 & I_\ell \end{pmatrix}}_{\bar{X}_2} \underbrace{\begin{pmatrix} Q_2 & R_2 \\ 0 & I_\ell \end{pmatrix}}_{S_2} + \underbrace{\begin{pmatrix} Y_2 & 0 \\ 0 & 0 \end{pmatrix}}_{\bar{Y}_2} \underbrace{\begin{pmatrix} T_2 & U_2 \\ -V_2 & W_2 \end{pmatrix}}_{P_2} = \begin{pmatrix} I_{r_2} & 0 \\ 0 & I_\ell \end{pmatrix} \tag{199}$$

Thus from the equations (191), (198) and (199)

$$\begin{pmatrix} S_1 & P_1 \\ \bar{Y}_2 & -\bar{X}_2 \end{pmatrix} \begin{pmatrix} \bar{X}_1 & P_2 \\ \bar{Y}_1 & -S_2 \end{pmatrix} = \begin{pmatrix} I_{r_1+m} & 0 \\ \bar{J} & I_{r_2+\ell} \end{pmatrix} \tag{200}$$

where $\bar{J} = \bar{Y}_2 \bar{X}_1 - \bar{X}_2 \bar{Y}_1$. Now Post-multiply (200) by $\begin{pmatrix} I & 0 \\ -\bar{J} & I \end{pmatrix}$ to obtain

$$\underbrace{\begin{pmatrix} M & P_1 \\ \bar{Y}_2 & -\bar{X}_2 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} \tilde{X}_1 & P_2 \\ \tilde{Y}_1 & -N \end{pmatrix}}_{B} = \begin{pmatrix} I_{r_1+p} & 0 \\ 0 & I_{r_2+m} \end{pmatrix} \tag{201}$$

where $\tilde{X}_1 = \bar{X}_1 - P_2\bar{J}$ and $\tilde{Y}_1 = \bar{Y}_1 + N\bar{J}$. Since $A_{(r+q)\times(p+s)}$ and $B_{(p+s)\times(r+q)}$ are square, the same size and are polynomial inverses of each other they must be unimodular and so commute, to give

$$\underbrace{\begin{pmatrix} \tilde{X}_1 & P_2 \\ \tilde{Y}_1 & -N \end{pmatrix}}_{B} \underbrace{\begin{pmatrix} M & P_1 \\ \bar{Y}_2 & -\bar{X}_2 \end{pmatrix}}_{A} = \begin{pmatrix} I_{r_1+p} & 0 \\ 0 & I_{r_2+m} \end{pmatrix}. \tag{202}$$

Thus the (1,2) equation gives

$$\tilde{X}_1 P_1 = P_2 \bar{X}_2$$

and it is known Theorem 3.9 that the correct coprime conditions hold, hence all that is required to prove is that the transforming matrices have the correct structure, namely that of S_1 and S_2. It is immediately obvious that $\bar{X}_2$ has the correct structure. Now $\tilde{X}_1 = \bar{X}_1 - P_2\bar{J}$

$$\begin{aligned}
\bar{J} &= \underbrace{\begin{pmatrix} Y_2 & 0 \\ 0 & 0 \end{pmatrix}}_{\bar{Y}_2} \underbrace{\begin{pmatrix} X_1 & 0 \\ V_1Y_1 - R_1X_1 & I_m \end{pmatrix}}_{\bar{X}_1} - \underbrace{\begin{pmatrix} X_2 & -X_2R_2 - Y_2U_2 \\ 0 & I_\ell \end{pmatrix}}_{\bar{X}_2} \underbrace{\begin{pmatrix} Y_1 & 0 \\ 0 & 0 \end{pmatrix}}_{\bar{Y}_1} \\
&= \begin{pmatrix} Y_2X_1 - X_2Y_1 & 0 \\ 0 & 0 \end{pmatrix} \\
\tilde{X}_1 &= \underbrace{\begin{pmatrix} X_1 & 0 \\ V_1Y_1 - R_1X_1 & I_m \end{pmatrix}}_{\bar{X}_1} - \underbrace{\begin{pmatrix} T_2 & U_2 \\ -V_2 & W_2 \end{pmatrix}}_{P_2} \begin{pmatrix} Y_2X_1 - X_2Y_1 & 0 \\ 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} X_1 - T_2(Y_2X_1 - X_2Y_1) & 0 \\ V_1Y_1R_1X_1 + V_2(Y_2X_1X_2Y_1) & I_m \end{pmatrix} \\
&= \begin{pmatrix} \tilde{Q}_1 & 0 \\ \tilde{R}_1 & I_m \end{pmatrix}.
\end{aligned}$$

Thus it has been shown that the relation is reflexive, transitive and symmetric hence it is an equivalence relation. □

As a consequence of this theorem if $P_2(z_1, z_2)$ is zero system equivalent to $P_1(z_1, z_2)$ then $P_1(z_1, z_2)$ is also zero system equivalent to $P_2(z_1, z_2)$. Hence it is not necessary to specify an order of the two matrices that are related by the transformation of zero system equivalence; it is equally valid to say that $P_1(z_1, z_2)$, $P_2(z_1, z_2)$ are zero system equivalent. Equally the same is true for zero equivalence. This property does not hold for minor equivalence and minor system equivalence as the transformations are not symmetric due to the nature of the Bézout identity. However, the properties of reflexivity and transitivity are valid for these transformations: the proofs follow in

an analogous manner to those for zero equivalence, Theorem 3.9, and zero system equivalence, Theorem 3.10. The property of minor system equivalence not being symmetric can be shown by considering the following example.

Example 3.3: Consider the two system matrices

$$P_1(z_1,z_2) = \left(\begin{array}{c|c} z_2 & z_1^2 \\ \hline -1 & 0 \end{array}\right) \tag{203}$$

$$P_2(z_1,z_2) = \left(\begin{array}{c|c} z_2 & z_1 \\ \hline -z_1 & 0 \end{array}\right) \tag{204}$$

with transfer function

$$G(z_1,z_2) = \frac{z_1^2}{z_2}$$

$P_2(z_1,z_2)$ is minor system equivalent to $P_1(z_1,z_2)$, as demonstrated by the equation

$$\begin{pmatrix} z_1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} z_2 & z_1 \\ -z_1 & 0 \end{pmatrix} = \begin{pmatrix} z_2 & z_1^2 \\ -1 & 0 \end{pmatrix}\begin{pmatrix} z_1 & 0 \\ 0 & 1 \end{pmatrix} \tag{205}$$

Each pair of matrices corresponding to the Definition 3.7 have non-zero high-order minors $-z_1$, z_2, $-z_1^2$ and z_1^2 thus defining minor coprimeness of the transforming and transformed matrices.

Conversely consider interchanging the roles of $P_1(z_1,z_2)$, $P_2(z_1,z_2)$ to give

$$\begin{pmatrix} q_1 & 0 \\ r_1 & 1 \end{pmatrix}\begin{pmatrix} z_2 & z_1^2 \\ -1 & 0 \end{pmatrix} = \begin{pmatrix} z_2 & z_1 \\ -z_1 & 0 \end{pmatrix}\begin{pmatrix} q_2 & r_2 \\ 0 & 1 \end{pmatrix} \tag{206}$$

It is immediately obvious, from Hilbert Nullstellensatz, that a polynomial solution does not exist for $q_1(z_1,z_2)$, $q_2(z_1,z_2)$, $r_1(z_1,z_2)$ and $r_2(z_1,z_2)$ by inspecting the (2,1) equation of (206):

$$r_1 z_2 - 1 = -z_1 q_2.$$

Thus $P_1(z_1,z_2)$ is not minor system equivalent to $P_2(z_1,z_2)$. □

In fact, this example provides some information about when a polynomial transformation of the form (161) exists. Consider the zeros of the matrix pairs T, U (input zeros) and T, V (output zeros) of the system matrices (203) and (204). The matrix $P_1(z_1,z_2)$ does not have any output zeros and the input zeros given by

$$(0,0), \quad (0,0).$$

The matrix $P_2(z_1,z_2)$ has (0,0) as an input and output zero. Notice that the set of input zeros of $P_2(z_1,z_2)$ are contained in the set of input zeros of $P_1(z_1,z_2)$ and the

set of output zeros of $P_1(z_1, z_2)$ are contained in the set of output zeros of $P_2(z_1, z_2)$; and $P_2(z_1, z_2)$ is minor system equivalent to $P_1(z_1, z_2)$. However, a polynomial transformation does not exist when the roles of $P_1(z_1, z_2)$ and $P_2(z_1, z_2)$ are interchanged. Thus it may be conjectured that for a polynomial form of the transformation

$$\begin{pmatrix} Q_1(z_1, z_2) & 0 \\ R_1(z_1, z_2) & I_m \end{pmatrix} \begin{pmatrix} T_2(z_1, z_2) & U_2(z_1, z_2) \\ -V_2(z_1, z_2) & W_2(z_1, z_2) \end{pmatrix}$$
$$= \begin{pmatrix} T_1(z_1, z_2) & U_1(z_1, z_2) \\ -V_1(z_1, z_2) & W_1(z_1, z_2) \end{pmatrix} \begin{pmatrix} Q_2(z_1, z_2) & R_2(z_1, z_2) \\ 0 & I_\ell \end{pmatrix}$$

to exist the set of zeros of $T_2(z_1, z_2)$, $U_2(z_1, z_2)$ are contained in the set of zeros of $T_1(z_1, z_2)$, $U_1(z_1, z_2)$ and the set of zeros of $T_1(z_1, z_2)$, $V_1(z_1, z_2)$ are contained in the set of zeros of $T_2(z_1, z_2)$, $V_2(z_1, z_2)$.

The previous two theorems indicate that zero system equivalence is the 2-D system theory equivalent of extended strict system equivalence for 1-D system theory. This fact may be further enhanced by considering some properties of extended strict system equivalence. Pugh *et al* [52] have proved that extended strict system equivalence preserves, the transfer function matrix, the set of system poles, the set of system zeros and all sets of decoupling zeros. Analogous results for zero system equivalence may be derived by making use of the 2-D MFD Structure Theorem, Theorem 2.5. However, as will be seen, some of these properties are also possessed by minor system equivalence. Thus an equivalence relation is not necessary for the invariance of certain fundamental system properties.

Theorem 3.11: *The transformations of minor system equivalence* (165) *and zero system equivalence* (163) *preserve*

(i) the transfer function matrix;

(ii) the invariant polynomials of $T_1(z_1, z_2)$, $T_2(z_1, z_2)$;

(iii) the invariant polynomials of $P_1(z_1, z_2)$, $P_2(z_1, z_2)$;

(iv) the invariant polynomials of the pair $T_i(z_1, z_2)$, $U_i(z_1, z_2)$ *for* $i = 1, 2$;

(v) the invariant polynomials of the pair $T_i(z_1, z_2)$, $V_i(z_1, z_2)$ *for* $i = 1, 2$.

Proof: The theorem needs only to be proved for minor system equivalence as zero system equivalence is a subset of minor system equivalence.

(i) This proved by Theorem 3.4.

(ii) Consider two system matrices $P_1(z_1, z_2)$, $P_2(z_1, z_2)$ with $P_2(z_1, z_2)$ being minor system equivalent to $P_1(z_1, z_2)$ thus there exist $S_1(z_1, z_2)$, $S_2(z_1, z_2)$ such that

$$S_1(z_1, z_2) P_2(z_1, z_2) = P_1(z_1, z_2) S_2(z_1, z_2) \tag{207}$$

where $S_1(z_1, z_2)$, $P_1(z_1, z_2)$ are minor (left) coprime and $S_2(z_1, z_2)$, $P_2(z_1, z_2)$ are minor (right) coprime with

$$S_1(z_1, z_2) = \begin{pmatrix} Q_1(z_1, z_2) & 0 \\ R_1(z_1, z_2) & I \end{pmatrix}; \qquad S_2(z_1, z_2) = \begin{pmatrix} Q_2(z_1, z_2) & R_2(z_1, z_2) \\ 0 & I \end{pmatrix}. \tag{208}$$

Thus the constituent equations are

$$Q_1 T_2 = T_1 Q_2 \tag{209}$$

$$Q_1 U_2 = T_1 R_2 + U_1 \tag{210}$$

$$R_1 T_2 - V_2 = -V_1 Q_2 \tag{211}$$

$$R_1 U_2 + W_2 = -V_1 R_2 + W_1 \tag{212}$$

By Theorem 3.5 equation (209) possesses the coprimeness conditions given by $Q_1(z_1, z_2)$, $T_1(z_1, z_2)$ minor (left) coprime and $Q_2(z_1, z_2)$, $T_2(z_1, z_2)$ minor (right) coprime and may be written as a minor coprime matrix fraction description

$$T_1^{-1}(z_1, z_2) Q_1(z_1, z_2) = Q_2(z_1, z_2) T_2^{-1}(z_1, z_2). \tag{213}$$

Thus by the 2-D MFD Structure Theorem, Theorem 2.5, $T_1(z_1, z_2)$ and $T_2(z_1, z_2)$ possess the same invariant polynomials modulo a multiplicative constant.

(iii) The result is obtained using a similar argument to (ii) and Corollary 3 of Theorem 2.5.

(iv) Consider the (1,1) and (1,2) equations of (207) in the form

$$Q_1 \begin{pmatrix} T_2 & U_2 \end{pmatrix} = \begin{pmatrix} T_1 & U_1 \end{pmatrix} \begin{pmatrix} Q_2(z_1, z_2) & R_2(z_1, z_2) \\ 0 & I \end{pmatrix} \tag{214}$$

and using Corollary 3 of Theorem 2.5 the result is established.

(v) In an analogous way to (iv), using (1,1) and (2,1) equations of (207) in the form

$$\begin{pmatrix} Q_1(z_1, z_2) & 0 \\ R_1(z_1, z_2) & I \end{pmatrix} \begin{pmatrix} T_2 \\ -V_2 \end{pmatrix} = \begin{pmatrix} T_1 \\ -V_1 \end{pmatrix} Q_2 \tag{215}$$

the result is established. □

However, the coprimeness of the T, U and T, V pairs can only be preserved under zero system equivalence, due to the relation being a true equivalence relation:

Theorem 3.12: *Zero system equivalence* (163) *preserves the coprimeness of the pairs* $T_i(z_1, z_2)$, $U_i(z_1, z_2)$ *and the pairs* $T_i(z_1, z_2)$, $V_i(z_1, z_2)$ *for* $i = 1, 2$ *of two system matrices* $P_1(z_1, z_2)$, $P_2(z_1, z_2)$.

Proof: Let the two system matrices $P_1(z_1, z_2), P_2(z_1, z_2)$ be related by a transformation of zero system equivalence then there exist polynomial matrix equations of the form

$$\begin{pmatrix} Q_1 & 0 \\ R_1 & I_m \end{pmatrix} \begin{pmatrix} T_2 & U_2 \\ -V_2 & W_2 \end{pmatrix} = \begin{pmatrix} T_1 & U_1 \\ -V_1 & W_1 \end{pmatrix} \begin{pmatrix} Q_2 & R_2 \\ 0 & I_\ell \end{pmatrix} \tag{216}$$

and by the symmetry of the relation

$$\begin{pmatrix} \bar{Q}_1 & 0 \\ \bar{R}_1 & I_m \end{pmatrix} \begin{pmatrix} T_1 & U_1 \\ -V_1 & W_1 \end{pmatrix} = \begin{pmatrix} T_2 & U_2 \\ -V_2 & W_2 \end{pmatrix} \begin{pmatrix} \bar{Q}_2 & \bar{R}_2 \\ 0 & I_\ell \end{pmatrix} \tag{217}$$

where $\bar{Q}_1(z_1, z_2)$, $T_1(z_1, z_2)$ and $\bar{Q}_1(z_1, z_2)$, $T_2(z_1, z_2)$ are zero (left) coprime and $\bar{Q}_2(z_1, z_2)$, $T_2(z_1, z_2)$ and $\bar{Q}_2(z_1, z_2)$, $T_2(z_1, z_2)$ are zero (right) coprime. The constituent equations of (216) are

$$Q_1 T_2 = T_1 Q_2 \tag{218}$$

$$Q_1 U_2 = T_1 R_2 + U_1 \tag{219}$$

$$R_1 T_2 - V_2 = -V_1 Q_2 \tag{220}$$

$$R_1 U_2 + W_2 = -V_1 R_2 + W_1 \tag{221}$$

and the constituent equations of (217) are

$$\bar{Q}_1 T_1 = T_2 \bar{Q}_2 \tag{222}$$

$$\bar{Q}_1 U_1 = T_2 \bar{R}_2 + U_2 \tag{223}$$

$$\bar{R}_1 T_1 - V_1 = -V_2 \bar{Q}_2 \tag{224}$$

$$\bar{R}_1 U_1 + W_1 = -V_2 \bar{R}_2 + W_2 \tag{225}$$

Assume that $T_1(z_1, z_2)$, $U_1(z_1, z_2)$ are minor (left) coprime, thus by the Bézout identities

$$T_1 X + U_1 Y = \psi_i I \quad \text{for} \quad i = 1, 2 \tag{226}$$

$$\bar{Q}_1 \bar{X} + T_2 \bar{Y} = I. \tag{227}$$

Thus pre-multiplying (226) by $\bar{Q}_1$ and using (222) and (223) gives

$$T_2 \bar{Q}_2 X + (T_2 \bar{R}_2 + U_2) Y = \psi_i \bar{Q}_1 \tag{228}$$

Collecting terms in T_2 and post multiplying by $\bar{X}$, using (227) gives

$$T_2(\bar{Q}_2 X + \bar{R}_2 Y)\bar{X} + U_2 Y \bar{X} = \psi_i (I - T_2 \bar{Y}) \tag{229}$$

Rearranging (229) gives the required equation for T_2, U_2 to be minor (left) coprime. The converse may be proved in an analogous manner.

Similarly T_1, V_1 are minor (right) coprime if and only if T_2, V_2 are minor (right) coprime.

The result for zero coprimeness holds by putting $\psi_i \equiv 1$ in (226), (228) and (229). □

Thus it has been seen that even though minor system equivalence is not a true equivalence relation, the fundamental properties of transfer function matrix, invariant polynomials of the T-blocks and the whole system matrix are preserved, and the invariant polynomials of the T, U and T, V pairs remain invariant, but to prove the invariance of the coprimeness of these pairs the property of symmetry is required.

To close this section consider the results of the previous 5 theorems in relation to the transformation that exists between the two types of least order system matrices defined in Section 3.3.1.

Theorem 3.13: *If $P_1(z_1, z_2)$, $P_2(z_1, z_2)$ are two (minor) least order system matrices related by a polynomial transformation*

$$\begin{pmatrix} Q_1(z_1,z_2) & 0 \\ R_1(z_1,z_2) & I_m \end{pmatrix} \begin{pmatrix} T_2(z_1,z_2) & U_2(z_1,z_2) \\ -V_2(z_1,z_2) & W_2(z_1,z_2) \end{pmatrix}$$
$$= \begin{pmatrix} T_1(z_1,z_2) & U_1(z_1,z_2) \\ -V_1(z_1,z_2) & W_1(z_1,z_2) \end{pmatrix} \begin{pmatrix} Q_2(z_1,z_2) & R_2(z_1,z_2) \\ 0 & I_\ell \end{pmatrix} \tag{230}$$

then $P_2(z_1, z_2)$ is minor system equivalent to $P_1(z_1, z_2)$.

Proof: Theorem 3.5 has established that if $P_1(z_1, z_2)$, $P_2(z_1, z_2)$ are (minor) least order system matrices and they possess the polynomial transformation (230) then $Q_1(z_1, z_2)$, $T_1(z_1, z_2)$ are minor (left) coprime and $Q_2(z_1, z_2)$, $T_2(z_1, z_2)$ are minor (right) coprime. Thus by Theorem 3.8 the coprimeness on the (1,1) block equation of (230) is synonymous with the coprimeness of the polynomial transformation (230). Therefore in (230) $P_2(z_1, z_2)$ is minor system equivalent to $P_1(z_1, z_2)$. □

Theorem 3.14: *If $P_1(z_1, z_2)$, $P_2(z_1, z_2)$ are two least order system matrices related by a polynomial transformation*

$$\begin{pmatrix} Q_1(z_1,z_2) & 0 \\ R_1(z_1,z_2) & I_m \end{pmatrix} \begin{pmatrix} T_2(z_1,z_2) & U_2(z_1,z_2) \\ -V_2(z_1,z_2) & W_2(z_1,z_2) \end{pmatrix}$$
$$= \begin{pmatrix} T_1(z_1,z_2) & U_1(z_1,z_2) \\ -V_1(z_1,z_2) & W_1(z_1,z_2) \end{pmatrix} \begin{pmatrix} Q_2(z_1,z_2) & R_2(z_1,z_2) \\ 0 & I_\ell \end{pmatrix} \tag{231}$$

then $P_1(z_1, z_2)$, $P_2(z_1, z_2)$ *are zero system equivalent.*

Proof: Theorem 3.6 has established that $P_1(z_1, z_2)$, $P_2(z_1, z_2)$ are least order system matrices and possess the polynomial transformation (231) then $Q_1(z_1, z_2)$, $T_1(z_1, z_2)$ are zero (left) coprime and $Q_2(z_1, z_2)$, $T_2(z_1, z_2)$ are zero (right) coprime. Thus by Theorem 3.8 the coprimeness on the (1,1) block equation of (231) is synonymous with the coprimeness of the polynomial transformation (231). Therefore (231) is a transformation of zero system equivalence. □

From the results derived in this section it has been seen that although the definition of (minor) least order in which the pairs T, U and T, V are both minor coprime was previously discounted as unsatisafactory, as a definition of least order, the only real difference existing between (minor) least order and least order is that the former allows the system matrices to be related by a true equivalence relation (zero system equivalence). However, for many of the properties studied in this section it is not necessary for the polynomial transformation to be an equivalence relation.

Unfortunately it has not proved possible to define a 2-D least order matrix so that it has similar properties to a 1-D least order matrix. Namely a necessary and sufficient condition for the existence of a polynomial transformation between two system matrices representing the same system, i.e. to be input-ouput equivalent.

3.4 Equivalence of n-D Polynomial Matrices

In the previous section (3.3) the relation existing between two system matrices, that are input-output equivalent, has been considered and links made with the definition of extended strict system equivalence. In this section the results for 2-D system matrices will be considered in an n-D framework. The system matrix now has the form

$$P(z) = \begin{pmatrix} T(z) & U(z) \\ -V(z) & W(z) \end{pmatrix}. \tag{232}$$

In many-indeterminate system theory the types of possible coprime matrices increases to three, namely zero, minor and factor; thus the definitions put forward in the previous section are sometimes not obviously generalised to many dimensions. For 2-D systems the n-D definitions of minor and factor coprimeness are equivalent; thus some definitions using minor coprimeness may be replaced by factor coprimeness, which is a less restrictive condition because a Bézout identity does not exist for this type of coprimeness.

In view of the three notions of coprimeness consider again the types of transformations that may be defined using these three notions. The definitions of zero equivalence, zero system equivalence, minor equivalence and minor system equivalence are defined by Definitions 3.6 and 3.7 but now a third type, factor equivalence and factor system equivalence, may also be given.

Definition 3.8: (*Factor Equivalence*)
Let $T_1(z)$, $T_2(z)$ be two polynomial matrices. If an equation of the form

$$Q_2(z)T_1(z) = T_2(z)Q_1(z) \tag{233}$$

exists where $T_1(z)$, $Q_1(z)$ are factor (left) coprime and $T_2(z)$, $Q_2(z)$ are factor (right) coprime then $T_2(z)$ is said to be *factor equivalent* to $T_1(z)$. Furthermore if two polynomial system matrices $P_1(z)$, $P_2(z)$ are related by an equation of the form

$$\begin{aligned} &\overbrace{\begin{pmatrix} Q_1(z) & 0 \\ R_1(z) & I_m \end{pmatrix}}^{S_1(z)} \overbrace{\begin{pmatrix} T_2(z) & U_2(z) \\ -V_2(z) & W_2(z) \end{pmatrix}}^{P_2(z)} \\ &\qquad = \underbrace{\begin{pmatrix} T_1(z) & U_1(z) \\ -V_1(z) & W_1(z) \end{pmatrix}}_{P_1(z)} \underbrace{\begin{pmatrix} Q_2(z) & R_2(z) \\ 0 & I_\ell \end{pmatrix}}_{S_2(z)} \end{aligned} \tag{234}$$

where $S_1(z)$, $P_1(z)$ are factor (right) coprime and $S_2(z)$, $P_2(z)$ are factor (left) coprime then $P_2(z)$ is said to be *factor system equivalent* to $P_1(z)$. □

The analogue of Theorem 3.8 may be defined, but the proof is not a trivial generalisation of the 2-D result.

Theorem 3.15: *Let the two system matrices $P_1(z), P_2(z)$ be related by a transformation of the form*

$$S_1(z)P_2(z) = P_1(z)S_2(z) \tag{235}$$

where

$$S_1(z) = \begin{pmatrix} Q_1(z) & 0 \\ R_1(z) & I_m \end{pmatrix} \quad S_2 = \begin{pmatrix} Q_2(z) & R_2(z) \\ 0 & I_\ell \end{pmatrix}.$$

Then

(a) *$P_1(z)$, $S_1(z)$ are factor (left) coprime if and only if $T_1(z)$, $Q_1(z)$ are factor (left) coprime;*

(b) *$P_1(z)$, $S_1(z)$ are minor (left) coprime if and only if $T_1(z)$, $Q_1(z)$ are minor (left) coprime;*

(c) *$P_1(z)$, $S_1(z)$ are zero (left) coprime if and only if $T_1(z)$, $Q_1(z)$ are zero (left) coprime.*

Similar statements may be formulated for the pairs $P_2(z)$, $S_2(z)$ and $T_2(z)$, $Q_2(z)$.

Proof:

(a) Assume that T_1, Q_1 are factor (left) coprime, then any common (left) factor of T_1 and Q_1 is unimodular, i.e. an n-D polynomial matrix with non-zero constant determinant. Now suppose on the contrary that P_1, S_1 are not factor (left) coprime. Let D be a greatest common (left) divisor of P_1 and S_1 then $|D| \neq$ constant, i.e.

$$\underbrace{\begin{pmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{pmatrix}}_{D} \begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \end{pmatrix} = \begin{pmatrix} Q_1 & 0 & T_1 & U_1 \\ R_1 & I & -V_1 & W_1 \end{pmatrix} \tag{236}$$

Thus from the (1,2) and (2,2) equations

$$D_{11}A_{12} + D_{12}A_{22} = 0 \tag{237}$$

$$D_{21}A_{12} + D_{22}A_{22} = I \tag{238}$$

hence from (238) A_{12} and A_{22} are zero (right) coprime. By Theorem 2.3 there exist polynomial matrices $N_1(z)$ and $N_2(z)$ such that

$$\bar{U}_1 = \begin{pmatrix} N_1 & A_{12} \\ N_2 & A_{22} \end{pmatrix}$$

is unimodular. Let $\bar{U}$ be the unimodular matrix

$$\bar{U} = \begin{pmatrix} N_1 & A_{12} \\ N_2 & A_{22} \end{pmatrix} \begin{pmatrix} I & 0 \\ -D_{21}N_1 - D_{22}N_2 & I \end{pmatrix}$$
$$= \begin{pmatrix} N_1 - A_{12}(D_{21}N_1 + D_{22}N_2) & A_{12} \\ N_2 - A_{22}(D_{21}N_1 + D_{22}N_2) & A_{22} \end{pmatrix}$$

then

$$D\bar{U} = \begin{pmatrix} D_{11}N_1 + D_{12}N_2 - (D_{11}A_{12} + D_{12}A_{22})(D_{21}N_1 + D_{22}N_2) & D_{11}A_{12} + D_{12}A_{22} \\ D_{21}N_1 + D_{22}N_2 - (D_{21}A_{12} + D_{22}A_{22})(D_{21}N_1 + D_{22}N_2) & D_{21}A_{12} + D_{22}A_{22} \end{pmatrix}$$

From (237) and (238)

$$D\bar{U} = \begin{pmatrix} \bar{D}_{11} & 0 \\ 0 & I \end{pmatrix} \tag{239}$$

where $\bar{D}_{11} = D_{11}N_1 + D_{12}N_2$. Thus from (239) $|\bar{D}_{11}| = |D| \times |\bar{U}|$ where $|\bar{U}|$ is a non-zero constant. Thus (236) can be written as

$$\begin{pmatrix} \bar{D}_{11} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} \bar{A}_{11} & \bar{A}_{12} & \bar{A}_{13} & \bar{A}_{14} \\ \bar{A}_{21} & \bar{A}_{22} & \bar{A}_{23} & \bar{A}_{24} \end{pmatrix} = \begin{pmatrix} Q_1 & 0 & T_1 & U_1 \\ R_1 & I & -V_1 & W_1 \end{pmatrix}$$

where

$$\begin{pmatrix} \bar{A}_{11} & \bar{A}_{12} & \bar{A}_{13} & \bar{A}_{14} \\ \bar{A}_{21} & \bar{A}_{22} & \bar{A}_{23} & \bar{A}_{24} \end{pmatrix} = \bar{U}^{-1} \begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \end{pmatrix}$$

is a polynomial matrix. Hence $[Q_1|T_1] = \bar{D}_{11}[\bar{A}_{11}|\bar{A}_{13}]$. Thus T_1, Q_1 are not factor (left) coprime, which is a contradiction to the assumption. Hence P_1, S_1 are factor (left) coprime.

Conversely, assume that P_1, S_1 are factor (left) coprime and suppose that T_1, Q_1 are not factor (left) coprime. Thus there exists an n-D polynomial matrix D with $|D| \neq$ constant such that $[T_1|Q_1] = D[\bar{T}_1|\bar{Q}_1]$. Now

$$\begin{pmatrix} I & 0 & 0 & -U_2 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & R_2 \\ 0 & 0 & 0 & I \end{pmatrix} \begin{pmatrix} I & 0 & 0 & U_2 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & -R_2 \\ 0 & 0 & 0 & I \end{pmatrix} = \begin{pmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{pmatrix}.$$

Thus

$$\begin{pmatrix} Q_1 & 0 & T_1 & U_1 \\ R_1 & I & -V_1 & W_1 \end{pmatrix}$$
$$= \begin{pmatrix} D\bar{Q}_1 & 0 & D\bar{T}_1 & U_1 \\ R_1 & I & -V_1 & W_1 \end{pmatrix} \begin{pmatrix} I & 0 & 0 & -U_2 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & R_2 \\ 0 & 0 & 0 & I \end{pmatrix} \begin{pmatrix} I & 0 & 0 & U_2 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & -R_2 \\ 0 & 0 & 0 & I \end{pmatrix}.$$

From (235) $-Q_1U_2 + T_1R_2 + U_1 = 0$, i.e. $-D\bar{Q}_1U_2 + D\bar{T}_1R_2 + U_1 = 0$

$$= \begin{pmatrix} D\bar{Q}_1 & 0 & D\bar{T}_1 & 0 \\ R_1 & I & -V_1 & W_1 - R_1U_2 - V_1R_2 \end{pmatrix} \begin{pmatrix} I & 0 & 0 & U_2 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & -R_2 \\ 0 & 0 & 0 & I \end{pmatrix}$$

$$= \begin{pmatrix} D & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} \bar{Q}_1 & 0 & \bar{T}_1 & 0 \\ R_1 & I & -V_1 & W_1 - R_1U_2 - V_1R_2 \end{pmatrix} \begin{pmatrix} I & 0 & 0 & U_2 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & -R_2 \\ 0 & 0 & 0 & I \end{pmatrix}$$

$$= \begin{pmatrix} D & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} \bar{Q}_1 & 0 & \bar{T}_1 & \bar{Q}_1U_2 - \bar{T}_1R_2 \\ R_1 & I & -V_1 & W_1 \end{pmatrix}.$$

But this gives a contradiction as

$$\begin{vmatrix} D & 0 \\ 0 & I \end{vmatrix} = |D| \neq \text{constant}.$$

Hence T_1 and Q_1 are factor (left) coprime.

(b) The proof is analogous to Theorem 3.8 (a).

(c) The proof is analogous to Theorem 3.8 (b). □

Theorem 3.16: *The relation of*

(i) zero equivalence is reflexive, transitive and symmetric;

(ii) minor equivalence is reflexive and transitive;

(iii) factor equivalence is reflexive.

Proof:

(i) The proof is analogous to Theorem 3.9.

(ii) Let $T_1(z)$ and $T_2(z)$ be two polynomial matrices, respectively of sizes $p \times q$ and $r \times s$ with $p - q = r - s$. Let $Q_1(z)$ and $Q_2(z)$ be two polynomial matrices such that

$$Q_2T_1 = T_2Q_1 \tag{240}$$

with $Q_1(z)$, $T_1(z)$ minor (right) coprime and $Q_2(z)$, $T_2(z)$ minor (left) coprime.

Reflexivity: Let $T_1 \equiv T_2$ in (240). Then $p = r$ and $q = s$. If $Q_1 = I_q$ and $Q_2 = I_r$ then

$$I_rT_1 = T_2I_\ell$$

and since $\det I = 1$ T_1, I_r are zero (left) coprime and T_2, I_q are zero (right) coprime they are also minor coprime.

Transitivity: Suppose that

$$Q_2T_1 = T_2Q_1 \tag{241}$$

$$\bar{Q}_2T_2 = T_3\bar{Q}_1 \tag{242}$$

with the usual minor equivalence coprime conditions. From (241)

$$\begin{aligned}\bar{Q}_2(Q_2T_1) &= \bar{Q}_2(T_2Q_1)\\ &= T_3\bar{Q}_1Q_1.\end{aligned} \tag{243}$$

Thus it is required to prove that $\bar{Q}_2Q_2$, T_3 are minor (left) coprime and that $\bar{Q}_1Q_1$, T_1 are minor (right) coprime. From the coprime conditions and the Bézout identities there exist $X_i^{(1)}$, $X_i^{(2)}$, $Y_i^{(1)}$ and $Y_i^{(2)}$ for $i = 1, 2, \ldots, n$ such that

$$Q_2X_i^{(1)} + T_2Y_i^{(1)} = \psi_i(z_i^c)I \tag{244}$$

$$\bar{Q}_2X_i^{(2)} + T_3Y_i^{(2)} = \phi_i(z_i^c)I \tag{245}$$

where $\psi_i(z_i^c)$, $\phi_i(z_i^c)$ are polynomials independent of z_i. Pre-multiplying (244) by $\bar{Q}_2$ and using (243) gives

$$\begin{aligned}\bar{Q}_2Q_2X_i^{(1)} + \bar{Q}_2T_2Y_i^{(1)} &= \psi_i(z_i^c)\bar{Q}_2\\ \bar{Q}_2Q_2X^{(1)} + T_3\bar{Q}_1Y_i^{(1)} &= \psi_i(z_i^c)\bar{Q}_2.\end{aligned}$$

Now post-multiply by $X_i^{(2)}$ and use (245) to give

$$\bar{Q}_2Q_2X_i^{(1)}X_i^{(2)} + T_3\bar{Q}_1Y_i^{(1)}X_i^{(2)} = \psi_i(z_i^c)(\phi_i(z_i^c)I - T_3Y_i^{(2)})$$

and rearrange to give

$$\bar{Q}_2Q_2(X^{(1)}X_i^{(2)}) + T_3(\bar{Q}_1Y_i^{(1)}X_i^{(2)} + \psi_i(z_i^c)Y_i^{(2)}) = \psi_i(z_i^c)\phi_i(z_i^c)I.$$

Thus $\bar{Q}_2Q_2$ and T_3 are minor (left) coprime. In the same way it can be proved that $\bar{Q}_1Q_1$ and T_2 are minor (right) coprime.

(iii) Let the matrices defined in (241) have the properties $Q_1(z)$, $T_1(z)$ factor (right) coprime and $Q_2(z)$, $T_2(z)$ factor (left) coprime. Reflexivity follows as in (ii) above. □

From Example 3.3 the relations of minor and factor equivalence can not be symmetric. The transitivity of factor equivalence is unlikely due to the non-existence of a Bézout identity. Following a similar analysis to the proof of the transitive property of minor equivalence equations (241), (242) hold with factor coprime conditions. Thus in equation (243) T_1, Q_1 are factor (right) coprime but this does provide information about the coprimeness of T_1 and $\bar{Q}_1Q_1$, similarly for $\bar{Q}_2Q_2$ and T_3.

The two previous theorems may now be used to derive the types of relation that exist between the system equivalence relations.

Corollary: *The relation of*

(i) zero system equivalence is reflexive, transitive and symmetric;

(ii) minor system equivalence is reflexive and transitive but not symmetric;

(iii) factor system equivalence is reflexive but not symmetric.

Proof: The proof follows in a manner analagous to the proofs of Theorem 3.10, 3.16 and the counter-example Example 3.3. □

Again the transitivity of factor system equivalence is doubtful for the same reasons derived for factor equivalence, above.

Theorem 3.11, concerning some invariant properties of minor and zero system equivalence, holds for the n-D case with analogous proofs using the n-D MFD Structure Theorem (Theorem 2.12). However, the majority of the results can not be proved for factor system equivalence due to Theorem 2.12 not being valid for factor coprime MFDs and the reliance of the proof on the existence of a Bézout Identity.

Theorem 3.17: *The transformations of minor system equivalence and zero system equivalence preserves*

(i) the transfer function matrix;

(ii) the invariant polynomials of $T_1(z)$, $T_2(z)$;

(iii) the invariant polynomials of $P_1(z)$, $P_2(z)$;

(iv) the invariant polynomials of the pair $T_i(z)$, $U_i(z)$ for $i = 1, 2$;

(v) the invariant polynomials of the pair $T_i(z)$, $V_i(z)$ for $i = 1, 2$.

Proof: The proof is analogous to Theorem 3.11. □

Theorem 3.10, concerning the invariance of certain coprime conditions on the constituent matrices of the system matrix, also hold for the n-D case. However, the theorem statement is not as elegant as for the 2-D case. This arises from the third notion of coprimeness, factor coprimeness, not possessing a Bézout identity.

Theorem 3.18: *Let the system matrices $P_1(z)$ and $P_2(z)$ defined by*

$$P_i(z) = \begin{array}{c} \\ r_i \\ m \end{array} \begin{array}{c} \begin{array}{cc} r_i & \ell \end{array} \\ \begin{pmatrix} T_i(z) & U_i(z) \\ -V_i(z) & W_i(z) \end{pmatrix} \end{array} \quad \text{for} \quad i = 1, 2 \tag{246}$$

be zero system equivalent. Then

(a) $T_1(z)$, $U_1(z)$ are zero (left) coprime if and only if $T_2(z)$, $U_2(z)$ are zero (right) coprime;

(b) $T_1(z)$, $U_1(z)$ *are minor (left) coprime if and only if* $T_2(z)$, $U_2(z)$ *are minor (right) coprime;*

(c) $T_1(z)$, $V_1(z)$ *are zero (right) coprime if and only if* $T_2(z)$, $V_2(z)$ *are zero (right) coprime;*

(d) $T_1(z)$, $V_1(z)$ *are minor (right) coprime if and only if* $T_2(z)$, $V_2(z)$ *are minor (right) coprime.*

Proof: The proof is analogous to Theorem 3.12. □

Finally consider the definition of least order (Definition 3.5) and the polynomial transformation connecting least order system matrices in n-D systems theory. It does not seem appropriate to adopt Definition 3.5, where the T, U matrices of the system matrix are zero (left) coprime and T, V are minor (right) coprime due to the notion of factor coprimeness. Minor coprimeness was adopted in the definition of least order for 2-D system matrices because by considering the matrix fraction description every transfer function matrix $G(z_1, z_2)$ could be realised as such. For n-D systems minor coprime MFDs can not be guaranteed in general. However, factor coprimeness for n-D systems fulfils the role of minor coprimeness for 2-D systems in this instance. Thus a natural extension to Definition 3.5 for least order seems to be the following.

Definition 3.9: (*Least Order*)
A n-D polynomial system matrix $P(z)$ of the form (246) will be said to be *least order* if the pair $T(z)$, $U(z)$ are factor (left) coprime and the pair $T(z)$, $V(z)$ are zero (right) coprime. □

Equally least order could be defined in which $T(z)$ and $U(z)$ are zero (left) coprime and $T(z)$ and $V(z)$ are factor (right) coprime. Once again the important point seems to be that it is necessary to agree at the outset exactly how the non-essential singularities of the second kind are to be realised. However, it is not possible to deduce that if two least order system matrices are related by a polynomial transformation of the form (230) then they are zero system equivalent. Thus Theorem 3.6 becomes the weaker statement:

Theorem 3.19: *If two least order system matrices (Definition 3.9) $P_i(z)$ for $i = 1, 2$ are related by a polynomial form of the relationship*

$$\begin{pmatrix} Q_1(z) & 0 \\ R_1(z) & I_m \end{pmatrix} \begin{pmatrix} T_2(z) & U_2(z) \\ -V_2(z) & W_2(z) \end{pmatrix} = \begin{pmatrix} T_1(z) & U_1(z) \\ -V_1(z) & W_1(z) \end{pmatrix} \begin{pmatrix} Q_2(z) & R_2(z) \\ 0 & I_\ell \end{pmatrix} \quad (247)$$

then $Q_1(z)$, $T_1(z)$ are factor (left) coprime and $Q_2(z)$, $T_2(z)$ are zero (right) coprime.

Proof: The (1,2) equation of (247) is

$$U_1 = Q_1 U_2 - T_1 R_2 \quad (248)$$

Suppose that $Q_1(z)$, $T_1(z)$ are not factor (left) coprime then there exists a non-unimodular polynomial matrix $D(z)$ such that $[Q_1 \; T_1] = D[\bar{Q}_1 \; \bar{T}_1]$ for polynomial matrices $\bar{Q}_1(z)$, $\bar{T}_1(z)$. Thus (248) gives

$$U_1 = D(\bar{Q}_1 U_2 - \bar{T}_1 R_2)$$

i.e. $D(z)$ is a common left divisor of $T_1(z)$, $U_1(z)$. This is a contradiction because the system matrix $P_1(z)$ has the property of least order and therefore $T_1(z)$, $U_1(z)$ do not have any non-unimodular polynomial divisors. Thus $Q_1(z)$, $T_1(z)$ are factor (left) coprime.

It may now be proved analogously to Theorem 3.6 that $Q_2(z)$, $T_2(z)$ are zero (right) coprime. □

Consider now an alternative definition of a least order system matrix $P(z)$ such that $T(z)$, $U(z)$ are factor (left) coprime and $T(z)$, $V(z)$ are minor (right) coprime. This clearly is a more general type of system matrix than the least order system matrix defined in Definition 3.9, since zero coprimeness is contained as a proper subset of minor coprimeness (see Theorem 1.3); in fact, if the number of indeterminates is two this alternative definition is (minor) least order. Therefore it is expected that the coprimeness conditions induced on a polynomial transformation of the type (247) are less restrictive than in the least order case. Hence the following theorem may be stated.

Theorem 3.20: *If two system matrices $P_1(z)$, $P_2(z)$ are related by a polynomial transformation of the form*

$$\begin{pmatrix} Q_1(z) & 0 \\ R_1(z) & I_m \end{pmatrix} \begin{pmatrix} T_2(z) & U_2(z) \\ -V_2(z) & W_2(z) \end{pmatrix} = \begin{pmatrix} T_1(z) & U_1(z) \\ -V_1(z) & W_1(z) \end{pmatrix} \begin{pmatrix} Q_2(z) & R_2(z) \\ 0 & I_\ell \end{pmatrix}$$

where $T_i(z)$, $V_i(z)$ are minor (right) coprime for $i = 1, 2$ then $Q_1(z)$, $T_1(z)$ are factor (left) coprime and $Q_2(z)$, $T_2(z)$ are minor (right) coprime.

Proof: The factor coprimeness of $T_1(z_1, z_2)$, $U_1(z_1, z_2)$ is proved analogously to Theorem 3.19 and the minor coprimeness of $Q_2(z)$, $T_2(z)$ follows from Theorem 3.5. □

Thus n-D least order system matrices are not related by minor or zero equivalence whenever a polynomial form of the transformation (248) exists.

3.5 Linear Differential Multipass Processes

One of the major considerations for the equivalence of system matrices is the invariance of the zeros possessed by that system. In this section the zeros of a particular type of 2-D system are investigated, known as linear differential multipass processes. Also the feasibility of constructing a least order system matrix, in the sense of Definition 3.5, is considered.

3.5.1 Non-Unit Memory Multipass Processes

A general multipass process is a dynamic system, possibly non-linear, with a repetitive, or recursive, action with interaction between successive passes, which may have differing lengths. A subclass of these multipass processes are those with linear dynamics and constant pass length, α. A number of examples are known [73] to satisfy these conditions, e.g. a type of long-wall coal cutting, [74]. Formally the pass profile (output vector) $Y_k(t)$, $0 \leq t \leq \alpha$ (t is the distance variable along each pass), generated during the kth pass acts as a forcing function on and hence contributes to the dynamics of the next output vector $Y_{k+1}(t)$ $0 \leq t \leq \alpha$ $k \geq 0$. If information from a number of previous passes contributes to the current pass the process is known as a linear differential non-unit memory multipass process.

Two parameters are required to specify each of the system variables thus the transfer function arising from such a system will be a polynomial matrix in two indeterminates. One 'dimension' is discrete representing the pass number, k, and the second 'dimension' is continuous representing the 'distance' along each pass, t. The defining state-space model [73], [75] has the form

$$\begin{aligned} \dot{X}_{k+1}(t) &= AX_{k+1}(t) + BU_{k+1}(t) + \sum_{j=1}^{M} B_{j-1}Y_{k+1-j}(t) \\ Y_{k+1}(t) &= CX_{k+1}(t) + D_0U_{k+1} + \sum_{j=1}^{M} D_jY_{k+1-j}(t) \end{aligned} \tag{249}$$

where the input $X_{k+1}(t) \in \mathbb{R}^n$, the output $Y_{k+1}(t) \in \mathbb{R}^m$, the control $U_{k+1}(t) \in \mathbb{R}^\ell$ for $0 \leq t \leq \alpha$, the initial conditions $X_{k+1}(0) = d_{k+1}$ for $k \geq 0$ and M is the number of previous passes contributing to the current pass.

The well known transformations of 'z' and 'Laplace' can be generalised and applied to the multipass process to define the following [76].

The 'z-transform' with respect to the pass number and the Laplace or 's-transform' with respect to the along the pass variable t of the series $X_{k+1}(t)$, $Y_{k+1}(t)$, $U_{k+1}(t)$ are defined by

$$\begin{aligned} X(s,z) &= \mathfrak{L}X_1(t) + z^{-1}\mathfrak{L}X_2(t) + z^{-2}\mathfrak{L}X_3(t) + \cdots \\ Y(s,z) &= \mathfrak{L}Y_1(t) + z^{-1}\mathfrak{L}Y_2(t) + z^{-2}\mathfrak{L}Y_3(t) + \cdots \\ U(s,z) &= \mathfrak{L}U_1(t) + z^{-1}\mathfrak{L}U_2(t) + z^{-2}\mathfrak{L}U_3(t) + \cdots \end{aligned}$$

where $\mathfrak{L}$ denotes the Laplace transform with respect to t.

Note: The indeterminate z may be viewed as a backward shift operator and therefore only appears with negative powers. The expressions $X(s,z)$, $Y(s,z)$, $U(s,z)$ are thus polynomial in z^{-1}.

Applying these transforms to the differential equations (249) the input-output description becomes

$$Y(s,z) = G(s,z)U(s,z)$$

where $Y(s,z)$ is the output vector, $G(s,z)$ is the transfer function matrix and $U(s,z)$ is the input vector. The transfer function $G(s,z)$ is defined by

$$\begin{aligned} G(s,z) = &\Big(I_m - D(z)\Big)^{-1} C\Big\{sI_n - A - B(z)\Big(I_m - D(z)\Big)^{-1} C\Big\}^{-1} \\ &\times \Big\{B + B(z)\Big(I_m - D(z)\Big)^{-1} D_0\Big\} + \Big(I_m - D(z)\Big)^{-1} D_0 \end{aligned} \tag{250}$$

where

$$D(z) = \sum_{j=1}^{M} D_j z^{-j} \in \mathbb{R}^{m\times m}[z^{-1}], \qquad B(z) = \sum_{j=1}^{M} B_{j-1} z^{-j} \in \mathbb{R}^{n\times m}[z^{-1}]$$

and

$$A \in \mathbb{R}^{n\times n}, \qquad B \in \mathbb{R}^{n\times \ell}, \qquad C \in \mathbb{R}^{m\times n}, \qquad D_0 \in \mathbb{R}^{m\times \ell}$$

It is now shown, Theorem 3.21, that this transfer function matrix $G(s,z)$ can be written as a product of polynomial matrices, one polynomial in s and z^{-1} and the other polynomial in s only. To prove this first recall the general matrix identities:

Lemma 3.3: *Let the matrices X and Y be of sizes $m \times \ell$ and $\ell \times m$. Then*

(i) $I_m + X(I_\ell - YX)^{-1}Y = (I_m - XY)^{-1}$,

(ii) $(XY)^{-1} = Y^{-1}X^{-1}$ if $m = \ell$.

Proof:

(i)

$$\begin{aligned}(I_m - XY)(I_m + X(I_\ell - YX)^{-1}Y) &= I_m + X(I_\ell - YX)^{-1}Y - XY - XYX(I_\ell - YX)^{-1}Y \\ &= I_m + X[(I_\ell - YX)^{-1} - I_\ell - YX(I_\ell - YX)^{-1}]Y \\ &= I_m + X[(I_\ell - YX)(I_\ell - YX)^{-1} - I_\ell]Y \\ &= I_m\end{aligned}$$

(ii)

$$\begin{aligned}(XY)(Y^{-1}X^{-1}) &= X(YY^{-1})X^{-1} \\ &= XI_mX^{-1} \\ &= XX^{-1} \\ &= I_m\end{aligned}$$

□

Using the above identities an equivalent form of the transfer function matrix can be derived, as noted in [76]. However, the derivation of this alternative form does not appear to be available in the literature; it is given here:

Theorem 3.21:

$$G(s,z) = \left(I_m - \sum_{j=1}^{M} G_j(s)z^{-j}\right)^{-1} G_0(s)$$

where

$$G_0(s) = C(sI_n - A)^{-1}B + D_0 \in \mathbb{R}^{m\times\ell}(s)$$

$$G_j(s) = C(sI_n - A)^{-1}B_{j-1} + D_j \quad \text{for} \quad 1 \le j \le M \in \mathbb{R}^{m\times m}(s)$$

Proof:

From (250)

$$\begin{aligned}G(s,z) = &\Big(I_m - D(z)\Big)^{-1} C\left[sI_n - A - B(z)\Big(I_m - D(z)\Big)^{-1}C\right]^{-1} \\ &\times \left[B + B(z)\Big(I_m - D(z)\Big)^{-1}D_0\right] + \Big(I_m - D(z)\Big)^{-1}D_0\end{aligned}$$

Remove $(sI_n - A)^{-1}$ from the first square bracket

$$=\Big(I_m - D(z)\Big)^{-1}C\Big[I_n - \underbrace{(sI_n - A)^{-1}}_{X}\underbrace{B(z)\Big(I_m - D(z)\Big)^{-1}C}_{Y}\Big]^{-1}\ (sI_n - A)^{-1}$$
$$\times\left[B + B(z)\Big(I_m - D(z)\Big)^{-1}D_0\right] + \Big(I_m - D(z)\Big)^{-1}D_0$$

Using Lemma 3.3 (i) with X, Y as shown

$$=\Big(I_m - D(z)\Big)^{-1}C$$
$$\times\left\{I_n + \ (sI_n - A)^{-1}\left[I_n - B(z)\Big(I_m - D(z)\Big)^{-1}C\ (sI_n - A)^{-1}\right]^{-1}\right.$$
$$\left.\times B(z)\Big(I_m - D(z)\Big)^{-1}C\right\}\ (sI_n - A)^{-1}\left[B + B(z)\Big(I_m - D(z)\Big)^{-1}D_0\right]$$
$$+\Big(I_m - D(z)\Big)^{-1}D_0$$

Recombine $(sI_n - A)^{-1}$ into the first square bracket to give

$$=\Big(I_m - D(z)\Big)^{-1}C\left\{I_n + \left[sI_n - A - B(z)\Big(I_m - D(z)\Big)^{-1}C\right]^{-1}\right.$$
$$\left.\times B(z)\Big(I_m - D(z)\Big)^{-1}C\right\}\ (sI_n - A)^{-1}\left[B + B(z)\Big(I_m - D(z)\Big)^{-1}D_0\right]$$
$$+\Big(I_m - D(z)\Big)^{-1}D_0$$

Multiplying out the brackets gives

$$=\Big(I_m - D(z)\Big)^{-1}C\ (sI_n - A)^{-1}B(z)\Big(I_m - D(z)\Big)^{-1}D_0$$
$$+\Big(I_m - D(z)\Big)^{-1}C\ (sI_n - A)^{-1}B$$
$$+\Big(I_m - D(z)\Big)^{-1}C\left[sI_n - A - B(z)\Big(I_m - D(z)\Big)^{-1}C\right]^{-1}$$
$$\times B(z)\Big(I_m - D(z)\Big)^{-1}C\ (sI_n - A)^{-1}B(z)\Big(I_m - D(z)\Big)^{-1}D_0$$
$$+\Big(I_m - D(z)\Big)^{-1}C\left[sI_n - A - B(z)\Big(I_m - D(z)\Big)^{-1}C\right]^{-1}$$
$$\times B(z)\Big(I_m - D(z)\Big)^{-1}C\ (sI_n - A)^{-1}B$$
$$+\Big(I_m - D(z)\Big)^{-1}D_0$$

Rewrite, combining first and third terms

$$
\begin{aligned}
=&\Big(I_m - D(z)\Big)^{-1} D_0 + \Big(I_m - D(z)\Big)^{-1} C\ (sI_n - A)^{-1} B \\
&+ \Big(I_m - D(z)\Big)^{-1} C \left[sI_n - A - B(z)\Big(I_m - D(z)\Big)^{-1} C\right]^{-1} \\
&\quad \times B(z)\Big(I_m - D(z)\Big)^{-1} C\ (sI_n - A)^{-1} B \\
&+ \Big(I_m - D(z)\Big)^{-1} C \left\{\left[sI_n - A - B(z)\Big(I_m - D(z)\Big)^{-1} C\right]^{-1}\right. \\
&\quad \left. \times B(z)\Big(I_m - D(z)\Big)^{-1} C + I_n\right\}\ (sI_n - A)^{-1} B(z)\Big(I_m - D(z)\Big)^{-1} D_0
\end{aligned}
$$

Remove a factor of $[sI_n - A - B(z)\Big(I_m - D(z)\Big)^{-1} C]^{-1}$ from curly bracket

$$
\begin{aligned}
=&\Big(I_m - D(z)\Big)^{-1} D_0 + \Big(I_m - D(z)\Big)^{-1} C\ (sI_n - A)^{-1} B \\
&+ \Big(I_m - D(z)\Big)^{-1} C \left[sI_n - A - B(z)\Big(I_m - D(z)\Big)^{-1} C\right]^{-1} \\
&\quad \times B(z)\Big(I_m - D(z)\Big)^{-1} C\ (sI_n - A)^{-1} B \\
&+ \Big(I_m - D(z)\Big)^{-1} C \left[sI_n - A - B(z)\Big(I_m - D(z)\Big)^{-1} C\right]^{-1} \\
&\quad \times \left\{B(z)\Big(I_m - D(z)\Big)^{-1} C + sI_n - A - B(z)\Big(I_m - D(z)\Big)^{-1} C\right\} \\
&\quad \times\ (sI_n - A)^{-1} B(z)\Big(I_m - D(z)\Big)^{-1} D_0
\end{aligned}
$$

Combining curly bracket terms gives $sI_n - A$ which then cancels with $(sI_n - A)^{-1}$

$$
\begin{aligned}
=&\Big(I_m - D(z)\Big)^{-1} D_0 + \Big(I_m - D(z)\Big)^{-1} C\ (sI_n - A)^{-1} B \\
&+ \Big(I_m - D(z)\Big)^{-1} C \left[sI_n - A - B(z)\Big(I_m - D(z)\Big)^{-1} C\right]^{-1} \\
&\quad \times B(z)\Big(I_m - D(z)\Big)^{-1} C\ (sI_n - A)^{-1} B \\
&+ \Big(I_m - D(z)\Big)^{-1} C \left[sI_n - A - B(z)\Big(I_m - D(z)\Big)^{-1} C\right]^{-1} \\
&\quad \times\ (sI_n - A)^{-1} B(z)\Big(I_m - D(z)\Big)^{-1} D_0
\end{aligned}
$$

Combine terms to give a factor of $C\ (sI_n - A)^{-1}B + D_0$

$$
\begin{aligned}
&= \Big(I_m - D(z)\Big)^{-1}\Big[C\ (sI_n - A)^{-1}B + D_0\Big] \\
&\quad + \Big(I_m - D(z)\Big)^{-1} C \Big[sI_n - A - B(z)\Big(I_m - D(z)\Big)^{-1} C\Big]^{-1} \\
&\quad \times B(z)\Big(I_m - D(z)\Big)^{-1}\Big[C\ (sI_n - A)^{-1}B + D_0\Big]
\end{aligned}
$$

Remove a factor of $(sI_n - A)^{-1}$

$$
\begin{aligned}
&= \Big(I_m - D(z)\Big)^{-1}\Big[C\ (sI_n - A)^{-1}B + D_0\Big] \\
&\quad + \Big(I_m - D(z)\Big)^{-1} \underbrace{C\ (sI_n - A)^{-1}}_{X} \Big[I_n - \underbrace{B(z)\Big(I_m - D(z)\Big)^{-1}}_{Y} \underbrace{C\ (sI_n - A)^{-1}}_{X}\Big]^{-1} \\
&\quad \times \underbrace{B(z)\Big(I_m - D(z)\Big)^{-1}}_{Y}\Big[C\ (sI_n - A)^{-1}B + D_0\Big]
\end{aligned}
$$

Use Lemma 3.3 (i) to obtain

$$
\begin{aligned}
&= \Big(I_m - D(z)\Big)^{-1}\left\{I_m + \Big[I_m - C\ (sI_n - A)^{-1}B(z)\Big(I_m - D(z)\Big)^{-1}\Big]^{-1} - I_m\right\} \\
&\quad \times \Big[C\ (sI_n - A)^{-1}B + D_0\Big]
\end{aligned}
$$

$$
\begin{aligned}
&= \Big\{I_m - D(z) - C\ (sI_n - A)^{-1}B(z)\Big\}^{-1}\Big[C\ (sI_n - A)^{-1}B + D_0\Big] \\
&= \left\{I_m - \sum_{j=1}^{M} G_j(s) z^{-j}\right\}^{-1} G_0(s) \qquad (251)
\end{aligned}
$$

where

$$G_0(s) = C(sI_n - A)^{-1}B + D_0 \in \mathbb{R}^{m\times\ell}(s) \qquad (252)$$

$$G_j(s) = C(sI_n - A)^{-1}B_{j-1} + D_j \quad \text{for} \quad 1 \le j \le M \in \mathbb{R}^{m\times m}(s) \qquad (253)$$

□

From this form of the transfer function matrix the linear differential non-unit multipass process can be represented as an interconnection of subsystems shown in Figure 3.1.

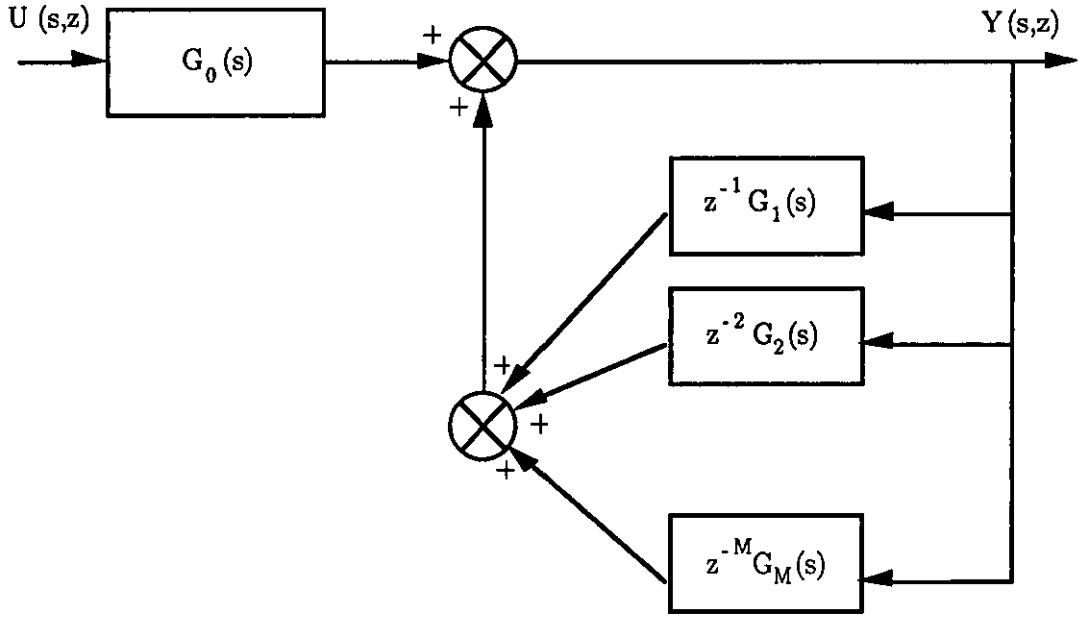

Figure 3.1

Also from the form of the transfer functions of these subsystems, (252), (253) each possesses a state-space realisation, given by

$$\begin{pmatrix} sI - A & B \\ -C & D_0 \end{pmatrix}, \quad \begin{pmatrix} sI - A & B_{j-1} \\ -C & D_j \end{pmatrix} \quad 1 \leq j \leq M$$

In addition to this, from (250) the transfer function matrix $G(s,z)$ admits a polynomial state-space system matrix realisation over $\mathbb{R}(z)[s]$

$$P(s,z) = \begin{pmatrix} sI_n - \bar{A} & \bar{B} \\ -\bar{C} & \bar{D} \end{pmatrix}$$

where

$$\bar{A} = A + B(z)\Big(I_m - D(z)\Big)^{-1} C$$

$$\bar{B} = B + B(z)\Big(I_m - D(z)\Big)^{-1} D_0$$

$$\bar{C} = \Big(I_m - D(z)\Big)^{-1} C$$

$$\bar{D} = \Big(I_m - D(z)\Big)^{-1} D_0$$

3.5.2 Unit Memory Multipass Processes

If only the previous pass has an influence on the current pass, i.e. $M = 1$, the process is described as a linear differential unit memory multipass process. Thus the transfer function matrix is given by

$$G(s,z) = \left[I_m - z^{-1}G_1(s)\right]^{-1} G_0(s) \tag{254}$$

where

$$G_0(s) = C(sI_n - A)^{-1}B + D_0 \in \mathbb{R}^{m\times\ell}(s)$$
$$G_1(s) = C(sI_n - A)^{-1}B_0 + D_1 \in \mathbb{R}^{m\times m}(s)$$

Now write

$$G_1(s) = \bar{N}(s)\bar{D}^{-1}(s) \in \mathbb{R}^{m\times m}(s)$$
$$G_0(s) = D^{-1}(s)N(s) \in \mathbb{R}^{m\times\ell}(s)$$

where $\bar{N}$, $\bar{D}$ are (zero) right coprime and N, D are (zero) left coprime (the coprimeness type is described as 'zero' because the matrices are polynomial in one indeterminate and therefore all three definitions of coprimeness are equivalent, see Theorem 1.3). Then $G(s,z)$ may be written

$$\begin{aligned} G(s,z) &= \bar{D}(s)\left[\bar{D}(s) - z^{-1}\bar{N}(s)\right]^{-1} D^{-1}(s)N(s) \\ &= \bar{D}(s)\left[D(s)\left(\bar{D}(s) - z^{-1}\bar{N}(s)\right)\right]^{-1} N(s) \end{aligned}$$

Thus the system matrix over $\mathbb{R}[s,z^{-1}]$ is a $(m+m)\times(m+\ell)$ polynomial matrix given by

$$P(s,z) = \begin{pmatrix} D(s)\left[\bar{D}(s) - z^{-1}\bar{N}(s)\right] & N(s) \\ -\bar{D}(s) & 0 \end{pmatrix} \tag{255}$$

In this case the zeros of the system matrix, $P(s,z)$, can be characterised: completely for the output matrix pair $D(s)\left[\bar{D}(s) - z^{-1}\bar{N}(s)\right]$, $\bar{D}(s)$ and partially for the input matrix pair $D(s)\left[\bar{D}(s) - z^{-1}\bar{N}(s)\right]$, $N(s)$. Also conditions may be derived such that the system matrix $P(s,z^{-1})$ is least order in the sense of Definition 3.5, i.e. the output pair are zero (right) coprime and the input pair are minor (left) coprime. This is the subject of what follows.

Firstly consider the output zeros, which are formally defined as:

Definition 3.10: (*Output Zeros*)
The output zeros of a system matrix

$$P(s,z) = \begin{pmatrix} T(s,z) & U(s,z) \\ -V(s,z) & W(s,z) \end{pmatrix}$$

are defined to be the values $(s,z) \in \mathbb{C}\times\mathbb{C}$ that causes the following matrix to not have full rank

$$\begin{pmatrix} T(s,z) \\ -V(s,z) \end{pmatrix}$$

□

Thus for the unit memory multipass process the output zeros are defined as the values of $(s,z) \in \mathbb{C}\times\mathbb{C}$ such that

$$\begin{pmatrix} D(s)\left[\bar{D}(s) - z^{-1}\bar{N}(s)\right] \\ -\bar{D}(s) \end{pmatrix} \tag{256}$$

does not have full rank. Thus the following result may be derived.

Lemma 3.4: *The matrix (256) does not have full rank whenever*

$$\begin{pmatrix} -z^{-1}D(s)\bar{N}(s) \\ \bar{D}(s) \end{pmatrix}$$

does not have full rank.

Proof: Pre-multiply (255) by the unimodular matrix $\begin{pmatrix} I_m & D(s) \\ 0 & I_m \end{pmatrix}$ to obtain

$$\begin{pmatrix} I_m & D(s) \\ 0 & I_m \end{pmatrix} \begin{pmatrix} D(s)\left[\bar{D}(s) - z^{-1}\bar{N}(s)\right] \\ -\bar{D}(s) \end{pmatrix} = \begin{pmatrix} -z^{-1}D(s)\bar{N}(s) \\ -\bar{D}(s) \end{pmatrix}$$

and the result is obtained. □

Thus from this result for any z^{-1} the 's'-part of the zeros for $z^{-1} \neq 0$ of (255) are the zeros of

$$\begin{pmatrix} D(s)\bar{N}(s) \\ \bar{D}(s) \end{pmatrix}. \tag{257}$$

Let $Q(s)$ be a greatest common right divisor of $D(s)\bar{N}(s)$, $\bar{D}(s)$ then

$$\begin{pmatrix} D(s)\bar{N}(s) \\ \bar{D}(s) \end{pmatrix} = \begin{pmatrix} \tilde{N}(s) \\ \tilde{D}(s) \end{pmatrix} Q(s) \tag{258}$$

where $\tilde{N}(s)$, $\tilde{D}(s)$ are $m \times m$ polynomial matrices, i.e. the zeros of (257) correspond to the values of s for which $|Q(s)| = 0$. Thus these zeros are removable and are attributable to the realisation and not to the transfer function matrix.

The only other type of zeros of (255) are those such that $z^{-1} = 0$. Thus the non-removable output zeros of (255) are given by the set

$$\Omega_1 = \{(s_0, 0) : |\tilde{D}(s_0) = 0|\} \tag{259}$$

where $\tilde{D}(s)$ is defined by (258). These are the non-essential singularities of the second kind of (255) since (255) does not have full rank and

$$|D(s)||\tilde{D}(s)Q(s) - z^{-1}\tilde{N}(s)Q(s)| = 0.$$

It is now possible to state a condition for the unit memory multipass process to possess a least order realisation, in the sense of Definition 3.5.

Theorem 3.22: *A least order realisation of (254) is derivable from (256) using 1-D techniques if $\tilde{D}(s)$ of (258) is a unimodular matrix.*

Proof: A system matrix realisation is least order if the input pair are minor (left) coprime and the output pair are zero (right) coprime. A system matrix realisation in which both the input and output pairs are minor coprime is achieved by extracting

the greatest common divisors of the input and output pair, i.e. for the unit memory multipass process (254) let $Q(s)$ be a greatest common (right) divisor of the output pair so that

$$P(s,z) = \begin{pmatrix} \tilde{D}(s) - z^{-1}\tilde{N}(s) & N(s) \\ -\tilde{D}(s) & 0 \end{pmatrix} \begin{pmatrix} Q(s) & 0 \\ 0 & I \end{pmatrix}$$

and let $R(s)$ be a greatest common divisor of $\tilde{D}(s) - z^{-1}\tilde{N}(s)$, $\tilde{N}(s)$, i.e.

$$\begin{pmatrix} \tilde{D}(s) - z^{-1}\tilde{N}(s) & N(s) \end{pmatrix} = R(s) \begin{pmatrix} \tilde{N}'(s,z) & N' \end{pmatrix}$$

Thus

$$P(s,z) = \begin{pmatrix} R(s) & 0 \\ 0 & I \end{pmatrix} \underbrace{\begin{pmatrix} \tilde{N}'(s,z) & N' \\ -\tilde{D}(s) & 0 \end{pmatrix}}_{P'(s,z)} \begin{pmatrix} Q(s) & 0 \\ 0 & I \end{pmatrix}.$$

If $\tilde{D}(s)$ is a unimodular matrix the output pair possesses a constant high-order minor therefore $\tilde{D}(s)$, $\tilde{N}'(s,z)$ are zero (right) coprime. □

However, if $\tilde{D}(s)$ is not unimodular then it may be possible to apply constant output feedback around $G_1(s)$ to remove these non-essential singularities of the second kind, i.e. use feedback around $G_1(s)$ to create a unimodular $\tilde{D}(s)$ so that the new system matrix is least order. Effectively applying feedback to achieve a least order system matrix relocates the non-essential singularities to infinity [68].

Due to the structure of the system matrix, $P(s,z)$, it is not possible to explicitly characterise the input zeros of the unit memory process. However, if $m \leq \ell$ then it is possible give a set of values for which the zeros are a subset, thus narrowing the possible candidates to a finite set. This condition is not unrealistic since in practice $m \leq \ell$ means that there are not more outputs than inputs, which can always be made true (if there are more outputs than inputs then it can be shown that some of the outputs are dependent on others, i.e. there are only the same number of independent outputs as there are inputs).

Definition 3.11: (*Input Zeros*)
The input zeros of a system matrix,

$$P(s,z) = \begin{pmatrix} T(s,z) & U(s,z) \\ -V(s,z) & W(s,z) \end{pmatrix}$$

are defined to be the values $(s,z) \in \mathbb{C} \times \mathbb{C}$ that causes the following matrix to lose rank

$$\begin{pmatrix} T(s,z) & U(s,z) \end{pmatrix}$$

□

Clearly a necessary condition for (259) to not possess any zeros is that $N(s)$ has full rank for all $s \in \mathbb{C}$. A necessary condition for (s, z) being a zero of (259) is that it is simultaneously a zero of the two constituent matrices $D(s)[\bar{D}(s) - z^{-1}\bar{N}(s)]$ and $N(s)$ but this does not guarantee it being a zero of the compound matrix (259). Thus a set of candidate zeros for (259) may be derived by characterising the set of values of (s, z) that are simultaneous zeros of $D(s)[\bar{D}(s) - z^{-1}\bar{N}(s)]$ and $N(s)$.

Theorem 3.23: *Suppose that $m \leq \ell$. Let Ω_{input} be the set of input zeros of the transfer function for the unit memory process described above. Then*

$$\Omega_{\text{input}} \subseteq \Omega_a \cup \bigcup_{s_i \in \Lambda_b} \Omega_{s_i}$$

where

$$\Omega_{s_i} = \left\{(s_i, z_0^{-1}) \mid \ s_i \in \Lambda_b, \quad |\bar{D}(s_i) - z_0^{-1}\bar{N}(s_i)| = 0\right\}$$

$$\Omega_a = \left\{(s_0, z^{-1}) \mid N(s_0) \in \Lambda_a\right\}$$

$$\Lambda_a = \{s_0 \in \mathbb{C} \mid N(s_0) \text{ has less than full rank and } |D(s_0)| = 0\}$$

$$\Lambda_b = \{s_0 \in \mathbb{C} \mid N(s_0) \text{ has less than full rank and } |D(s_0)| \neq 0\}$$

Proof: The input zeros of the unit memory process are defined to be the values of $(s_0, z_0^{-1}) \in \mathbb{C} \times \mathbb{C}$ such that

$$\left[\, D(s_0)\left[\bar{D}(s_0) - z_0^{-1}\bar{N}(s_0)\right] \quad N(s_0) \,\right] \tag{260}$$

does not have full rank. This occurs at the values for which $N(s_0)$ does not have full rank, i.e. the values of the set Λ_N

$$\Lambda_N = \{s_0 \in \mathbb{C} \mid N(s_0) \text{ has less than full rank}\}.$$

Additionally for (260) to lose rank $|D(s_0)||\bar{D}(s_0) - z^{-1}\bar{N}(s_0)| = 0$ for $s_0 \in \Lambda_N$ thus

$$|D(s_0)| = 0 \qquad \text{or} \qquad |\bar{D}(s_i) - z_0^{-1}\bar{N}(s_i)| = 0$$

Thus the sets of values which are necessary for (260) to lose rank are

$$\Omega_{s_i} = \left\{(s_i, z_0^{-1}) \mid \ s_i \in \Lambda_b, \quad |\bar{D}(s_i) - z_0^{-1}\bar{N}(s_i)| = 0\right\}$$

$$\Omega_a = \left\{(s_0, z^{-1}) \mid N(s_0) \in \Lambda_a\right\}$$

where

$$\Lambda_a = \{s_0 \in \mathbb{C} \mid N(s_0) \text{ has less than full rank and } |D(s_0)| = 0\}$$

$$\Lambda_b = \{s_0 \in \mathbb{C} \mid N(s_0) \text{ has less than full rank and } |D(s_0)| \neq 0\}$$

□

From the above discussion it has been seen that the output zeros of a system matrix representation of the transfer function $G(s, z)$ of the unit memory multipass process can be characterised more fully than the input zeros. Thus existence conditions may be given for a least order system matrix, in the sense of Definition 3.5.

3.6 Conclusions

In this chapter the questions of equivalence of polynomial matrices have been individually addressed for 1-D systems (Section 3.2), 2-D systems (Section 3.3) and n-D systems (Section 3.4). The first of these concentrated on the finite and infinite zero structure of system matrices. The relation of full equivalence was proved to preserve both the finite and infinite zero structure of two system matrices. The definition of full equivalence contained a restriction on the McMillan degrees of certain matrices which has recently appeared in the work of Zhang [61] to guarantee the absence of infinite zeros.

The results for 2-D systems considered two types of relations, termed zero equivalence and minor equivalence; these were then evaluated in terms of invariant properties. The former was seen to be a true equivalence relation that preserves the transfer function matrix, the invariant polynomials of the T, U and T, V pair, the invariant polynomials of the system matrices and also the invariant polynomials of the T-blocks. Additionally, the coprimeness of the T, U and T, V pairs is also invariant under zero equivalence. Minor equivalence does not enjoy such a wealth of invariant properties forgoing the invariance of the coprimeness on the T, U and T, V pairs for a weaker condition on the polynomial transformation.

The results obtained for 2-D systems are then considered in an n-D framework. Now the number of definitions for coprime matrices is increased from two to three. Thus three definitions of equivalence may be defined. In particular, least order matrices were defined by factor coprimeness of the T, U pair and zero coprimeness of the T, V pair. If two of these least order matrices possess a polynomial relationship, that preserves the transfer function matrix, then the relationship was seen to be one of zero system equivalence, moreover, if the T, V pair were minor (right) coprime the underlying transformation is one of minor system equivalence.

Finally a specific type of 2-D system is considered in terms of the zeros possessed by that system and conditions are given for the existence of a least order system matrix realisation.

PART TWO

A MAPLE Program for the Symbolic Computation of the Greatest Common Divisor of Polynomial Matrices in Two Indeterminates

---

Preface

The second part of this thesis is devoted to the symbolic computation of the greatest common divisor (GCD) of polynomial matrices in two indeterminates. The motivation for such a procedure is evident by considering the complexity of performing the coprime MFD Algorithm (Algorithm 2.1) for seemingly simple examples, see Example 4.1. The program is based on certain theoretical ideas suggested in the literature but the practicalities of these ideas have not been considered. It is the intention of this program is to transform these ideas into a working automatic procedure.

The calculation of a 2-D GCD is more complex than its single indeterminate counterpart due to the nature of the ring structure to which the elements of the matrices

belong, i.e. $F[z_1, z_2]$. The procedure to determine the GCD of single indeterminate polynomial matrices is based on elementary operations to create, in effect, factors in the rows or columns, [13], [77]; this is possible because the underlying ring structure is Euclidean. A similar procedure fails for two indeterminate polynomial matrices because a division algorithm does not exist. A further hindrance is provided by some unimodular matrices not being formed as a product of elementary matrices. Thus the techniques used in the 1-D case are insufficient to calculate 2-D matrix GCDs.

The algorithms are implemented using the MAPLE symbolic manipulation package. This software has the ability to leave the elements of a computation unevaluated and to perform algebraic simplification on such expressions. This feature makes MAPLE particularly suitable for the representation of data structures involving polynomials. Another advantage of a symbolic language is the similarity of notation to the abstract algebraic representation. Thus the resulting code closely resembles the theoretical algorithms. This is further enhanced by the procedural nature of the language. The form of a procedural language, such as that possessed by MAPLE, also allows programs to be built up in small stand-alone segments. These segments may be tested before being linked in the main controlling procedure, thus potential problems may be identified at an early stage.

The documentation is ordered as follows:

Chapter 4 introduces the theoretical background to the algorithms and discusses the modifications made to the original ideas required for automatic implementation.

Chapter 5 introduces the MAPLE symbolic manipulation software package by defining the features used in the program documentation.

Chapter 6 documents the MAPLE code that carries out the calculation of the greatest common divisor of polynomial matrices in two indeterminates.

Chapter 7 presents the examples used to test the program and also some points that arise from these examples.

Chapter 4

Mathematical Basis

4.1 Introduction

The object of this chapter is to set out the theoretical ideas used to formulate the procedure to compute the greatest common divisor of 2-D polynomial matrices. These ideas give rise to an algorithm which is based on two canonical forms for 2-D polynomial matrices, namely the 2-D primitive form and the 2-D Hermite form [33], [1]. The first is a generalisation of the primitive form for a 2-D scalar polynomial and the second is a generalisation of the Hermite form for 1-D polynomial matrices.

The algorithmic basis for the above canonical forms are elementary operations over a Euclidean ring; thus the notion of a division algorithm is required. This is achieved

by performing computations over a generalised ring, namely $F(z_1)[z_2]$ or $F(z_2)[z_1]$, resulting in expressions with rational terms in one of the indeterminates. The polynomial situation is recovered by multiplying the expression by a polynomial to cancel the rational parts; this is known as renormalisation. The algorithm is then said to have been performed over $F[z_1][z_2]$ or $F[z_2][z_1]$, respectively.

The algorithms in this chapter are defined over the ring $F[z_1][z_2]$ and, wherever appropriate, the process of division algorithm and renormalisation are combined to form the pseudo-division algorithm; this avoids rational polynomials being formed during the computation.

The main theoretical contribution contained in this chapter is the defintion and algorithmic determinatation of the Hermite form for a rectangular polynomial matrix that does not have full rank. This appears to have been overlooked in the literature and more importantly causes the primitive factorisation algorithm to fail in some instances. One other theoretical contribution is the derivation of a simple coprimeness test based on the 2-D Hermite form.

4.2 Program Motivation

The motivation for this program lies with the complexity of computing a greatest common divisor. The absence of a division algorithm in the ring $F[z_1, z_2]$ further complicates the procedure as the techniques used for computing the greatest common divisor of 1-D polynomial matrices are not sufficient in the 2-D case. In particular, in $F[z_1, z_2]$ there exist unimodular matrices that can not be expressed as a product of elementary matrices [25]. Therefore some polynomial factors can not be computed using elementary operations alone over $F[z_1, z_2]$, i.e.

(a) Multiplication of any row by an element of F.

(b) Addition of any row with any other row, multiplied by an element of $F[z_1, z_2]$.

(c) Permutation of any two rows.

For these reasons it is impractical to compute a GCD by hand, even for matrices with a small number of rows and columns, using a symbolic manipulator interactively. To illustrate these difficulties consider the following example.

Example 4.1: Compute the greatest common right divisor of the matrix $A(z_1, z_2)$ formed by

$$A(z_1,z_2) = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}}_{\bar{A}} \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & z_1+1 \end{pmatrix}}_{D} \underbrace{\begin{pmatrix} 1-z_1z_2 & -z_2^2 \\ z_1^2 & 1+z_1z_2 \end{pmatrix}}_{U}$$

$$= \begin{pmatrix} 1-z_1z_2 & -z_2^2 \\ z_1^2(1+z_1) & (1+z_1z_2)(1+z_1) \\ 1-z_1z_2 & -z_2^2 \end{pmatrix}$$

By considering the way $A(z_1, z_2)$ has been formed, the greatest common right divisor has determinant $1 + z_1$. To mimic the procedure used for 1-D polynomial matrices the elementary operations over $F[z_1, z_2]$ are used to form the factor $1 + z_1$ common to all elements in one column. The purpose of this example is to show that this is not possible.

Firstly, consider column operations on $A(z_1, z_2)$. It is required to form a factor $z_1 + 1$ in one of the elements of the first row, since the second row will always possess this factor. This may be attempted in two ways:

1. Add a polynomial multiple, say $a(z_1, z_2)$, of column 1 to column 2. Therefore the (1,1) element is

$$a(z_1, z_2)(1 - z_1z_2) - z_2^2 = b(z_1, z_2)(1 + z_1)$$

for some polynomial $b(z_1, z_2)$. Rearranging this equation gives

$$a(z_1, z_2)(1 - z_1 z_2) - b(z_1, z_2)(1 + z_1) = z_2^2$$

If $z_1 = -1$ and $z_2 = -1$ then

$$a(z_1, z_2) \times 0 - b(z_1, z_2) \times 0 = 1^2.$$

Therefore there exists no elementary column operation of this type to give the required form.

2. Add a polynomial multiple, say $a(z_1, z_2)$, of column 2 to column 1. Therefore the (1,2) element is

$$a(z_1, z_2)z_2^2 + b(z_1, z_2)(1 + z_1) = 1 - z_1 z_2$$

for some polynomial $b(z_1, z_2)$. Put $z_1 = -1$ and $z_2 = 0$ then

$$a(z_1, z_2) \times 0 + b(z_1, z_2) \times 0 = 1.$$

Therefore there exists no elementary column operation of this type to give the required form.

Secondly, consider elementary row operations to form a factor $1 + z_1$ in the first row of $A(z_1, z_2)$. Again this may be performed in two ways.

1. Add a polynomial multiple, say $a(z_1, z_2)$, of row 1 to row 2. Therefore the (1,1) element is

$$a(z_1, z_2)(1 - z_1 z_2) + b(z_1, z_2)(1 + z_1) = (1 + z_1)z_1^2$$

for some $b(z_1, z_2)$. Put $z_1 = -1$ then

$$a(z_1, z_2)(1 + z_2) = 0$$

but if $a(z_1, z_2) = 0$ the elementary operation does not alter the elements of the matrix. Therefore there exists no elementary row operation of this type to give the required form.

2. Add a polynomial multiple, say $a(z_1, z_2)$, of row 2 to row 1. Therefore the (1,2) element is

$$a(z_1, z_2)z_1^2(1 + z_1) + b(z_1, z_2)(1 + z_1) = 1 - z_1 z_2$$

for some $b(z_1, z_2)$. Put $z_1 = -1$ and then

$$a(z_1, z_2) \times 0 + b(z_1, z_2) \times 0 = 1 + z_2.$$

Therefore there exists no elementary row operation of this type to give the required form.

From the above discussion it has been seen that the polynomial matrix right factor can not be extracted by elementary operations alone. This is due to the matrix $U(z_1, z_2)$, which is termed a secondary matrix, and the fact that $F[z_1, z_2]$ does not possess a division algorithm. □

From the above example it is obvious that not all polynomial matrix factors can be computed using elementary operations over the ring $F[z_1, z_2]$ and secondary operations are required (this is achieved by pre- or post-multiplication by matrices similar to $U(z_1, z_2)$, which can not be expressed as a product of elementary matrices). The formation of these secondary operations is not obvious and therefore an alternative procedure is required. The solution is provided by Part Two of this thesis together with the code necessary to compute the greatest common divisor using the symbolic manipulator MAPLE.

4.3 Formal Definitions

Before discussing the actual algorithms it is necessary to define formally the canonical forms used and the precise definition of the problem.

Definition 4.1: (*Greatest Common Right Divisor*)
Let $A_{p\times m}(z_1, z_2)$ and $B_{q\times m}(z_1, z_2)$ be two polynomial matrices. Then $Q_{m\times m}(z_1, z_2)$ is said to be a greatest common right divisor (GCRD) of $A(z_1, z_2)$, $B(z_1, z_2)$ if $Q(z_1, z_2)$ is a right divisor of $A(z_1, z_2)$, $B(z_1, z_2)$ and any other right divisor $R(z_1, z_2)$ is also a right divisor of $Q(z_1, z_2)$, i.e.

$$Q(z_1, z_2) = E(z_1, z_2)R(z_1, z_2)$$

where $E(z_1, z_2)$ is a polynomial matrix. □

Note: A greatest common left divisor (GCLD) can be defined similarly by transposition. Also a greatest common left or right divisor is only unique modulo a right or left unimodular matrix, respectively. Therefore if $Q_1(z_1, z_2)$ and $Q_2(z_1, z_2)$ are two greatest common right divisors of the same polynomial matrix there exists a unimodular matrix $U(z_1, z_2)$ such that

$$Q_1(z_1, z_2) = U(z_1, z_2)Q_2(z_1, z_2). \tag{261}$$

Suppose that $Q(z_1, z_2)$ is a GCRD of the two polynomial matrices $A(z_1, z_2)$ and $B(z_1, z_2)$, as defined in the above definition, then

$$\begin{pmatrix} A(z_1, z_2) \\ B(z_1, z_2) \end{pmatrix} = \begin{pmatrix} \bar{A}(z_1, z_2) \\ \bar{B}(z_1, z_2) \end{pmatrix} Q(z_1, z_2) \tag{262}$$

where $\bar{A}(z_1, z_2)$, $\bar{B}(z_1, z_2)$ are minor (right) coprime polynomial matrices.

The precise definition of the problem solved by the documented program that follows is:

> *Given two polynomial matrices, $A(z_1, z_2)$ and $B(z_1, z_2)$, in two indeterminates, z_1 and z_2. Determine the greatest common divisor and also the coprime pair of matrices that result from extracting this greatest common divisor.*

The basic operations governing the formation of the canonical forms are the elementary row and column operations performed over a generalised ring. This allows the definition of a division algorithm and consequently the definition of a pseudo-division algorithm:

Definition 4.2: (*Pseudo-division Algorithm over* $F[z_1][z_2]$)
Suppose that $a(z_1, z_2)$ and $b(z_1, z_2)$ are polynomials in the two indeterminates z_1 and z_2, the pseudo-remainder, $r(z_1, z_2)$, and pseudo-quotient, $q(z_1, z_2)$, over $F[z_1][z_2]$ are given by

$$m(z_1)a(z_1, z_2) = q(z_1, z_2)b(z_1, z_2) + r(z_1, z_2)$$

where $\deg_{z_2}(b) > \deg_{z_2}(r)$ and $m(z_1)$, the pseudo-multiplier, is a polynomial in z_1. This form is not unique but can be made so modulo a multiplicative constant, by insisting that $m(z_1)$ is of lowest degree possible such that $q(z_1, z_2)$ and $r(z_1, z_2)$ are polynomial. □

Definition 4.3: (*Elementary Row Operations over* $F[z_1][z_2]$)

(a) Multiplication of any row by an element of $F[z_1]$.

(b) Addition of any row with any other row, multiplied by an element of $F[z_1, z_2]$.

(c) Permutation of any two rows. □

The elementary column operations may be similarly formulated.

The matrices effecting the elementary row operations are unimodular in $F[z_1][z_2]$, i.e. the determinants are elements of $F[z_1]$. These matrices are termed elementary matrices over $F[z_1][z_2]$ and are formally defined in Definition 4.4.

Definition 4.4: (*Elementary Matrices over* $F[z_1][z_2]$)
The elementary matrices that effect the elementary row operations in Definition 4.3 on a $m \times n$ polynomial matrix are given by pre-multiplication by:

(a) Multiplication of row i by $a \in F[z_1]$

$$E_1 = \begin{array}{c} \\ 1 \\ i \\ \\ m \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & & i & & m \end{array} \\ \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & a & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \end{array} \tag{263}$$

where $|E_1| = a \in F[z_1]$.

(b) Addition to row j with $b \in F[z_1, z_2]$ times row i.

$$E_2 = \begin{array}{c} \\ 1 \\ \\ i \\ \\ j \\ \\ m \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & & i & & j & & m \end{array} \\ \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & b & \cdots & 1 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & o & \cdots & 1 \end{pmatrix} \end{array} \tag{264}$$

where $|E_2| = 1 \in F[z_1]$.

(c) Permutation of rows i and j.

$$E_3 = \begin{array}{c} \\ 1 \\ \\ i \\ \\ j \\ \\ m \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & & i & & j & & m \end{array} \\ \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & \cdots & 1 & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & o & \cdots & 1 \end{pmatrix} \end{array} \tag{265}$$

where $|E_3| = 1 \in F[z_1]$.

The elementary matrices effecting the column operations are $n \times n$ post-multiplicative matrices of the form E_1, E_2, E_3.

Definition 4.5: (*2-D Hermite Form over* $F[z_1][z_2]$)
The 2-D upper triangular Hermite form of the polynomial matrix $A(z_1, z_2)$, of size $m \times n$, is

$$H(z_1, z_2) = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ 0 & h_{22} & \cdots & h_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & h_{nn} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \quad \text{if} \quad m \geq n \tag{266}$$

$$H(z_1, z_2) = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1m} & \cdots & h_{1n} \\ 0 & h_{22} & \cdots & h_{2m} & \cdots & h_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & h_{mm} & \cdots & h_{mn} \end{pmatrix} \quad \text{if} \quad m < n \tag{267}$$

where the column degree condition

$$\deg_{z_2}(h_{ii}) > \deg_{z_2}(h_{ji}) \quad \text{for} \quad j \neq i$$

is satisfied. □

The above 2-D Hermite form may be formed solely by elementary row operations over $F[z_1][z_2]$, so that there exists a pre-multiplicative polynomial matrix $U_{m\times m}(z_1, z_2)$ such that $|U(z_1, z_2)| \in F[z_1]$; this can be seen by considering the 2-D Hermite algorithm later, Algorithm 4.1. An equivalent definition may be formed by using elementary column operations to derive the 2-D lower triangular Hermite form:

$$H(z_1, z_2) = \begin{pmatrix} h_{11} & 0 & \cdots & 0 \\ h_{21} & h_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ h_{n1} & h_{n2} & \cdots & h_{nn} \\ \vdots & \vdots & \vdots & \vdots \\ h_{m1} & h_{m2} & \cdots & h_{mn} \end{pmatrix} \quad \text{if} \quad m \geq n \tag{268}$$

$$H(z_1, z_2) = \begin{pmatrix} h_{11} & 0 & \cdots & 0 & 0 & \cdots & 0 \\ h_{21} & h_{22} & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ h_{m1} & h_{m2} & \cdots & h_{mm} & 0 & \cdots & 0 \end{pmatrix} \quad \text{if} \quad m < n \tag{269}$$

where the row degree condition

$$\deg_{z_2}(h_{ii}) > \deg_{z_2}(h_{ij}) \quad \text{for} \quad j \neq i \tag{270}$$

is satisfied. Thus there exists a post-multiplicative polynomial matrix $V_{n\times n}(z_1, z_2)$ with $|V(z_1, z_2)| \in F[z_1]$ that derives the 2-D lower triangular Hermite form.

Definition 4.6: (*Primitive Matrix over* $F[z_1][z_2]$)
Let $A(z_1, z_2)$ be a full rank $m \times n$ polynomial matrix. Then $A(z_1, z_2)$ is said to be a primitive matrix with respect to $F[z_1][z_2]$ if the $r \times r$ minors of $A(z_1, z_2)$ have no common divisors that are polynomial in z_1 only, where $r = \min(m, n)$. □

Note: When F is algebraically closed this definition reduces to the statement that $A(z_1, z_2)$ is said to be primitive if $A(z_1^{(0)}, z_2)$ has full rank for all fixed $z_1^{(0)} \in F$. These values of z_1 are said to be the primitive roots of the matrix $A(z_1, z_2)$ over $F[z_1][z_2]$.

In general matrices are not primitive with respect to a particular ring but contain primitive roots. However, it is always possible to factorise a non-primitive matrix into

a primitive part and a primitive factor; the theoretical proof is contained in [33]. To define this more explicitly consider the polynomial matrix $A_{m\times n}(z_1, z_2)$ with $m \leq n$ and let the content of the polynomials formed by the $m \times m$ minors be $c(z_1, z_2)$. This polynomial may be factored into two parts, one solely in z_1 and the other by the remainder, i.e. $c(z_1, z_2) = \bar{c}(z_1)c^*(z_1, z_2)$. The primitive roots are defined by the zeros of $\bar{c}(z_1)$ and the primitive factor is defined by the matrix $\bar{A}(z_1, z_2)$ such that

$$|\bar{A}(z_1, z_2)| = \bar{c}(z_1)$$

and

$$A(z_1, z_2) = \bar{A}(z_1, z_2)A^*(z_1, z_2). \tag{271}$$

By definition the matrix is $A^*(z_1, z_2)$ is primitive over $F[z_1][z_2]$.

Note: If $m \geq n$ the primitive matrix and factor are defined by

$$A(z_1, z_2) = A^*(z_1, z_2)\bar{A}(z_1, z_2) \tag{272}$$

where $|\bar{A}(z_1, z_2)|$ is defined to be the polynomial content in z_1 of the $n \times n$ minors of $A(z_1, z_2)$. Thus if $A(z_1, z_2)$ is a square polynomial matrix the primitive factor may be extracted on the left or on the right.

4.4 Theoretical Algorithms

4.4.1 Introduction

The basis of the algorithms are derived from theoretical ideas put forward by Morf *et al* [33] and contained in Lévy [34]. However, the form of the algorithms presented in the above two works do not lend themselves to direct implementation. Certain modifications have been made to facilitate the automation. It is assumed in the literature that the well known process of forming the Hermite form of a full rank matrix can be directly applied to form the Hermite form of a matrix that does not possess full rank. However, this is not the case. In fact the formal definition and algorithmic determination are not trivial generalisations of the full rank case. These are given in Section 4.4.3.

As discussed in the introduction to this chapter it is necessary to favour one of the indeterminates when performing the algorithms. Clearly here there is a choice between the two indeterminates z_1 and z_2. It may seem that the choice is quite arbitrary but the physical interpretation of the problem may suggest that one indeterminate is more appropriate than the other. Even in the abstract mathematical context considered here, there is a sometimes a 'better' choice. This is demonstrated by the amount of time taken to compute the test examples in Chapter 7, although it is almost impossible to determine the 'better' choice by inspection. In general the definition of the algorithms will be given over the ring $F[z_1][z_2]$, i.e. polynomials in z_2 with coefficients in $F[z_1]$ where F denotes a field; usually this will be $\mathbb{R}$ but at certain points in the algorithms it is necessary to use the algebraic closure of this ring, $\mathbb{C}$, e.g. in the calculation of the primitive roots.

4.4.2 2-D Hermite Form

The Hermite form is well defined for polynomial matrices in one indeterminate [1] and is considered a standard canonical form for such matrices. The object in this section is to define a similar type of canonical form for 2-D polynomial matrices. The derivation is arrived at by favouring one of the indeterminates and then using the pseudo-division algorithm, Definition 4.2, and the elementary operations, Definition 4.3. Thus when the 2-D Hermite form is calculated over $F[z_1][z_2]$ the degree condition of the pseudo-division algorithm is defined with respect to z_2. Clearly the 2-D Hermite form of a polynomial matrix will not necessarily be the same when derived over $F[z_1][z_2]$ and $F[z_2][z_1]$.

Algorithm 4.1: *(2-D Hermite Form over $F[z_1][z_2]$)*
Let $A(z_1, z_2)$ be an $m \times n$ full rank polynomial matrix. The upper triangular 2-D Hermite form is derived using elementary row operations in the following way.

Step 1: The first column of $A(z_1, z_2)$ may be assumed to contain a non-zero element, otherwise $A(z_1, z_2)$ is not full rank. Amongst the non-zero elements of the first column choose one with the lowest degree, in z_2, and by means of row permutations bring it to the (1,1) position to form the matrix $A^{[1]}(z_1, z_2)$ with elements $a_{ij}^{[1]}(z_1, z_2)$.

Step 2: Using the pseudo-division algorithm, Definition 4.2, compute $m_{i1}^{[1]}(z_1)$, $q_{i1}^{[1]}(z_1, z_2)$ and $r_{i1}^{[1]}(z_1, z_2)$, for $i = 2, 3, \ldots, m$, such that

$$m_{i1}^{[1]}(z_1) a_{i1}^{[1]}(z_1, z_2) = a_{11}^{[1]}(z_1, z_2) q_{i1}^{[1]}(z_1, z_2) + r_{i1}^{[1]}(z_1, z_2) \quad \text{for} \quad i = 2, 3, \ldots, m$$

where

$$\deg_{z_2}\left(r_{i1}^{[1]}(z_1, z_2)\right) < \deg_{z_2}\left(a_{i1}^{[1]}(z_1, z_2)\right) \quad \text{for} \quad i = 2, 3, \ldots, m.$$

Step 3: Replace row i by subtracting $m_{i1}^{[1]}(z_1)$ times row i from $q_{i1}^{[1]}(z_1, z_2)$ times row 1 for $i = 2, 3, \ldots, m$, i.e.

$$a_{ij}^{[2]}(z_1, z_2) = m_{i1}^{[1]}(z_1) a_{ij}^{[1]}(z_1, z_2) - q_{i1}^{[1]}(z_1, z_2) a_{1j}^{[1]}(z_1, z_2) \quad \text{for} \quad \begin{cases} i = 2, 3, \ldots, m \\ j = 1, 2, \ldots, n \end{cases}$$

Step 4: Repeat steps 1–3 until all elements below the first are identically zero. Call this matrix $B(z_1, z_2)$. This is achieved in a finite number of steps equal to or less than the degree in z_2 of $a_{11}^{[1]}(z_1, z_2)$.

Step 5: Using the element $b_{22}(z_1, z_2)$ apply Steps 1–4 on rows $3, 4, \ldots, m$ of the second column of $B(z_1, z_2)$. Repeat this procedure on subsequent columns until an upper triangular matrix is formed. The matrix now has the form:

$$\bar{H}(z_1, z_2) = \begin{pmatrix} \bar{h}_{11} & \bar{h}_{12} & \cdots & \bar{h}_{1n} \\ 0 & \bar{h}_{22} & \cdots & \bar{h}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \bar{h}_{nn} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \quad \text{if} \quad m \geq n \tag{273}$$

$$\bar{H}(z_1, z_2) = \begin{pmatrix} \bar{h}_{11} & \bar{h}_{12} & \cdots & \bar{h}_{1m} & \cdots & \bar{h}_{1n} \\ 0 & \bar{h}_{22} & \cdots & \bar{h}_{2m} & \cdots & \bar{h}_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \bar{h}_{mm} & \cdots & \bar{h}_{mn} \end{pmatrix} \quad \text{if} \quad m < n \tag{274}$$

Step 6: If $\bar{h}_{22}$ is non-zero add polynomial multiples in z_1 and z_2 of row 2 to multiples in z_1 of row 1 to reduce the degree, in z_2, of $\bar{h}_{12}(z_1, z_2)$ below that of $\bar{h}_{22}(z_1, z_2)$ (if the degree of $\bar{h}_{22}(z_1, z_2)$ in z_2 is zero then $\bar{h}_{12}(z_1, z_2)$ becomes identically equal to zero). In the same way $\bar{h}_{33}(z_1, z_2)$ can be used to make the degrees, in z_2, of $\bar{h}_{13}(z_1, z_2)$ and $\bar{h}_{23}(z_1, z_2)$ less than that of $\bar{h}_{33}(z_1, z_2)$, without changing the degree in z_2 of the element $\bar{h}_{12}(z_1, z_2)$. By continuing this procedure for all the columns the upper triangular 2-D Hermite form over $F[z_1][z_2]$ is derived (Definition 4.5). □

Note: If the matrix has full rank and the number of rows is greater than the number of columns, all of the elements on the principal diagonal are non-zero, for if $h_{ii}(z_1, z_2) = 0$, for some i, then by the form of (273) every $n \times n$ order minor is zero; contradicting the full rank property.

All the operations employed in forming the above 2-D Hermite form over $F[z_1][z_2]$ can be effected by elementary matrices in $F[z_1][z_2]$ thus the 2-D Hermite form, $H(z_1, z_2)$ defined in Definition 4.5, is related to the original matrix $A(z_1, z_2)$ by an equation of the form

$$U(z_1, z_2)A(z_1, z_2) = H(z_1, z_2) \tag{275}$$

where $|U(z_1, z_2)| \in F[z_1]$.

4.4.3 Modified 1-D Hermite Form

The motivation for a modified 1-D Hermite form is provided by the primitive factorisation algorithm, where the Hermite form of a 1-D matrix with less than full rank is required with the last row identically zero. The Hermite form only ensures that a number of rows, equal to the rank of the matrix, are independent, since $A(z_1)$ and its Hermite form $H(z_1)$ are related by an equation of the form

$$A(z_1) = U(z_1)H(z_1)$$

where $|U| \in F$. Thus the rank of $A(z_1)$ is equal to the rank of $H(z_1)$. However, the form of the Hermite algorithm does not provide information about the location of the independent rows.

The modified Hermite algorithm is formed by ensuring each new row is independent of the previous rows. Therefore the dependent rows occur as the last rows and as a consequence of the Hermite structure these last rows are guaranteed to be identically zero. This is achieved by effectively ignoring a column in which a pivot element can not be found.

The algorithm is stated for 1-D polynomial matrices, which is the form required in the primitive factorisation algorithm; the generalisation to 2-D follows in a similar manner to the generalisation of the Hermite form to the 2-D Hermite form.

Algorithm 4.2: (*Upper Triangular Modified Hermite Algorithm*)
To compute the upper triangular modified Hermite form of a matrix $A(z_1)$ with m rows and n columns:

Step 1: If the first column is non-zero search the non-zero elements for the one with lowest degree in z_1. Then by means of an elementary row permutation bring this element to the (1,1) position called the pivot element. Now use elementary row operations and the pseudo-division algorithm to reduce the degree in z_1 of all elements below the pivot element to be less than the degree of the pivot element. Continue this procedure until all elements below the pivot are zero.

If the first column is zero then ignore this column and move onto the second column. Assign one to a counter called *pivotzeros*, which counts the number of zero pivot elements encountered so far.

Step 2: For column $1 < i \leq n$, search the elements below and including the element in position $i - pivotzeros$ for the one with lowest degree in z_1 and by means of a row permutation bring this row to the pivotal position, i.e. position $(i - pivotzeros, i)$. Now use elementary row operations and the pseudo-division algorithm to reduce the degree in z_1 of all elements of column i in rows $i - pivotzeros + 1$ to m. Continue this procedure until all elements below the pivot position are zero.

If a non-zero pivot element can not be found, i.e. all the elements below and including the pivot element are zero, increase the counter *pivotzeros* by one and continue to the next column.

Step 3: By means of elementary row operations, starting with the first non-zero element in row two, reduce the degree of the element in row one to below that of the element in row two. Continue this procedure for each of the first non-zero elements in each row. □

Note: If the matrix $A_{m\times n}(z_1)$, with $m \geq n$, has full rank the algorithms for the 1-D modified Hermite form and the 1-D Hermite form coincide since the only possible non-zero high-order minor is formed by the product of the elements on the principle diagonal; if this minor is zero the matrix does not possess full rank. Thus the two algorithms deliver the same result, modulo a unimodular matrix. However, the two algorithms are not necessarily equivalent for matrices with more columns than zeros, i.e. $m < n$, since the minor involving the principal diagonal is not the only possible non-zero high-order minor.

Since the upper triangular modified Hermite form is constructed using elementary operations alone there exists a unimodular matrix $U(z_1)$ such that the matrix $A(z_1)$ and its modified Hermite form, $H(z_1)$, are linked by an equation of the form

$$H_{m\times n}(z_1) = U_{m\times m}(z_1)A_{m\times n}(z_1) \tag{276}$$

Therefore $A(z_1)$ and $H(z_1)$ have the same rank.

When discussing the procedure code to form the modified Hermite form, $H(z_1)$ of $A(z_1)$, it is necessary to use the terms pseudo-principal diagonal and quasi-principal diagonal these are defined as:

Definition 4.7: (*Pseudo-principal Diagonal*)
The pseudo-principal diagonal of the upper triangular modified Hermite form, $H_{m\times n}$, contains $r = \text{rank } A$ elements and is defined as the elements:

(i) The first element is the first non-zero element in the first row;

(ii) The $i+1$th element, for $i = 1, \ldots, \min(m,n) - 1$ is given by the element in position $(i+1, j+1)$ where the element in position (i,j) is the first non-zero element in row i. □

Definition 4.8: (*Quasi-principal Diagonal*)
The quasi-principal diagonal of the upper triangular modified Hermite form, $H_{m\times n}$, contains $r = \text{rank } (A)$ elements and is defined as the elements occupying the first non-zero position in each row. □

4.4.4 2-D Primitive Factorisation

The algorithm to factorise a non-primitive matrix into a primitive part and a primitive factor, as defined by Definition 4.6, is derived by a procedure similar to that provided by the proof to the primitive factorisation theorem [34]. In fact this proof, which is also given in [33], fails for a certain class of matrices. However, by the procedure developed here and the proof given in [34] the theorem holds for all matrices. The justification is provided in Section 4.4.

Theorem 4.1: (*Left Primitive Factorisation Theorem over* $F[z_1][z_2]$)
Let $R(z_1, z_2)$ *be a given* $m \times n$ *full rank 2-D polynomial matrix, where* $m \leq n$, *then there exists a unique* $\bar{R}(z_1, z_2)$ *(modulo a right unimodular matrix) and a unique* $R^*(z_1, z_2)$ *(modulo a left unimodular matrix) with*

$$|\bar{R}(z_1, z_2)| = \bar{r}(z_1)$$

where $\bar{r}(z_1)$ is the content in z_1 of the greatest common divisor of the high-order minors of $R(z_1, z_2)$ and $R^*(z_1, z_2)$ is primitive with respect to $F[z_1][z_2]$ such that

$$R(z_1, z_2) = \bar{R}(z_1, z_2)R^*(z_1, z_2).$$

$\bar{R}(z_1, z_2)$ is called the primitive factor of $R(z_1, z_2)$ and $R^*(z_1, z_2)$ the primitive matrix of $R(z_1, z_2)$. □

Note: The content in z_1 of a polynomial $A(z_1, z_2)$ is the gcd of $\alpha_0(z_1), \alpha_1(z_1), \ldots, \alpha_k(z_1), \ldots$ where

$$a(z_1, z_2) = \alpha_0(z_1) + \alpha_1(z_1)z_2 + \cdots + \alpha_k(z_1)z_2^k + \cdots$$

or equivalently the factors of $a(z_1, z_2)$ solely in z_1.

Algorithm 4.3: (*Left Primitive Factorisation over $F[z_1][z_2]$*)
Let $R_{m\times n}(z_1, z_2)$ be a full rank polynomial matrix. The left primitive factorisation is given by the following three steps.

Step 1: Find all the primitive roots $z_1^{[i]}$, which make $R(z_1^{[i]}, z_2)$ not full rank. This is achieved by factorising the greatest common divisor, $g(z_1, z_2)$, of the high-order minors of $R(z_1, z_2)$ into $\bar{g}(z_1)$ and $g^*(z_1, z_2)$ such that $\bar{g}(z_1)$ contains all the factors solely in z_1 and

$$g(z_1, z_2) = \bar{g}(z_1)g^*(z_1, z_2)$$

i.e. $\bar{g}(z_1)$ is the content in z_1 of $g(z_1, z_2)$. The primitive roots are given by the solution to the equation

$$\bar{g}(z_1) = 0.$$

Step 2: Since $R(z_1^{[i]}, z_2)$ has not full rank, the (upper triangular) modified Hermite form of $R(z_1^{[i]}, z_2)$ has its last row identically equal to zero[4]. Thus it is possible to find a unimodular matrix $V_1(z_2)$ such that, where the matrix X is a $(m-1)\times n$ matrix,

$$V_1(z_2)R(z_1^{[1]}, z_2) = \left(\begin{array}{c} \mathbf{X} \\ \hline 0 \quad 0 \quad \cdots \quad 0 \quad 0 \end{array}\right)$$

where $\mathbf{X}$ is a $m-1$ by n polynomial matrix, i.e.

$$V_1(z_2)R(z_1, z_2) = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & z_1 - z_1^{[1]} \end{pmatrix} \tilde{R}_1(z_1, z_2)$$

[4] *The original proof* [33], [34] *uses the 1-D upper triangular form, which is only valid for full rank matrices.*

for some $\tilde{R}_1(z_1, z_2)$, or

$$R(z_1, z_2) = V_1^{-1}(z_2) \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & z_1 - z_1^{[1]} \end{pmatrix} \tilde{R}_1(z_1, z_2) \equiv A_1(z_1, z_2)\tilde{R}_1(z_1, z_2).$$

Step 3: Continue the procedure of Step 2 for all primitive roots $z_1^{[i]}$ for $i = 2, 3, \ldots, k$, where k is the number of primitive roots, using the matrix $\tilde{R}_{i-1}(z_1, z_2)$ in each case. A final result of the form

$$R(z_1, z_2) = \underbrace{A_1 A_2 \ldots A_k}_{\bar{R}(z_1, z_2)} R^*(z_1, z_2) \equiv \bar{R}R^*$$

is obtained where

$$|\bar{R}| = |A_1||A_2| \cdots |A_k| = \prod_{i=1}^{k} (z_1 - z_1^{[i]}) = \bar{r}(z_1)$$

and $R^*(z_1, z_2)$ is primitive. □

Note: The elementary operations used in forming the primitive factorisation are over the ring $F[z_2]$. Therefore the matrices $V_i(z_2)$, that effect the modified Hermite form are unimodular over $F[z_2]$ and hence $\det V_i(z_2) = \textit{constant}$.

4.4.5 GCD Algorithm

The following is an algorithm to calculate the greatest common right divisor (GCRD) using the canonical forms of 2-D Hermite form and 2-D primitive matrix (Definitions 4.5 and 4.6) over the ring $F[z_1][z_2]$. (The algorithm can be performed over the ring $F[z_2][z_1]$.) The GCRDs derived over $F[z_1][z_2]$ and $F[z_2][z_1]$ are both GCRDs over the ring $F[z_1, z_2]$ and therefore are linked by a unimodular matrix. This is demonstrated by the test examples in Chapter 7. The validity of the algorithm is provided in [34].

Algorithm 4.4: (*GCRD Algorithm over* $F[z_1][z_2]$)
The greatest common right divisor of the polynomial matrices $A_{p\times m}(z_1, z_2)$, $B_{q\times m}(z_1, z_2)$, with $m \leq p + q$, is given by:

Step 1: Use the right primitive factorisation algorithm over $F[z_1][z_2]$ to compute $\bar{A}^*(z_1, z_2)$, $\bar{B}^*(z_1, z_2)$ and $R_0(z_1, z_2)$ such that

$$\underbrace{\begin{pmatrix} A(z_1, z_2) \\ B(z_1, z_2) \end{pmatrix}}_{M(z_1, z_2)} = \underbrace{\begin{pmatrix} \bar{A}^*(z_1, z_2) \\ \bar{B}^*(z_1, z_2) \end{pmatrix}}_{\bar{M}^*(z_1, z_2)} R_0(z_1, z_2) \tag{277}$$

with $\bar{M}^*$ being primitive in $F[z_1][z_2]$ and $|R_0(z_1, z_2)| \in F[z_1]$.

Step 2: Use the right primitive factorisation algorithm with respect to $F[z_2][z_1]$ to form $A^*(z_1, z_2)$, $B^*(z_1, z_2)$ and $R_1(z_1, z_2)$ such that

$$\underbrace{\begin{pmatrix} \bar{A}^*(z_1, z_2) \\ \bar{B}^*(z_1, z_2) \end{pmatrix}}_{\bar{M}^*(z_1, z_2)} = \underbrace{\begin{pmatrix} A^*(z_1, z_2) \\ B^*(z_1, z_2) \end{pmatrix}}_{M^*(z_1, z_2)} R_1(z_1, z_2) \tag{278}$$

with $M^*(z_1, z_2)$ being primitive in $F[z_2][z_1]$ and $|R_1(z_1, z_2)| \in F[z_2]$.

Step 3: Form the 2-D Hermite form over $F[z_1][z_2]$, i.e.

$$U(z_1, z_2) \begin{pmatrix} A^*(z_1, z_2) \\ B^*(z_1, z_2) \end{pmatrix} = \begin{pmatrix} R(z_1, z_2) \\ 0 \end{pmatrix} \tag{279}$$

where $|U(z_1, z_2)| \in F[z_1]$ and $R(z_1, z_2)$ is the upper triangular square matrix as defined by (6) over $F[z_1][z_2]$.

Step 4: Form the left primitive factorisation of $R(z_1, z_2)$ over $F[z_1][z_2]$, giving

$$R(z_1, z_2) = \bar{R}(z_1, z_2) R^*(z_1, z_2) \tag{280}$$

where $|\bar{R}| \in F[z_1]$ is the primitive factor and $R^*(z_1, z_2)$ is primitive over $F[z_1][z_2]$.

Step 5: The greatest common right divisor, $D(z_1, z_2)$, of $A(z_1, z_2)$ and $B(z_1, z_2)$ is given by

$$D(z_1, z_2) = R^*(z_1, z_2) R_1(z_1, z_2) R_0(z_1, z_2) \tag{281}$$

where $R^*(z_1, z_2)$, $R_1(z_1, z_2)$ and $R_0(z_1, z_2)$ are defined respectively by (280), (278) and (277). □

The algorithm is dependent on the requirement that $A(z_1, z_2)$ and $B(z_1, z_2)$ are matrices for which $p+q < m$ due to the row-column condition of the 2-D primitive form. In practice (the formation of a coprime matrix fraction description, see Section 7.5) the GCD is required of two matrices, one of which is square. Therefore the row-column criterion will be satisfied.

The greatest common left divisor of two matrices, with more columns than rows, may be calculated using the above algorithm on the transposed pair $A^T(z_1, z_2)$ and $B^T(z_1, z_2)$.

4.5 Discussion of the Algorithms

The main discussion of the algorithms concerns the use of the modified Hermite algorithm. It is important to distinguish between the two types of Hermite algorithm used in the derivation of the greatest common divisor, namely the 2-D Hermite form and the 1-D modified Hermite form; although, in essence, the algorithms are equivalent. The situation is further complicated by the need to compute the 1-D Hermite form of a rectangular matrix with less than full rank (called a singular rectangular matrix).

4.5.1 Motivation for the Modified Hermite Algorithm

The original proof to the primitive factorisation theorem, [33], is set out as a constructive algorithm. This algorithm fails for a certain class of matrices. The problem arises from the assumption that the last row of the 1-D Hermite form, defined analogously to Algorithm 4.1, of a singular matrix is identically equal to zero; this is not always the case. To demonstrate this consider the following example.

Example 4.2: Consider the left primitive factorisation over $F[z_1][z_2]$ of the matrix $R(z_1, z_2)$,

$$R(z_1, z_2) = \begin{pmatrix} z_2 & z_2 + z_1 & z_2 & z_2 \\ z_2 & z_2 + z_1 & z_2 + z_1 & z_2 \\ z_2 & z_2 + 2z_1 & z_2 + 1 & z_2 + z_1 \end{pmatrix}.$$

The high-order minors are

$$-z_1^2 z_2, \quad 0, \quad z_1^2 z_2, \quad z_1^3.$$

Thus the primitive roots are the zeros of the polynomial $p(z_1) = z_1^2$, i.e. $z_1 = 0$. Consider the primitive root $z_1 = 0$. The matrix $R(0, z_2)$ is given by

$$R(0, z_2) = \begin{pmatrix} z_2 & z_2 & z_2 & z_2 \\ z_2 & z_2 & z_2 & z_2 \\ z_2 & z_2 & z_2 + 1 & z_2 \end{pmatrix}.$$

The original 1-D Hermite form (as defined in Algorithm 4.1 where $F[z_1]$ is replaced by the field F) is given by pre-multiplication by $V_1(z_2)$ where

$$V_1(z_2) = \begin{pmatrix} 1 + z_2 & 0 & -z_2 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$

to give

$$\begin{pmatrix} z_2 & z_2 & 0 & z_2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{282}$$

which does not possess the last row identically zero. However, a row permutation will bring the matrix to the correct form for continuation with the primitive factorisation algorithm but corrupts the Hermite form. The general situation can be easily rectified by a slight modification to the Hermite algorithm by ignoring the column that contains a zero in the pivotal position, i.e. the modified Hermite algorithm.

Thus in the above example the first column may be treated as normal by using element (1,1) to reduce the elements (2,1) and (3,1) to zero. Now the second column has a zero in the pivotal position (2,2), therefore use column 3 as the next column; the pivotal position is (2,3). The 1 in position (3,3) is now moved to this pivotal position. Thus the modified Hermite algorithm delivers the correct form for the primitive factorisation algorithm, i.e.

$$\begin{pmatrix} z_2 & z_2 & 0 & z_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

The matrix $R(z_1, z_2)$ can now be factored as

$$R(z_1,z_2) = \underbrace{\begin{pmatrix} 1 & 0 & z_2 \\ 1 & 0 & 1+z_2 \\ 1 & 1 & z_2 \end{pmatrix}}_{V_1^{-1}(z_1,z_2)} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & z_1 \end{pmatrix} \underbrace{\begin{pmatrix} z_2 & z_2+z_1 & z_2-z_1z_2 & z_2 \\ 0 & z_1 & 1 & z_1 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\tilde{R}(z_1,z_2)}.$$

□

The existence of an identically zero last row, as demonstrated in this example, may be proved for any singular matrix. Moreover if a matrix $A_{m\times n}$ with $m \leq n$ has rank r the last $m-r$ rows are identically zero. This is formally derived by the following theorem.

Theorem 4.2: *The upper triangular modified Hermite form of a polynomial matrix $A_{m\times n}(z_1)$ with $m \leq n$ and rank $m-1$ has the last row identically zero.*

Proof: The operations used to derive the upper triangular modified Hermite form may be expressed as elementary matrices over the ring $F[z_1]$, thus there exists a polynomial matrix $U(z_1)$ such that

$$H(z_1) = U(z_1)A(z_1) \tag{283}$$

where $H(z_1)$ is the Hermite form of $A(z_1)$ over $F[z_1]$ and $|U(z_1)| \in F$. Therefore the high-order minors of $H(z_1)$ and $A(z_1)$ are identical modulo a multiplicative element of F, the base field. Thus the ranks of $A(z_1)$ and $H(z_1)$ are identical, namely $m-1$.

This may be interpreted as $H(z_1)$ possessing $m-1$ linearly independent rows. By construction the first $m-1$ rows are linearly independent, thus row m is linearly dependent on rows 1 to $m-1$.

It is now required to show that row m is identically equal to zero. By the modified Hermite algorithm, Algorithm 4.2, the elements of row m in columns 1 to $m-1+$ *pivotzeros* are zero. But any non-zero linear combination of rows 1 to $m-1$ would contradict the first $m-1+$ *pivotzeros* elements of row m being zero. Therefore row m is identically zero. □

Corollary 1: *The Modified Hermite form of a matrix $A_{m\times n}(z_1)$ with $m \leq n$ and rank r has rows $r+1$ to m identically zero.*

Proof: This follows by extending the argument of the previous theorem. □

4.5.2 Alternative to the Modified Hermite Algorithm

An alternative method exists for defining a Hermite form of a singular matrix. When a non-zero pivot element can not be found consider permuting the columns so that a non-zero element is located in the current column, which may then be used as the pivot. The consequence of this action is to re-order the columns of the original matrix, which may easily be recovered by re-permuting the relevant columns once the Hermite form has fulfilled its purpose in the primitive algorithm. This 'alternative Hermite algorithm' more closely reflects the non-singular definition, in one sense, as all the elements on the principal diagonal are non-zero but requires pre- and post-multiplication by unimodular matrices, which defeats the purpose of the Hermite form, i.e. a canonical form using only row or column operations. Thus the modified Hermite algorithm defined above seems to be the more natural generalisation to singular rectangular matrices.

4.5.3 2-D Hermite Form as a Coprimeness Test

An interesting property of the 2-D Hermite form is its use as a simple coprimeness test. After the completion of Step 3 it may be applied as a termination condition for the algorithm. If the matrices A^*, B^* are coprime (either minor or zero) Step 4 may be waived, thereby increasing the efficiency of the algorithm and, due to the simplicity of the test the efficiency of the algorithm is not impaired by the extra step, if the matrices are not coprime.

Theorem 4.3: *(2-D Hermite Coprime Test)*
Let $M_{m\times n}(z)$ be a polynomial matrix with $m \geq n$ and suppose that $M(z_1, z_2)$ has no right divisor with determinant solely in z_1 or solely in z_2. If the 2-D Hermite form of $M(z_1, z_2)$ over $F[z_1][z_2]$ is a polynomial matrix solely in z_1, $M(z_1, z_2)$ is a right coprime matrix.

Proof: Since $M(z_1, z_2)$ has no right divisors with determinant solely in z_1 or z_2 the only type of right divisors are those with determinant polynomial in both z_1 and z_2 or a constant, i.e. a unimodular matrix (in which case $M(z_1, z_2)$ is a right coprime matrix).

Suppose that $D_{n\times n}$ is a right divisor of $M(z_1, z_2)$ with determinant polynomial in both z_1 and z_2, i.e.

$$M(z_1, z_2) = M'(z_1, z_2)D(z_1, z_2).$$

By computing the 2-D Hermite form of $M(z_1, z_2)$ over $F[z_1][z_2]$ there exists a polynomial matrix $U(z_1, z_2)$ such that

$$\begin{pmatrix} U_1 & U_2 \\ U_3 & U_4 \end{pmatrix} \begin{pmatrix} M_1 \\ M_2 \end{pmatrix} = \begin{pmatrix} R_{m\times m} \\ 0 \end{pmatrix}$$

where $U_1(z_1, z_2)$, $U_2(z_1, z_2)$, $U_3(z_1, z_2)$, $U_4(z_1, z_2)$ form a suitable partition of $U(z_1, z_2)$ and $M_1(z_1, z_2)$, $M_2(z_1, z_2)$ also form a suitable partition of $M(z_1, z_2)$. Additionally, $R(z_1, z_2)$ is in Hermite form with respect to $\mathbb{R}[z_1][z_2]$ and $|U(z_1, z_2)| \in \mathbb{R}[z_1]$.

Assume that $R(z_1, z_2)$ is a polynomial matrix with elements solely in z_1. Then

$$U_1M_1 + U_2M_2 = R$$
$$U_3M_1 + U_4M_2 = 0.$$

Write $M(z_1, z_2)$ as

$$M = \begin{pmatrix} M_1 \\ M_2 \end{pmatrix} = \begin{pmatrix} M_1' \\ M_2' \end{pmatrix} D$$

thus

$$(U_1M_1' + U_2M_2')D = R.$$

Hence $R(z_1, z_2)$ has a right divisor, D, i.e.

$$R = R'D$$
$$\Rightarrow \quad |R| = |R'||D|.$$

Now $|D|$ is polynomial in z_1 and z_2 but $|R(z_1, z_2)|$ is polynomial in z_1 only. Therefore there does not exist a $D(z_1, z_2)$ such that $|D(z_1, z_2)|$ is polynomial in z_1 and z_2. Hence $D(z_1, z_2)$ must have a determinant that is a constant, i.e. $D(z_1, z_2)$ is unimodular, thus $M(z_1, z_2)$ is a coprime matrix. □

This result is very efficient and computationally easy to apply. From Steps 1 and 2 of the GCRD algorithm over $F[z_1][z_2]$ the determinant condition of the right divisors is guaranteed. Thus after completion of Step 3 the matrix $R_{n\times n}(z_1, z_2)$ may be inspected for elements in z_2; this is a particularly easy procedure in MAPLE. If the resulting Hermite form possesses the form of the theorem Step 4 of the GCRD algorithm is redundant and may be omitted. If the Hermite form does possess both indeterminates then the simplicity of the test ensures minimal loss of efficiency.

4.5.4 GCD Modification

The main difference between the theoretical basis for the GCD algorithm presented in [33], [34] and the practical algorithm, given in Section 4.3, is the addition of Step 2. The advantage of this addition is that it allows the use of a particularly simple termination test using the Hermite form, as derived above.

One further advantage of including Step 2 is to reduce the complexity of the polynomial matrix before performing the 2-D Hermite form (over $F[z_1][z_2]$). The indication from test examples is that the degree in z_1 of the polynomial elements of the Hermite form quickly increases due to the nature of the pseudo-division algorithm. Therefore it is computationally more desirable to perform the 2-D Hermite algorithm on a matrix with as little complexity as possible, i.e. with polynomial elements as small as possible, since it is the Hermite procedure that has the most influence on the efficiency of the program, see Chapter 7.

The form of the GCD Algorithm suggests that the polynomial matrix factors of a 2-D matrix form three sets, characterised by the determinant of the matrix factor. These sets may be defined as:

$$S_1 = \{a_1 : a_1 \in F[z_1]\}$$

$$S_2 = \{a_2 : a_2 \in F[z_2]\}$$

$$S^* = \{a^* : a^* \in F[z_1, z_2] \text{ and is not divisible by an element of } F[z_1] \text{ or } F[z_2]\}$$

Step 1 of the GCRD Algorithm, Algorithm 4.4, computes the factors with determinants that lie in S_1 and Step 2 computes the factors with determinants that lie in S_2. At this stage of the algorithm the matrix $M^*(z_1, z_2)$ is primitive with respect to both z_1 and z_2. In performing the second primitive factorisation, factors in z_1 are not reintroduced because the primitive factorisation algorithm is achieved by elementary operations in a Euclidean ring, therefore the determinant of the elementary matrices are constants.

The factors that remain have determinants in S^*. Their removal is achieved by a generalisation of the 1-D technique to compute the GCD of a 1-D polynomial matrix, namely the Hermite form [13]. However, the 2-D Hermite form can not be derived by rank preserving operations alone and therefore does not necessarily preserve the primitiveness with respect to the ring $F[z_1][z_2]$ (in Step 3 of Algorithm 4.4). Thus Step 4 is necessary to restore the primitiveness with respect to $F[z_1][z_2]$. The primitiveness with respect to $F[z_2][z_1]$ is guaranteed by the form of the Hermite Algorithm, since all operations are unimodular in z_2.

The three types of polynomial factors that exist for a particular 2-D polynomial matrix can be determined at the outset of the algorithm. Consider the matrix $A_{m\times n}(z_1, z_2)$ with $m \geq n$ and high-order minors denoted by $a_i(z_1, z_2)$. The greatest common divisor of these polynomials, $g(z_1, z_2)$, is the determinant of the greatest common right divisor (modulo a multiplicative constant). Therefore $g(z_1, z_2)$ may be uniquely factorised (modulo a multiplicative constant) such that

$$g(z_1, z_2) = g_1(z_1, z_2)g_2(z_1, z_2)g^*(z_1, z_2)$$

where $g_1(z_1) \in S_1$, $g_2(z_2) \in S_2$ and $g^*(z_1, z_2) \in S^*$ define the three types of factor.

4.6 Conclusions

In this chapter the mathematical basis has been given to determine the greatest common divisor of a 2-D polynomial matrix. The complexity of the problem arises from the absence of a division algorithm in the ring $F[z_1, z_2]$, to which the elements of the matrices belong. Therefore 1-D techniques alone are not sufficient.

The algorithms presented here are improved versions of the original theoretical ideas by [33], [34] on two levels. Firstly, a theoretical deficiency due to the vague definition of the Hermite form for singular rectangular matrices has been fully explored and corrected. Secondly, modifications have been made to assist in the practical implementation of the algorithms.

The most notable modification is the inclusion of Step 2 in Algorithm 4.4. This was motivated by efficiency considerations on two accounts. Firstly, by reducing the complexity of the matrix before the 2-D Hermite is computed thus potentially reducing the degrees of the polynomial elements in the Hermite form and secondly a particularly simple and efficient termination condition may be applied to prevent unnecessary calculations being performed.

Chapter 5

MAPLE: A Symbolic Manipulator

5.1 Introduction

MAPLE is a symbolic manipulation language developed by the University of Waterloo, Canada. The language possesses two basic features that make it particularly appropriate for the implementation of the algorithms presented in the previous chapter, namely the ability to leave the elements of a computation unevaluated and the subsequent simplification of such elements. Thus polynomials are stored as algebraic entities, not as an array of coefficients. The basic system, or kernel, is written in terms of macros that are translated by a macro processor (Margay) into versions of the C

programming language on various operating systems. The algorithms are encoded using MAPLE version 4.3 on a SUN2/280S with a UNIX operating system.

MAPLE may be used in two different ways: either interactively or non-interactively. The former is mainly used for short computations or writing procedures and the latter for executing a sequence of time consuming procedures. The set of commands that occurs between the invocation of MAPLE and the quitting of MAPLE is termed a MAPLE session, thus a session may be interactive or non-interactive.

The MAPLE software contains many predefined functions that are stored as two main types; automatically loaded functions available upon invocation of MAPLE and functions that have to be manually loaded from one of the library packages, containing a collection of related functions. In particular, the linear algebra package, `linalg`, contains a collection of functions to manipulate matrices, used extensively in this program. Another example of a predefined function extensively used in the implementation of the algorithms is the pseudo-division algorithm, Definition 4.2. Other types of 'basic' algebraic operations are also predefined functions such as polynomial factorisation and equation solving.

The user communicates with the kernel by executing commands in a session using MAPLE's own high-level language, which is particularly suitable for describing algebraic algorithms, such as those detailed in the previous chapter, due to the procedural programming nature of the language. Each algorithm is broken into small stand-alone segments. These are then linked by a controlling procedure to execute the steps of the algorithm. Using this type of programming the code may be organised in such a way that the controlling procedure reads analogously to the theoretical algorithm.

Procedural programming such as that described above is the main method adopted to implement the GCRD algorithm. The code for each procedure is created in a text file and then read into a MAPLE session as required. A procedure remains defined in a MAPLE session until either the session is terminated, by quitting MAPLE, or redefined by reassigning the procedure name. Thus it is only necessary to read-in a procedure once during a session.

The purpose of this chapter is to introduce some of the terminology of MAPLE required for the understanding of the documentation and also some of the conventions used in the following chapters.

5.2 Conventions

(1) In the program documentation that follows a MAPLE expression is set in `typewriter` font. This will avoid confusion when discussing MAPLE commands within the text of a paragraph.

(2) Single quotes " ' " and back quotes " ` " are part of the language syntax, thus they are set in `typewriter` font to distinguish them from the punctuation marks of single quote " ' " and back quote " ` ".

(3) The statement separators in MAPLE are

   (i) semi-colon ";"

   (ii) colon ":"

   The distinction is that the latter prevents the preceding statement being printed during an interactive session. Notice that the statement separators are set in `typewriter` font, since they form part of a MAPLE expression. Every statement in MAPLE must be followed by a statement separator, hence every line ends with either a colon or semi-colon. These are not to be confused with the grammatical punctuation marks of ";" and ":" used in the text.

(4) The object enclosed between "[[" and "]]" denotes an insertion of the specified object. Specifically,

   (a) [[stat]] is replaced by a statement sequence;

   (b) [[expr]] is replaced by an expression;

   (c) [[string]] is replaced by a collection of elements from the character set;

   (d) [[indet]] is replaced by an unassigned name, i.e. an indeterminate;

   (e) [[name]] is replaced by a string of elements from the character set

   If the object is post-fixed by "seq" a sequence of the specified object is the replacement. For fuller definitions of the above objects and the character set see the MAPLE reference manual [78].

5.3 Statements

There are eight types of statement in MAPLE five of which are used in the documentation. These are described below.

1. **Assignment:** Takes the form of

   [[name]] := [[expression]];

   and associates a name with the value of an expression.

2. **Read:** Takes the form of

   `read` [[expression]];

   and causes a file to be read into the maple expression. The [[expression]] must evaluate to a name of a file.

3. **Save:** Takes the form of

   `save` [[expression]];

   and saves the current session into a file. The [[expression]] must evaluate to a name which specifies a file. If the file already exists it is overwritten.

4. **Selection:** The selection statement takes one of the following forms

   `if` [[expr]] `then` [[statseq]] `fi;`

   `if` [[expr]] `then` [[statseq]] `else` [[statseq]] `fi;`

   `if` [[expr]] `then` [[statseq]] `elif` [[expr]] `then` [[statseq]] `fi;`

   `if` [[expr]] `then` [[statseq]] `elif` [[expr]] `then` [[statseq]] `else` [[statseq]] `fi;`

   and '`elif` [[expr]] `then` [[statseq]]' may be repeated any number of times to yield a valid selection statement. The '`elif` [[expr]] `then` [[statseq]]' construct has the meaning '`else if` [[expr]] `then` [[statseq]]', but the latter form requires a closing '`fi`' for each opening '`if`'.

5. **Repetition:** The syntax takes the form of

   `for` [[name]] `from` [[expr]] `by` [[expr]] `to` [[expr]] `while` [[expr]] `do` [[statseq]] `od;`

   where any of '`for` [[name]]', '`from` [[expr]]', '`by` [[expr]]', '`to` [[expr]]' or '`while` [[expr]]' may be omitted. If the '`from` [[expr]]' or '`by` [[expr]]' is omitted the default value 1 is used. If '`to` [[expr]]' or '`while` [[expr]]' is present the corresponding tests for termination are checked for at the beginning of each iteration.

   There are two other loop constructs: `break` and `next`.

   `break`: exits from the innermost repetition statement within which it occurs. Execution then proceeds with the first statement following the repetition statement.

   `next`: exits from the current [[statseq]] and proceeds with the next iteration of the repetition statement.

5.4 Expressions

Expressions are the fundamental entities in the MAPLE language. The various types that are used for this program are set out below.

5.4.1 Names and Strings

A name in MAPLE has a value which may be any expression or, if no value has been assigned to it, it stands for itself. A name is usually a [[string]], which in its simplest form is a letter followed by zero or more letters, digits and underscores (with a maximum length of 495 characters). Lowercase and uppercase letters are distinct, also the underscore is a valid character. There are 30 reserved words, which are not normally available, and names beginning with an underscore are used by the system for global variables.

Another way to form a [[string]] is to enclose a sequence of characters in back quotes. (To allow the back quote to be used as a character two consecutive back quotes appearing after the opening of a string in back quotes are parsed as an enclosed back quote.) A reserved word enclosed in back quotes becomes a valid string, distinct from the usage of the reserved word as a token. The back quotes do not themselves become part of the string and are striped away when input into MAPLE. By using back quotes it is possible to use a newline as a valid character within a string. (This is not recommended.) If a name is too long to fit onto one line it may be continued on the next line by placing a backslash before the <RETURN> . This makes the newline character transparent.

A more general type of [[name]] may be formed by using the concatenation operator in one of the following forms.

[[name]] . [[natural integer]]
[[name]] . [[string]]
[[name]] . ([[expression]])

The concatenation operator is a binary operator which requires a [[name]] as the left operand. Its right operand is evaluated and concatenated to the left operand. If the right operand evaluates to an integer or a string then the result of the concatenation is a name. As an example if `n` has the value 4 then `p.n` evaluates to the name `p4`, while if `n` has no value `p.n` evaluates to `pn`.

Yet another form of name is the indexed name which has the form

[[name]] [[[expression sequence]]].

Note that since an indexed name is itself a valid name, there can be a succession of subscripts. The use of the indexed name `b[1]` does not imply that `b` is an array, as is true in other languages. It is not necessary that `b` has an array value, however, if `b` does evaluate to an array or table, `b[1]` is the element of the array or table selected by 1. The assignment of a value to an indexed name will implicitly create a table.

5.4.2 Sets and Lists

A set is an expression of the form

{ [[expression sequence]] }

and a list is an expression of the form

[[[expression sequence]]]

Note that the [[expression sequence]] may be empty so that the empty set { } and the empty list [ ] are valid. The difference between a set and a list is that a set is an unordered sequence of expressions, with duplicates removed, while a list retains the order of the expressions.

5.4.3 Ranges

A range is specified via the ellipsis operator:

[[expression]] .. [[expression]]

the operator here can be specified as two or more consecutive periods. To convert a range into an expression sequence, the $ operator may be applied. For example

1 .. 5 yields 1 .. 5

$ 1..5 yields 1, 2, 3, 4, 5

The [[range]] construct can also be used in combination with the concatenation operator to form an [[expression sequence]].

5.4.4 Selection Operation:

There are two alternative ways to select components from an aggregate object. The first is by using square brackets:

[[name]] [[[expression sequence]]]

where [[name]] must evaluate to one of the following:

name, table, array, list, set, expression sequence

and the second is to use the op function:

op([[expression sequence]],[[name]])

where [[name]] must evaluate to one of the following:

list, set.

If [[name]] evaluates to an unassigned name, then [[name]] [[[expression sequence]]] is an indexed name. If [[name]] evaluates to an array or table, the selection operation is an indexing operation. If [[name]] evaluates to a list, set or expression sequence, [[expression sequence]] must evaluate to an integer, range or null.

5.4.5 Procedures

The procedure definition takes the form of

```
proc ( [[nameseq]] ) local [[nameseq]];
options [[nameseq]];
[[statseq]]
end
```

where the 'local [[nameseq]];' part and the 'options [[nameseq]];' part may be omitted. A procedure is then invoked by

[[name]] ([[expression sequence]]);

For the semantics of procedure definition see Section 5.5 below.

5.4.6 Arrays and Tables

This type of construct is the most frequently used in this program as it is the type of data structure in which the elements of a matrix are stored. The matrix

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{2,1} & a_{2,2} \end{pmatrix}$$

is represented by

array([[a_{11},a_{12}],[a_{21},a_{22}]]);

However, this construct is much more general since it can have any number of dimensions, not just two as for matrix representation, see [78].

5.4.7 Unevaluated Expressions

An expression that is enclosed within a pair of single quotes is called an *unevaluated expression*. The effect of evaluating a quoted expression is to strip off one level of quotes, so that in some instances it is useful to use nested levels of quotes. Note the difference between 'evaluation' and 'simplification'. For example

```
x := '2 + 3';
```

will cause the value 5 to be assigned to the name x. The evaluator simply strips off the quotes but it is the simplifier that transforms the expression 2 + 3 into the constant 5.

A special case of 'unevaluation' arises when a name which may have been assigned a value needs to be unassigned, so that subsequently the name stands for itself. This is achieved by assigning the quoted name to itself. For example

```
x := 'x';
```

assigns the name x to the name x, so that if it previously had been assigned a value it will now stand for itself.

5.5 Procedure Definition

The procedure definition takes the form of

```
proc ( [[nameseq]] ) local [[nameseq]];
options [[nameseq]];
[[statseq]]
end
```

where the [[nameseq]] enclosed in `proc( )` are the formal parameters. The '`local` [[nameseq]];' part and the 'options [[nameseq]];' part may be omitted. A procedure is invoked by

```
[[name]] ( [[expression sequence]]);
```

The [[expression sequence]] is called the actual parameters of the invocation. When a procedure is invoked the statements in [[statseq]] are executed sequentially. The value of a procedure invocation is the value of the last statement in [[statseq]] that is executed.

The keywords 'proc' and 'end' may be viewed as brackets which specify that the [[statseq]] is to remain unevaluated when the procedure definition is evaluated as an expression.

5.5.1 Parameter Passing

The semantics of parameter passing are as follows. Suppose the procedure invocation is of the form

$$name\ (expr_1, expr_2, \ \ldots\ , expr_n).$$

First, *name*, is evaluated and suppose that it evaluates to a procedure definition with formal parameters

$$param_1, param_2, \ \ldots\ , param_n.$$

Next the actual parameters $expr_1$, ..., $expr_n$ are evaluated in order from left to right. Then every occurrence of $param_i$ in the [[statseq]] which makes up the body of the procedure is substituted by the value of the corresponding actual parameter $expr_i$.

It is possible for the number of actual parameters to be either greater than or less than the number of formal parameters specified. If there are too few parameters a semantic error will occur only if the corresponding formal parameter is referenced during execution. The case where there are more actual parameters than formal parameters is fully legitimate.

There are three special names that MAPLE understands within a procedure body.

(i) `args` is the expression sequence of actual parameters with which the procedure was invoked. Thus `args[i]` evaluates to the `i`th actual parameter;

(ii) `nargs` is the number of actual parameters that the procedure was invoked with;

(iii) `procname` is the procedure name.

5.5.2 Local Variables

The mechanism for introducing local variables into a MAPLE procedure is to use the `local` part of the procedure definition. It must appear immediately after the parentheses enclosing the formal parameters. The syntax is

```
local [[nameseq]];
```

where the names appearing in [[nameseq]] are the local variables of the procedure.

5.5.3 Explicit Returns

The most common way of returning from a procedure invocation is when execution falls through the end of the [[statseq]]. The value of the procedure invocation is the value of the last statement executed.

An *explicit return* occurs via the special function

```
RETURN( [[exprseq]] );
```

This function call causes an immediate return from the procedure and the value of the procedure invocation is the value of the [[exprseq]] given as the argument to `RETURN`. A particular form of explicit return is often used as a fail return, in the sense that the computation can not be carried out and it is desired to return the unevaluated invocations as the result. For this purpose two special names mentioned earlier are used as follows:

```
RETURN( 'procname ( args )' );
```

note the use of single quotes (not back quotes), so that the name is only evaluated when the procedure is returned from and not during the procedure, otherwise the procedure would invoke itself infinitely many times.

5.5.4 Reading and Saving Procedures

The procedures are stored in text files corresponding to the name of the procedure. A file containing MAPLE expressions is read into a MAPLE session with the syntax

```
read [[name]];
```

where [[name]] evaluates to a file name that contains a procedure. Once the procedure is error free it is desirable to save it in 'MAPLE internal format' so that whenever it is read into a MAPLE session the reading is fast. The method for saving a procedure in internal format is by ending the filename in the characters '.m'. For example, suppose a file named 'deg' contains a procedure definition, this procedure is read into a MAPLE session and saved in internal format via

```
read deg;
save `deg.m`;
```

the procedure can then be read into any MAPLE session via

```
read `deg.m`;
```

Note: filenames involving characters such as '/' or '.' must be enclosed in back quotes so that they are interpreted properly as [[name]]s

5.6 The MAPLE Library of Functions

MAPLE provides an extensive library of predefined functions. Some of the most commonly used are present upon invocation and others may be loaded when required. For ease of identification the library of functions is divided into packages, examples of these packages include `powseries` for formal power series, `orthopoly` for orthogonal polynomials, `numtheory` for number theory and the most extensively used package in this program `linalg` for linear algebra and matrix manipulation.

The predefined functions used in the program are documented below, together with the syntax as used in the program. Other uses of the functions below may be available, for a description of these and a fuller discussion of each function see the MAPLE reference manual [78]. The functions that have been used fall into four categories. Firstly, the standard library functions that are present at invocation; secondly, the `linalg` functions for matrix manipulation; thirdly, the `combinat` functions for forming combinations of a list; and finally the `miscellaneous` functions.

5.6.1 Standard Library Functions

`break` - break from a recursive procedure.

see section 5.3.

`convert(`[[name]]`, list);` - convert to a list.

Converts the elements of [[name]] to a list, where [[name]] is one of the following data types

table, vector, expression.

`convert(`[[name]]`, set);` - convert to a set.

Converts the elements of [[name]] to a set, where [[name]] is one of the following data types

table, array, expression.

`convert(`[[expression]]`, '*');` - convert to a product.

Converts the ops of [[expression]] to a product. Thus if [[expression]] evaluates to a list of elements the above function returns a single object formed by multiplying the elements of the list together.

`copy(a);` - create a duplicate array or table.

Parameter: a - a table or array.

The purpose of the copy function is to create a duplicate table or array, which may then be altered without affecting the original table or array. If a does not evaluate to a table or array a is returned. If a [[name]] is assigned to a table or array without using copy the elements of the original table or array are altered whenever the elements of [[name]] are altered. For example if the statements

```
s := table();
t := s;
```

are executed both `t` and `s` evaluate to the same table structure.

`degree(p,x);` - degree of a polynomial.

Parameters: p - an expression in the indeterminate(s) x,
x - an indeterminate or a set or list of indeterminate(s).

Computes the degree of p in the given indeterminates. The result is either a positive or negative integer if p is polynomial, or `FAIL` if it is not polynomial in the indeterminate(s) x.

If x is a set of indeterminates the total degree in the set of indeterminates is computed. If x is not specified `degree(p,indets(p))` is understood.

`divide(a,b,'q');` - exact polynomial division.

Parameters: a, b - polynomials with rational number coefficients,
q - (optional) an unevaluated name.

Checks if the polynomial a divides the polynomial b over the rational numbers. If so, `true` is returned, otherwise `false`.

If the division is successful and there is a third argument `'q'`, the value of the quotient is assigned to q. In the case of unsuccessful division the name q will not be affected.

`ERROR([[exprseq]]);` - error return from a procedure.

A call to the `ERROR` function causes an immediate return to the point where the current procedure was invoked. The error message [[exprseq]] is then printed.

`expand([[expr]]);` - expand an expression.

Parameter: [[expr]] - an algebraic expression.

The function expands [[expr]] so that products are distributed over sums for all polynomials.

`factor([[expr]]);` - factor a multivariate polynomial.

Parameter: [[expr]] - an algebraic expression.

Computes the factorisation of a multivariate polynomial with integer or rational coefficients.

`gcd(a,b);` - greatest common divisor.

Parameters: a, b - multivariate polynomials over the rationals.

Computes the greatest common divisor of two polynomials with rational coefficients.

`has(f,x);` - test for specified subexpression.

Parameters: f, x - expressions.

If the expression f contains the expression x, the result of the function is `true`. Otherwise the result is `false`.

`indets([[expr]]);` - indeterminates of an expression.

Parameter: [[expr]] - an algebraic expression.

Computes the set of indeterminates of the expression [[expr]].

`lcoeff(p,x);` - leading coefficient of a multivariate polynomial.

Parameters: p - multivariate polynomial,

x - (optional) indeterminate or list or set of indeterminates.

Computes the leading coefficient of p in the indeterminate or indeterminates x. If x is not specified the default value is `indets(p)`.

`max(`$x_1, x_2, \ldots, x_n$`);` - maximum of numbers.

Parameters: $x_1, x_2, \ldots, x_n$ - numbers.

Computes the maximum of the numbers $x_1, x_2, \ldots, x_n$.

`min(`$x_1, x_2, \ldots, x_n$`);` - minimum of numbers.

Parameters: $x_1, x_2, \ldots, x_n$ - numbers.

Computes the minimum of the numbers $x_1, x_2, \ldots, x_n$.

`nargs` - number of arguments.

Computes the number of actual parameters with which the procedure was invoked.

`nops(`[[expr]]`);` - number of operands.

Parameter: [[expr]] - any expression.

Computes the number of components of an expression. If [[expr]] is a list or set then **nops** returns the number of elements of the set or list. If [[expr]] is a product or sum, **nops** returns the number of operands in the product or summation.

`op(`*i*`,`*e*`);`
`op(`*i*`..`*j*`,`*e*`);`
`op(`*e*`);` } – extract operands.

Parameters: *i*, *j* - integers,
e - any expression.

Extracts the components of an expression. In the first form the *i*th component is extracted from *e*. In the second form components *i* to *j* are extracted from *e* and in the third form all components are extracted, i.e. `op(`*e*`)` ≡ `op(1..nops(`*e*`),`*e*`)`.

`'procname(args)'` - procedure name and actual arguments.

A construct used in conjunction with `RETURN`, see Section 5.5.3.

`radsimp(`[[expr]]`);` - simplify expressions containing radicals.

Radicals are expressions to a fractional power, e.g. $2^{1/3}$. The function `radsimp` simplifies radicals contained in [[expr]].

`RETURN(`[[expression sequence]]`);` - return from a procedure.

A special function that forces immediate return from the procedure. The value of the procedure is the value of the [[expression sequence]]. The null expression sequence is perfectly valid. A particular form of return is the fail return, formed in conjunction with `'procname(args)'`, see Section 5.5.3.

`solve(`[[eqn]]`,`[[indets]]`);` - solve equations.

Parameters: [[eqn]] - an equation or set of equations,
[[indets]] - an indeterminate or set of indeterminates.

Solve an equation or set of equations in the given indeterminate(s) [[indets]]. The solution is returned as an expression sequence. If an expression [[expr]] is supplied as one of the actual parameters the equation [[expr]]=0 is understood. If [[indets]] is not specified then `indets(`[[eqn]]`)` is understood.

If the equation, to be solved, is a high order polynomial the solve function will express the result as `RootOf([[eqn]])` and not evaluate the result fully. To obtain the full set of roots evaluated as exact numbers (radical notation) the construct

`allvalues(solve([[eqn]],[[indets]]));`

must be given. The construct

`evalf(solve([[eqn]],[[indets]]));`

obtains the numbers in decimal notation.

`sprem(`a`,`b`,`x`,'`m`','`q`');` - sparse pseudo-remainder.

Parameters: a, b - multivariate polynomials in the variable x,

x - indeterminate,

m, q - (optional) unevaluated names.

the function returns the pseudo-remainder r such that

$$ma = bq + r$$

where `degree(`r`,`x`)<degree(`b`,`x`)` and q is the pseudo-quotient. The multiplier m is the smallest possible power of `lcoeff(`b`,`x`)` such that the division process does not introduce fractions into q and r.

`subs(`$s_1, s_2, \ldots, s_n$`,[[expr]])` - substitute subexpressions into an expression.

Parameters: $s_1, s_2, \ldots, s_n$ - equations or sets or lists of equations,

[[expr]] - any expression.

Returns an expression resulting from substituting $s_1, s_2, \ldots, s_n$ into [[expr]]. The substitutions are performed sequentially starting with s_1, if one of the subexpressions is a set or list the substitutions are performed simultaneously. Only subexpressions in [[expr]] that correspond to an operand of a MAPLE object are matched. This is called *syntactical substitution.*

`type([[expr]],'+');` - check for summation.

Parameter: [[expr]] - any expression.

The function returns `true` if [[expr]] is a summation, otherwise `false` is returned.

Note that $x^2 + y + xy$ is of type '+' but $(x + y)(x^2 + xy)$ is not.

`type(`[[expr]]`,'*');` - check for product.

Parameter: [[expr]] - any expression.

The function returns `true` if [[expr]] is a product, otherwise `false` is returned. Note that $(x+y)(x^2+xy)$ is of type `'*'` but x^2+y+xy is not.

`type(`[[expr]]`,numeric);` - check for number.

Parameter: [[expr]] - any expression.

The function returns `true` if [[expr]] is numeric, i.e. an integer, fraction or floating point number, otherwise `false` is returned.

`type(`[[expr]]`,polynom,`[[indet]]`,`[[domain]]`);` - check for polynomial.

Parameters: [[expr]] - any expression,
[[indet]] - an indeterminate or a set or list of indeterminates,
[[domain]] - the coefficient domain.

The function returns `true` if [[expr]] is a polynomial in the given indeterminate(s) [[indet]], otherwise `false` is returned.

5.6.2 The `linalg` Package for Linear Algebra

The `linalg` package contains functions for the computation of matrices and associated properties. These functions have to be loaded as required by using one of the following constructs.

(i) The long form notation:

`linalg[`[[function]]`](`[[arguments]]`);`

this form must be used whenever there is a conflict of function name between a package function name and another function in a MAPLE session.

(ii) The short form notation:

[[`function`]]`(`[[arguments]]`);`

where the construct `with(linalg,`[[function]]`);` has been executed before the function is invoked. Alternatively, all the `linalg` functions may be loaded via

`with(linalg);`

The package contains a total of 57 functions. Those functions called in the program are detailed below in the context they are used.

`coldim(`*A*`);` - column dimension of a matrix.

Parameter: *A* - a matrix.

The result of this function is an integer corresponding to the number of columns in the matrix *A*.

`det(`*A*`);` - determinant of a matrix.

Parameter: *A* - a square matrix.

The determinant of a square matrix is computed by either minor expansion or Gauss elimination. If the matrix is sparse minor expansion is used, otherwise a combination of both methods are used.

`inverse(`*A*`);` - inverse of a square matrix.

Parameter: *A* - a square matrix.

Computes the inverse of a square matrix, if the matrix is non-singular, otherwise an error occurs. The method for computation is Cramer's rule for matrices of dimension less than or equal to 4 by 4 and sparse matrices, or by applying the operations of Gauss-Jordan elimination to a unit matrix of the same size, for matrices greater than 4 by 4.

`multiply(`*A*`,`*B*`);` - matrix multiplication.

Parameter: *A*, *B* - matrices.

The matrix product of *A* and *B* is computed if the matrices are compatible, i.e. `coldim(`*A*`)=rowdim(`*B*`)`. The result is a matrix with dimensions `rowdim(`*A*`)` by `coldim(`*B*`)`. An alternative function is available for matrix multiplication, which may also be used for other computations with matrices. The function syntax is

`evalm(`*A* `&*` *B*`);`

however, the number of arguments is not restricted to two and can be any matrix expression, see `evalm`.

`rank(`*A*`);` - rank of a matrix.

Parameter: *A* - a matrix.

The rank of the matrix *A* is computed by performing Gauss elimination on the rows of *A*.

`submatrix(`A`,`[[rows]]`,`[[cols]]`);` - extract a submatrix from a matrix.

Parameters: A - a matrix,

[[rows]] - a list or range of rows,

[[cols]] - a list or range of columns.

The result of this function is a submatrix of A whose (i,j)th element is the element in the row given by the ith component of [[rows]] and column given by the jth component of [[cols]].

`swaprow(`A`,`r_1`,`r_2`)` - swap two rows in a matrix.

Parameters: A - a matrix,

r_1, r_2 - integers denoting two different rows.

The result is a matrix whose entries are the same as the matrix A except that rows r_1 and r_2 are interchanged.

`transpose(`A`)` - transpose of a matrix.

Parameter: A - a matrix.

The result of this function is the transpose of the matrix A, i.e. element (i,j) becomes element (j,i) of the resulting matrix.

5.6.3 The `combinat` package for combinations

This package of functions computes the combinations and permutations of a group of data structures. Only two of the functions are used, these are listed below. The functions must first be loaded in a similar way to the `linalg` functions before they can be used, i.e. by either

(i) loading all the functions in the `combinat` package via

```
with(combinat);
```

or (ii) loading the individual function [[function]] via

```
with(combinat,[[function]]);
```

or (iii) if there is a conflict of function name the long form notation must be used

```
combinat[[[function]]]([[args]]);
```

which requires no preloading.

`combinations(n,m);` - count the number of combinations

Parameters: n - a list of objects or a natural number,

m - integer.

If n is a list the result is the number of combinations of the elements of n taken m at a time.

If n is a natural number the result is the number of combinations of selecting m integers from the first n natural numbers.

`combine(n,m)` - construct the combinations of a list.

Parameters: n - a list of objects or a natural number,

m - integer.

If n is a list, then `combine` returns a list of the combinations of the elements of n taken m at a time.

If n is a natural number, this is equivalent to using the list of the first n natural numbers.

5.6.4 Miscellaneous Functions

The functions occurring in this package are extensions of the standard library functions but do not fit into any large group that has its own library. Only two of these functions are used.

`allvalues([[expr]])` - evaluate `RootOf`'s and return all possible values.

Parameter: [[expr]] - any expression or table, list, or set of expressions.

The return from this function is the evaluation of expressions containing `RootOf`s. Typically, a `RootOf` represents more than one value. Thus expressions involving `RootOf`'s will generally evaluate to more than one value or expression. The function `allvalues` will return all such values (or expressions) generated by the combinations of different values of the `RootOf`'s, in an expression sequence.

This function must be loaded with

```
readlib(allvalues);
```

before it can be used.

`evalm(`[[matrix expression]]`)` - matrix evaluation.

Parameter: [[matrix expression]] - an expression involving matrices.

This function gives an alternative to the `linalg` function `multiply` for evaluating matrices. The syntax for this function is more concise than for the equivalent result using the `linalg` package, as more than two matrices may be contained in the matrix expression.

Matrix operations are non-commutative thus the character `&` must precede any operation symbol. For example, to multiply the three matrices A, B and C either the `linalg` construct `multiply(`A`,multiply(`B`,`C`))` or the more concise notation

`evalm(`A `&*` B `&*` C`)`

may be used. Note that a space must precede and succeed `&*`. The function must also be loaded via

```
readlib(evalm);
```

5.7 Implementation of the Algorithms

The algorithms described in the previous chapter are implemented in three main blocks corresponding to the three algorithms, modified Hermite, primitive factorisation and greatest common right divisor. The procedures that effect these algorithms are `shiftherm`, `lprimfac` and `gcrd`, respectively. Each of the procedures requires at least two actual parameters with a third being optional. The first of these parameters is a matrix, the second is the favoured indeterminate and the third optional parameter is an unevaluated name that is assigned during the procedure: in the case of

(a) `gcrd` the coprime matrix resulting from the extraction of the greatest common right divisor,

(b) `lprimfac` the primitive factor with determinant polynomial solely in the specified determinant;

(c) `shiftherm` the unimodular matrix effecting the elementary operations.

Note: The duals of these procedures, namely the greatest common left divisor (for matrices with more columns than rows), the lower triangular modified Hermite form and right primitive factorisation are obtained by transposing the result returned when using the transpose of the specified matrix as the actual parameter. For example the greatest common left divisor of the matrix $A_{m\times n}$, with $m \leq n$, is computed by applying the `gcrd` procedure with matrix argument A^T and then transposing the result returned from the procedure.

It is not necessary to define a separate procedure to calculate the 2-D Hermite algorithm (Algorithm 4.1) since by the discussion following Algorithm 4.2, the Hermite algorithm and modified Hermite algorithm are coincident for full rank matrices. This property is due to the row and column configuration of the matrix. Also by the design of the procedure the total number of indeterminates is not important provided that at least one exists. Therefore the procedure can be used to calculate both the 1-D Hermite form and the 2-D Hermite form for both full rank and singular matrices.

Two further points need to be emphasised regarding the displaying and storing of matrices as two dimensional arrays. An array in MAPLE is stored as a special form of the object `table` with specified dimensions indexed by an integer range. The elements of a matrix A are displayed with the function `op(`A`)`; not by executing A; (the way of displaying the 'value' of a name) which returns the value A. Also if a ⟦name⟧ is assigned to another ⟦name⟧ that evaluates to a table or an array without using the function `copy` both ⟦name⟧s evaluate to the same table structure, thus by changing the value of one of the elements of one ⟦name⟧ the value contained in

the corresponding position of the other [[name]] is also altered. To avoid this copy creates a duplicate table with elements equivalent at the time of execution only and subsequent reassignments are not reflected in both table structures.

Chapter 6

Code Documentation

6.1 Introduction

The code is organised into three main sections corresponding to the three major algorithms presented in Chapter 4, namely the greatest common divisor algorithm, the primitive factorisation algorithm and the modified Hermite algorithm. The first contains the controlling procedure for the calculation of the greatest common left divisor and the greatest common right divisor of a matrix. The second contains the procedures necessary for the calculation of the left and right primitive factorisations and the third contains the modified Hermite procedures. Some of the procedures are

common to all three sections and are only documented once in either the second or third section.

Within each section the documentation of each procedure contains two parts. Firstly a description of the user interface and secondly the actual procedure code, together with a line-by-line discussion of the procedure.

Since many procedures have been defined to encode the algorithms a hierarchical system of folders and files has been employed to store these procedures. When the program is to be used, all of these procedures must be loaded into a MAPLE session. The most efficient method of achieving this is to load the internal MAPLE format of the procedures in one block. The following four steps details such a method.

1. Start a new MAPLE session, `maple`;
2. read in the individual procedures into this MAPLE session;
3. save this file in internal format, e.g. using `save 'all.m';`;
4. quit MAPLE using `quit`.

However, if the definition of one procedure is altered the above process must be repeated so that the changes are reflected in the internal format file.

6.2 The GCD controlling Procedures

The two procedures that calculate the greatest common left and right divisor of a matrix are documented below. A greatest common left divisor is calculated by invoking the greatest common right divisor procedure on the transpose of the matrix; the transpose of this result is then the greatest common left divisor.

Two modifications have been made to the GCRD Algorithm given in Chapter 4 to improve efficiency. The first is a test for coprimeness that is dependent on the matrix being primitive with respect to both indeterminates and possessing a 2-D Hermite form void of one indeterminate, see Theorem 4.3. The test is applied after Step 3, at which stage the primitive factorisation has been performed with respect to both indeterminates (Steps 1 and 2) and the 2-D Hermite form has been derived (Step 3). This test is particularly easy to implement due to the simplicity of the criterion: scanning the elements of the 2-D Hermite form to check for the absence of one of the indeterminates. The predefined functions in MAPLE further enhance the simplicity of the test due to the functions `convert` and `indets`. The former is used to convert the elements of the matrix to a set, i.e. a list of the elements with duplicates removed, and the latter computes the number of indeterminates contained the set. Therefore if the number of indeterminates is not equal to two the algorithm may be terminated. Then the result of the GCD procedure is the product of the primitive factors obtained in Steps 1 and 2. In this case one evaluation of the primitive factorisation is avoided.

The second efficiency modification occurs after the calculation of the 2-D Hermite form and before the second primitive factorisation. Through practical experience it was discovered that when the 2-D Hermite form is calculated the degree of the non-favoured indeterminate rapidly increases, therefore introducing many primitive roots. The purpose of Step 4 of the GCRD Algorithm is to remove these primitive roots. However, due to the triangular nature of the 2-D Hermite form some primitive roots occur as obvious factors of the individual rows. These primitive roots are removed before performing the primitive factorisation by scanning each row in turn for factors purely in the non-favoured indeterminate (recall that the primitive roots are defined by the zeros of 1-D polynomials). This process is beneficial to the procedure since the primitive factorisation procedure removes the primitive roots one by one. To illustrate this consider the following example.

Example 6.1: Suppose after Step 3 of the GCRD Algorithm the matrix M is

$$\begin{pmatrix} x+y & 1 \\ y(x+y) & x+y \\ x+y & 2 \end{pmatrix}$$

The 2-D Hermite form over $\mathbb{R}[y][x]$ is computed by executing `shiftherm(`M`,`x`);` resulting in the matrix

$$H = \begin{pmatrix} x+y-y^2-xy & 0 \\ 0 & 1-y \\ 0 & 0 \end{pmatrix}$$

Therefore the primitive roots are given by the solutions to the equation

$$(1-y)^2 = 0 \qquad \Rightarrow \qquad y = 1, 1$$

However, by performing the row scanning procedure above, both of these primitive roots are seen to be factors of the rows

$$H = \begin{pmatrix} 1-y & 0 & 0 \\ 0 & 1-y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x+y & 0 \\ 0 & x+y \\ 0 & 0 \end{pmatrix}$$

Thus by removing these factors the primitive factorisation is unnecessary. □

The above two modification have been incorporated to improve the efficiency of the program. The two modifications actually decrease the efficiency of the program when either the 2-D Hermite form is polynomial in both indeterminates or when there exist no factors in the rows, of the type described above. However, the potential increase in efficiency from these two processes greatly outweighs the actual loss of efficiency if either or both of the above situations occur.

6.2.1 The `gcld` Procedure

The Function:

Syntax: `gcld(`A`,`x`,'`U`');`

Parameters: A - a matrix in two indeterminates,
x - an indeterminate of the matrix A,
U - (optional) an unevaluated name.

The function requires at least two arguments, the first argument is a two dimensional array (or matrix) in two unevaluated names (indeterminates), the second is one of the indeterminates. The third optional argument returns the result of extracting the greatest common left divisor from the original matrix, i.e. a factor (left) coprime

matrix. The result of the function is the greatest common left divisor of the matrix A.

If the greatest common left divisor of two matrices $A_{m\times p}$ and $B_{m\times q}$ is required the argument to the function is the augmented matrix $[A\ B]$.

If z_1 and z_2 are the indeterminates of the matrix A and z_1 is specified as the indeterminate, the procedure operates over the ring $\mathbb{R}[z_2][z_1]$, although the algebraic closure of the ring $(\mathbb{C}[z_2][z_1])$ may be used during the primitive factorisation.

Procedure Code:

```
1 gcld:=proc(A,var,U)
2       local D,V;
3         if nargs>2 then D:=gcrd(transpose(A),var,'V');U:=transpose(V)
4           else D:=gcrd(transpose(A),var)
5           fi;
6         D:=transpose(D);
7         op(D)
8       end;
```

Formal Parameters:

`A` - a two dimensional array in two indeterminates;

`var` - the favoured indeterminate;

`U` - a place holder for the factor coprime matrix that results from extracting the greatest common left divisor from `A`.

Local Variables:

`D` - the result of the procedure;

`V` - the transpose of the factor coprime matrix obtained from extracting the greatest common left divisor from the matrix `A`.

Line 1 defines `gcld` as the name of the procedure.

Line 2 declares the local variables of the procedure.

Lines 3-5 calls the procedure `gcrd` with the same number of actual parameters as `gcld`. The first parameter is the transpose of the first matrix in the procedure invocation and the second is the indeterminate specified in the function invocation. Thus the matrix `U` is only calculated if it is specified in the procedure invocation.

Line 6 transposes the result of the `gcrd` procedure to obtain the greatest common left divisor of the matrix `A`.

Line 7 returns the elements of the matrix `D` as the value of the procedure.

Line 8 terminates the procedure.

6.2.2 The gcrd Procedure

The Function:

Syntax: `gcrd(`A`,`x`,'`U`');`

Parameters: A - a matrix in two indeterminates,
x - an indeterminate of the matrix A,
U - (optional) an unevaluated name.

The function requires at least two arguments, the first is a two dimensional array (or matrix) in two unevaluated names (indeterminates), the second is one of the indeterminates. The third optional argument returns the result of extracting the greatest common right divisor from the original matrix, i.e. a factor (right) coprime matrix. The result of the function is the greatest common right divisor of the matrix A.

Notice that only one matrix is given as the argument, if the greatest common right divisor of two matrices $A_{p\times m}$ and $B_{q\times m}$ is required the argument to the function is the augmented matrix $\begin{pmatrix} A \\ B \end{pmatrix}$.

If z_1 and z_2 are the indeterminates of the matrix A and z_1 is specified as the indeterminate the function works over the ring $\mathbb{R}[z_2][z_1]$, although the algebraic closure of the ring $(\mathbb{C}[z_2][z_1])$ may be used during the primitive factorisation.

Procedure Code:

```
1 gcrd:=proc(A,var,U)
2       local m,rdim,cdim,var1,D,H,M,R,R0,R1;
3         var1:=otherindet(A,var);
4         M:=rprimfac(A,var,'R0');
5         M:=rprimfac(M,var1,'R1');
6         H:=shiftherm(M,var);
7         if nops(indets(convert(op(H),set)))<>2 then D:=evalm(R1 &* R0)
8           if nargs>2 then U:=multiply(A & inverse(D)) fi;
9           RETURN(op(D))
10        fi;
11        rdim:=rowdim(H);
12        cdim:=coldim(H);
13        m:=min(rdim,cdim);
14        H:=rowfactors(H,var1)
15        R:=submatrix(H,1..m,1..m);
```

```
16        D:=evalm(lprimfac(R,var1) &* R1 &* R0);
17        if nargs>2 then U:=multiply(A,inverse(D)) fi;
18        op(D);
19      end;
```

Formal Parameters:

A - a two dimensional array in two indeterminates;

var - the favoured indeterminate;

U - a place holder for the factor coprime matrix that results from extracting the greatest common right divisor from A.

Local Variables:

m - the smallest of the number of rows and number of columns of A;

D - the greatest common right divisor of A;

H - the 2-D Hermite form of A with respect to $\mathbb{R}$[var1][var];

M - the right primitive factorisation of A with respect to, firstly $\mathbb{R}$[var][var1] and secondly $\mathbb{R}$[var1][var];

R - the square part of the 2-D Hermite form of A;

R0 - the primitive factor of A with respect to $\mathbb{R}$[var][var1];

R1 - the primitive factor of A with respect to $\mathbb{R}$[var1][var] when R0 has been removed;

rdim - the number of rows of A;

cdim - the number of columns of A;

var1 - the non-favoured indeterminate of A.

Line 1 defines gcrd as the procedure with formal parameters A, var, U.

Line 2 defines the variables local to this procedure.

Line 3 assigns var1 as the non-favoured indeterminate of A, by using the procedure otherindet.

Line 4 assigns the local variable M as the right primitive matrix of A with respect to the ring $\mathbb{R}$[var][var1] and sets R0 as the primitive factor that is extracted on the right over this ring.

Line 5 assigns the local variable M as the right primitive matrix of M with respect to the ring $\mathbb{R}$[var1][var] and sets R1 as the primitive factor that is extracted on the right over this ring.

Line 6 assigns the local variable H as the 2-D Hermite form of M with respect to the ring $\mathbb{R}$[var1][var] using the modified Hermite algorithm. This is equivalent to the 2-D Hermite algorithm by the discussion in Chapter 4.

Lines 7-10 decides if H is a matrix in two indeterminates, if it is not the procedure terminates on line 8 with the greatest common right divisor being returned

(see Chapter 4 for the theory behind this decision). If a third actual parameter occurs in the calling sequence `U` is assigned the factor coprime matrix that results from extracting the greatest common right divisor from `A`.

Line 11 assigns the local variable `rdim` the number of rows in the matrix `H`, which is the 2-D Hermite form of `M`.

Line 12 assigns the local variable `cdim` the number of columns in the matrix `H`.

Line 13 assigns the local variable `m` as the smallest of the number of rows and the number of columns in `H`.

Line 14 uses the `rowfactors` procedure to extract any factors from the rows of `H` that are purely polynomial in the indeterminate `var1`. This reduces the amount of work performed by the `lprimfac` procedure by reducing any obvious factors in `H`; thus making the procedure more efficient.

Line 15 assigns `R` as the top square submatrix of the 2-D Hermite form of `M`.

Line 16 assigns `D` as the greatest right common divisor of `A` by multiplying together the left primitive form of `R`, over $\mathbb{R}$[`var1`][`var`], and the right primitive factors `R1`, `R0` computed in lines 4-5.

Line 17 decides if the number of actual parameters with which the procedure was called is greater than two, if this is true the local variable `U` is assigned the factor (right) coprime matrix resulting from the extraction of the greatest common right divisor from `A`.

Line 18 returns the elements of `D` as the value of the procedure. As the last statement line of the procedure, the value of this statement is the value of the procedure (unless the explicit return was executed earlier).

Line 19 indicates the termination of the procedure.

Note: When an array or table is returned as the value of a procedure, the construct `op([[array or table]])` must be used so that the actual components of the array or table are returned and not the name [[array or table]].

6.3 The Primitive Factorisation Procedures

There are two controlling procedures for the primitive factorisation of a matrix. One is the right primitive factorisation and the other is the left primitive factorisation. These two procedures are documented first, with the procedures that are called by these procedures documented, in alphabetical order, subsequently.

Only one main procedure is required to compute the left and right primitive factorisation. To compute the right primitive factorisation, firstly the matrix is transposed and secondly the left primitive factorisation is calculated which returns the transpose of the required primitive matrix. Thus the right primitive factorisation procedure only has to perform transposition. This is similar to the greatest common left and right divisor situation described earlier.

6.3.1 The `rprimfac` Procedure

The Function:

Syntax: `rprimfac(`A`,`x`,'`R`');`

Parameters: A - a matrix in two indeterminates,
x - an indeterminate of the matrix A,
R - (optional) an unevaluated name.

The result of the function is the right primitive part of the matrix A with respect to $\mathbb{R}[x][y]$, where y is the non-favoured indeterminate, see Chapter 4 for the definition of a primitive matrix. The primitive factor, R, may be obtained by supplying a third actual parameter in the function invocation.

If the matrix $A(x,y)$ is the first parameter, x is the second and $'R'$ the third, i.e. `rprimfac(`A`,`x`,'`R`');` the result of the procedure is the primitive matrix A^* over $\mathbb{R}[x][y]$ such that

$$A = A^*R$$

and R is the primitive factor with the property $|R| \in \mathbb{R}[x]$.

Procedure Code:

```
1 rprimfac:=proc(A,var,R)
2       local Q,S;
3         Q:=lprimfac(transpose(A),var,'S');
4         if nargs>2 then R:=transpose(S) fi;
5         Q:=transpose(Q);
6         op(Q)
```

```
7       end;
```

Formal Parameters:

`A` - a two dimensional array in two indeterminates;

`var` - the favoured indeterminate;

`R` - a place holder for the right primitive factor.

Local Variables:

`Q` - initially the transpose of the right primitive form of `A` and subsequently the result of the procedure;

`S` - the transpose of the right primitive factor of `A`.

Line 1 defines `rprimfac` as the procedure with formal parameters `A`, `var` and `R`.

Line 2 defines the variables `Q` and `S` as local variables.

Line 3 assigns `Q` as the left primitive form and `S` as the left primitive factor of the transpose of `A`.

Line 4 decides if the number of actual parameters in the function call is greater than two, if so `R` is assigned the transpose of the left primitive factor of the transpose of `A`.

Line 5 `Q` is reassigned as the transpose of itself.

Line 6 is the value of the procedure, as it is the last statement in the procedure.

Line 7 terminates the procedure.

6.3.2 The `lprimfac` Procedure

The Function:

Syntax: `lprimfac(`A`,`x`,'`R`')`;

Parameters: A - a matrix in two indeterminates,

x - an indeterminate of the matrix A,

R - (optional) an unevaluated name.

The result of the function is the left primitive part of the matrix A with respect to the ring $\mathbb{R}[x][y]$ where y is the non-favoured indeterminate. If a third actual parameter is present it is assigned the left primitive factor of the matrix A.

Procedure Code:

```
1 lprimfac:=proc(A,var,R)
2       local i,j,s,row,rdim,cdim,var1,B,C,M,Q,U;
3          rdim:=rowdim(A);
4          cdim:=coldim(A);
5          M:=copy(A);
6          Q:=unitmx(rdim);
```

```
7         s:=primroots(M,var);
8         var1:=otherindet(M,var);
9         for i to nops(s) do
10          B:=subs(var=s[i],op(M));
11          B:=emxc(B);
12          shiftherm(B,var1,'U');
13          M:=multiply(U,M);
14          row:=factorow(M,var,s[i]);
15          C:=unitmx(rdim);
16          C[row,row]:=var-s[i];
17          for j to cdim do
18            M[row,j]:=simplify(evalc(M[row,j]/(var-s[i])))
19          od;
20          Q:=evalm(Q &* inverse(U) &* C)
21        od;
22        Q:=emxc(Q);
23        if nargs>2 then R:=copy(Q) fi;
24        emxc(M)
25      end;
```

Formal Parameters:

`A` - a two dimensional array in two indeterminates;

`var` - the favoured indeterminate;

`R` - a place holder for the right primitive factor.

Local Variables:

`i` - index of iteration;

`j` - index of iteration;

`s` - the list of primitive roots of `A` in the indeterminate `var`;

`row` - an integer denoting the row number of the modified Hermite form that contains the factor `var-s[i]`;

`rdim` - the number of rows in `A`;

`cdim` - the number of columns in `A`;

`var1` - the non-favoured indeterminate of `A`;

`B` - the matrix that is obtained from substituting a primitive root for the indeterminate `var`;

`C` - the multiplying matrix that contains the factor `var-s[i]` for each of the primitive roots, i.e. the diagonal matrix in Algorithm 4.3 Step 2;

M - used to store the results of intermediate steps in the procedure.

Q - stores the primitive factor, i.e. the matrix $A_1A_2\ldots A_i$ at each iteration (see Algorithm 4.3 for the definition of A_i);

U - the unimodular matrix over $\mathbb{R}$[var] that effects the modified Hermite form, see Algorithm 4.2.

Line 1 defines `lprimfac` as the procedure with formal parameters `A`, `var` and `R`.

Line 2 declares the local variables as listed above.

Line 3 assigns the local variable `rdim` the number of rows in the matrix `A`.

Line 4 assigns the local variable `cdim` the number of columns in the matrix `A`.

Line 5 assigns `M` as a copy of `A`, see Chapter 5 for the reason for using `copy`.

Line 6 assigns `Q` as the unit matrix with dimension `rdim` × `rdim`.

Line 7 assigns `s` as the list of primitive roots of the matrix `M` in the indeterminate `var`.

Line 8 assigns `var1` as the non-favoured indeterminate of the matrix `A`.

Line 9 sets the repetition statement so that each of the primitive roots is extracted from `A`.

Line 10 substitutes a primitive root for the indeterminate `var` in the components of `M`. Notice that in the function invocation `op(M)` is used and not `M`, because the substitution is required in the components of `M` and not in the name `M`.

Line 11 uses the function `emxc` to simplify the components of `B` and reassigns `B` as this simplified matrix.

Line 12 computes the modified Hermite form of `B` over $\mathbb{C}$[`var1`]. The complex numbers have to be used in this instance because some of the primitive roots may be complex. Here it is not the modified Hermite form that is required but the multiplying matrix, `U`, that effects the modified Hermite form; thus the Hermite form is not assigned a name.

Line 13 reassigns `M` as `U`, as defined in line 12, multiplied by `M` calculated earlier (if `i=1` `M` is the matrix `A` otherwise it is the matrix `A` with primitive roots `s[1]...s[i-1]` removed).

Line 14 assigns `row` as the row number in `M` that contains the factor `var−s[i]`.

Lines 15-16 assigns `C` as the unit matrix, dimension `rdim` × `rdim`, with the '1' in position (`row`,`row`)replaced the factor `var−s[i]`.

Lines 17-19 divides the factor `var−s[i]` from each term in row `row` and reassigns `M` as the result.

Line 20 assigns `Q` as the product of the primitive factors, i.e. $A_1A_2\ldots A_i$ in the Algorithm 4.3.

Line 21 terminates the repetition statement on line 9.

Line 22 simplifies each component of the primitive factor matrix `Q`. The simplification is performed after all the primitive factors have been multiplied together in the repetition statement in an attempt to use as little CPU time as possible, because a call to `emxc` simplifies each term sequentially using `evalc` (a total number of `rdim` × `cdim` evaluations).

Line 23 assigns `R` a copy of the matrix `Q` if `lprimfac` has more than two formal parameters, i.e. `lprimfac` was invoked with more than two arguments.

Line 24 is the last statement in the procedure, thus the procedure takes the value of this last statement, which evaluates to the primitive part of the matrix `A`. The function call `emxc` is used in this line so that the elements of the matrix `M` containing complex coefficients are expressed in the form $a + \mathtt{I}b$.

Line 25 terminates the procedure.

6.3.3 The `factorow` Procedure

The Function:

Syntax: `factorow(`A`,`x`,`r`);`

Parameters: A - a matrix,
x - an unevaluated name contained in the matrix A,
r - an algebraic number.

The result of the procedure is an integer denoting the row in which all elements are divisible by the polynomial $x - r$.

Procedure Code:

```
1 factorow:=proc(A,var,root)
2       local i,Z,rdim,row;
3         rdim:=rowdim(A);
4         row:=0;
5         Z:=subs:=(var=root,op(A));
6         for i from rdim by -1 to 1 do
7           if zerow(A,i)=false and zerow(Z,i)=true then
8             RETURN(i)
9           fi
10        od;
11        row
12      end;
```

Formal Parameters:

`A` - a two dimensional array;
`var` - an unevaluated name representing an indeterminate of `A`,
`root` - an algebraic number.
Local Variables:
`i` - an iteration index;
`Z` - a simplified polynomial with complex coefficients;
`rdim` - an integer denoting the number of rows in `A`;
`row` - an integer denoting the row number with the factor `poly`;
Line 1 defines `factorow` as the name of the procedure.
Line 2 declares the local variables.
Line 3 assigns the number of rows in `A` to `rdim`.
Line 4 assigns zero to `row`.
Line 5 assigns `Z` the matrix `A` with all occurrences of the name `var` with the number `root`.
Lines 6-10 Searches for the row of `Z` in which the polynomial `var-root` is a factor of all the elements in the row with the corresponding row in the matrix `A` is non-zero, so that `var-root` is a true factor. The search starts with the last row of the matrix because in the primitive factorisation procedure `lprimfac` the last row contains the factor `var-root`. The condition that the corresponding row of `A` is non-zero is required in case the matrix `A` does not have full rank. The result of the procedure is the row number that contains the factor `var-root`.
Line 11 terminates the procedure.

6.3.4 The `gcdminors` Procedure

The Function:

Syntax: `gcdminors(`*A*`);`
Parameter: *A* - a matrix.

The procedure calculates the greatest common divisor of all the high-order minors of the matrix *A*.

Procedure Code:

```
1 gcdminors:=proc(A)
2       local m,g;
3         m:=highorderminors(A);
4         g:=gcdn(m)
5       end;
```

Formal Parameters:

`A` - a two dimensional array.

Local Variables:

`m` - the high-order minors of the matrix `A`;

`g` - the greatest common divisor of the high-order minors of `A`.

Line 1 defines `gcdminors` as the procedure name.

Line 2 declares the local variables of the procedure.

Line 3 assigns `m` the value of the procedure `highorderminors`, which calculates the high-order minors of the matrix `A`.

Line 4 assigns `g` the value of the procedure `gcdn`, which calculates the greatest common divisor of a list or set of polynomials. This is the value of the procedure as it is the last statement in the procedure.

Line 5 terminates the procedure.

6.3.5 The `gcdn` Procedure

The Function:

Syntax: `gcdn(`*m*`);`

Parameter: *m* - a set or list of polynomials.

The result of the function is the greatest common divisor of the elements of the list or set *m*.

Procedure Code:

```
1 gcdn:=proc(m)
2       local g,i;
3         g:=op(1,m);
4         for i from 2 to nops(m) do
5           g:=gcd(g,op(i,m))
6         od;
7         g
8       end;
```

Formal Parameters:

`m` - a set or list of polynomials.

Local Variables:

`g` - the greatest common divisor of the first `i` elements;

`i` - iteration index.

Line 1 defines `gcdn` as the name of the procedure.

Line 2 declares the local variables of the procedure.

Line 3 assigns g as the first element of m.

Line 4 defines the repetition statement so that all the elements of the set or list m are accessed.

Line 5 assigns g as the greatest common divisor of g and the ith element of m.

Line 6 terminates the repetition statement.

Line 7 returns g as the value of the procedure.

Line 8 terminates the procedure.

6.3.6 The highorderminors Procedure

The Function:

Syntax: `highorderminors(`A`);`

Parameter: A - a matrix.

The result of the procedure is a set of polynomials that represent the high-order minors of the matrix A, i.e. the $p \times p$ minors of $A_{p \times q}$ where $p \geq q$.

Procedure Code:

```
1 highorderminors:=proc(A)
2       local i,j,n,p,q,P,Q,M;
3         p:=rowdim(A);
4         q:=coldim(A);
5         n:=min(p,q);
6         P:=combine(p,n);
7         Q:=combine(q,n);
8         for i to op(P) do
9           for j to op(Q) do
10            M[i,j]:=det(submatrix(A,P[i],Q[j]))
11          od;
12        od;
13        M:=convert(M,set);
14        M;
15        end;
```

Formal Parameters:

A - a matrix.

Local Variables:

i - iteration index;

j - iteration index;

`n` - the minimum of the number of rows and the number of columns of the matrix `A`;

`p` - the number of rows of the matrix `A`;

`q` - the number of columns of the matrix `A`;

`P` - the list of the ways of choosing `n` from the first `p` natural numbers;

`Q` - the list of the ways of chosing `n` from the first `q` natural numbers;

`M` - the high-order minors of the matrix `A`.

Line 1 defines `highorderminors` as the name of the procedure.

Line 2 declares the local variables of the procedure.

Line 3 assigns `p` the number of rows of the matrix `A`.

Line 4 assigns `q` the number of columns of the matrix `A`.

Line 5 assigns `n` the minimum of the number of rows and the number of columns of the matrix `A`.

Line 6 assigns to `P` the list of all combinations of chosing `n` of the first `p` natural numbers.

Line 7 assigns to `Q` the list of all combinations of chosing `n` of the first `q` natural numbers.

Line 8 defines the repetition statement so that all elements of the list `P` are accessed.

Line 9 defines the repetition statement so that all elements of the list `Q` are accessed.

Line 10 assigns to `M[i,j]` the determinant of the submatrix formed by taking the rows as specified by the `i`th element of the list `P` and the columns as specified by the `j`th element of the list `Q`. Note that one of these lists will consist of exactly one term because `n` is either `p` or `q`. Thus all the high-order minors are calculated.

Line 11 terminates the repetition statement for accessing the elements of `Q`.

Line 12 terminates the repetition statement for accessing the elements of `P`.

Line 13 converts the elements of the table `M` to a set. Thus eliminating any duplicates.

Line 14 returns `M` as the value of the procedure.

Line 15 terminates the procedure.

6.3.7 The otherindet Procedure

The Function:

Syntax: `otherindet(`A`,`x`);`

Parameters: A - a matrix in two indeterminates,
x - an indeterminate of the matrix A.

The matrix A must have two indeterminates, one of which is x. The result of the procedure is then the other indeterminate.

Procedure Code:

```
1 otherindet:=proc(A,var)
2       local i,j,s;
3         s:=convert(A,set);
4         i:=indets(s);
5         if nops(i)<>2 then RETURN('procname(args)') fi;
6         for j to 2 do
7           if i[j]<>var then RETURN(i[j]) fi;
8         od
9       end;
```

Formal Parameters:

`A` - a two dimensional array;

`var` - an unevaluated name.

Local Variables:

`i` - a set containing the indeterminates of the matrix `A`;

`j` - an iteration index;

`s` - a set of the elements of the matrix `A`.

Line 1 defines the procedure name as `otherindet`.

Line 2 declares the local variables.

Line 3 converts the elements of the two dimensional array `A` to a set and assigns the result to `s`. The data structure `set` is used so that duplicates are removed from the set.

Line 4 the set of unevaluated names in the set `s` are assigned to `i`.

Line 5 if the number of indeterminates of the set `s` is not equal to two then the procedure name and the arguments with which it was called are returned as the value of the procedure. In effect this is an error return.

Line 6 defines the repetition statement that scans both elements of the set `i`.

Line 7 if the jth element of the set i is not the indeterminate specified the value of the procedure is the value of this element, i.e. the other indeterminate of the matrix A.

Line 8 terminates the repetition statement.

Line 9 terminates the procedure.

6.3.8 The primroots Procedure

The Function:

Syntax: `primroots(`A`,`x`);`

Parameters: A - a matrix in the indeterminate x,
x - an indeterminate of the matrix A.

The result of the procedure is a list of values of the indeterminate x that correspond to the primitive roots of the matrix A.

Procedure Code:

```
1 primroots:=proc(A,var)
2       local i,m,n,p,q,r,s,v;
3         p:=rowdim(A);
4         q:=coldim(A);
5         n:=rank(A);
6         if min(p,q)<>n then ERROR fi;
7         m:=gcdminors(A);
8         v:=varfac1(m,var);
9         if v=[] then RETURN ([]) else RETURN(roots(v,var)) fi
10      end;
```

Formal Parameters:

A - a two dimensional array;
var - an unevaluated name.

Local Variables:

m - a polynomial denoting the greatest common divisor of the high-order minors of A;
n - the rank of the matrix A;
p - the number of rows in A;
q - the number of columns in A;
v - a list of polynomials over the ring $\mathbb{C}$[var], i.e. polynomials in the indeterminate var with coefficients over the complex field.

Line 1 defines the name of the procedure as primroots.

Line 2 declares the local variables.

Line 3 assigns p the number of rows in A.

Line 4 assigns q the number of columns in A.

Line 5 assigns n the rank of A.

Line 6 if the matrix A does not have full rank then an error is returned.

Line 7 assigns m as the greatest common divisor of the high-order minors of the matrix A.

Line 8 assigns to v a list of factors of m that are in the ring $\mathbb{C}$[var].

Line 9 if the list v is empty, the empty list is returned as the value of the procedure. Otherwise the value of the procedure is a list of numbers that are the roots of the polynomials contained in the list v.

Line 10 terminates the procedure.

6.3.9 The roots Procedure

The Function:

Syntax: `roots(`a`,`x`);`

Parameters: a - a set, list, table or array of polynomials,
x - an indeterminate.

The result of the function is a list of roots of the polynomials contained in the expression a.

Procedure Code:

```
1 roots:=proc(f,var)
2       local i,u,v,w;
3         u:=convert(f,list);
4         for i to nops(u) do
5           v[i]:=solve(u[i],var)
6         od;
7         v:=convert(v,list);
8         for i to nops(v) do
9         if type(v[i],RootOf)=true
10          then w[i]:=allvalues(v[i])
11          else w[i]:=v[i]
12          fi;
13        od;
14        convert(w,list);
15      end;
```

Formal Parameters:

`f` - a data structure of the type set, list, table or array;

`var` - an unevaluated name.

Local Variables:

`i` - iteration index;

`u` - the elements of `f` converted to a list;

`v` - a list of roots to the elements of `u`;

`w` - the result of the procedure - a list of numbers.

Line 1 defines `roots` as the name of the procedure.

Line 2 declares the local variables of the procedure.

Line 3 converts the elements of `f` to a list and assigns the result to `u`.

Line 4 defines the repetition statement that accesses all the elements in the list `u`.

Line 5 assigns the `i`th element of `v` as the roots of the polynomial `u[i]`.

Line 6 terminates the repetition statement.

Line 7 converts `v` to a list. This is necessary because there may be more than one root of the polynomials contained in `u`, thus some elements of `v` may be a sequence of numbers. The construct `convert(v,list)` assigns one expression to each element of `v`.

Line 8 defines the repetition statement to access all of the elements of the list `v`.

Line 9 tests the `i`th element of `v` for the data structure `RootOf`. The result of `solve` may sometimes be of this type when the polynomial has a high order or the solution involves complicated algebraic numbers. The numbers become complicated because `solve` returns exact numbers, i.e. expressed as fractional powers of integers.

Line 10 `w[i]` is assigned the solution when `RootOf` is fully evaluated to numeric values. Thus the solution may be a sequence of numbers.

Line 11 assigns `w[i]` the value of `v[i]` if `v[i]` is not of the data structure `RootOf`, i.e. a number.

Line 12 terminates the conditional statement.

Line 13 terminates the repetition statement.

Line 14 returns a list of all the values contained by the name `w`. The construct `convert` is used for the same reason as described on line 7.

Line 15 terminates the procedure.

6.3.10 The rowfactors Procedure

The Function:

Syntax: `rowfactors(`H`,`x`);`

Parameters: H - a matrix,
x - an indeterminate.

The result of the function is a matrix with any factors common to a row, purely in the indeterminate x, removed. Each row is scanned individually for factors purely in the given indeterminate, these are removed by dividing each of the terms in that row by the factor.

Procedure Code:

```
1 rowfactors:=proc(A,var)
2       local c,i,j,m,rdim,cdim,B;
3         rdim:=rowdim(A);
4         cdim:=coldim(A);
5         B:=copy(A);
6         for i to rdim do
7           if zerow(B,i)=true then next fi;
8           for j to cdim do
9             if B[i,j]<>0 then m[j]:=B[i,j] fi
10          od;
11          m:=convert(m,set);
12          m:=gcdn(m);
13          c:=content(m);
14          m:=primpart(m);
15          m:=convert(varfac1(m,var),list);
16          if m=[] then m:=1
17          elif nops(m)>1 then m:=convert(m,'*') else m:=m[1] fi;
18          m:=m*c;
19          for j to cdim do
20            if B[i,j]<>0 then B[i,j]:=simplify (B[i,j]/m) fi
21          od
22        od;
23        op(B)
24      end;
```

Formal Parameters:

A - a two dimensional array;
var - an unevaluated name.
Local Variables:
c - a constant;
i - index of iteration;
j - index of iteration;
m - the gcd of each row of the matrix A;
rdim - the number of rows in A;
cdim - the number of columns in A;
B - the result of the procedure.

Line 1 defines **rowfactors** as the name of the procedure.

Line 2 declares the local variables.

Line 3 assigns rdim as the number of rows in A.

Line 4 assigns cdim as the number of columns in A.

Line 5 assigns B as a copy of A. The matrix B is then only altered if a factor is removed from one of the rows.

Line 6 defines the repetition statement to access each of the rows.

Line 7 test for the current row having only zeros as its elements. If the test is true then the next iteration is carried out.

Line 8 defines the repetition statement so that all the elements within each row are accessed.

Line 9 tests for the current element being non-zero. If the test is true then the (i,j)th element is added to a one dimensional table of previous non-zero elements in row i.

Line 10 terminates the repetition statement.

Line 11 converts the table of non-zero elements to a set. Thus unassigned elements and duplicates are removed form the table m.

Line 12 extracts the greatest common divisor of the non-zero elements in row i.

Line 13 assigns to c a multiplicative constant present in the greatest common divisor. This value has to be stored separately because only pure polynomials are returned from varfac1 and not constants. It is not entirely necessary to extract a constant from the non-zero elements of each row but it does prevent large constants building up within the elements.

Line 14 reassigns m as the primitive part of the polynomial m calculated in line 12, i.e. the polynomial obtained by dividing by the result of the previous line.

Line 15 returns, as a list, those factors within the greatest common divisor that are polynomial in the indeterminate var.

Line 16 tests for `m` being empty, in which case the row idoes not possess any polynomial factors. If the test is true the procedure moves to the next iteration.

Line 17 tests for more than one factor in the list `m`. If the test is true the list is transformed into a product of the terms, thus recombining all the factors of the greatest common divisor polynomial purely in the indeterminate **var**. Otherwise `m` is assigned the only factor of the list.

Line 18 multiplies the polynomial factor by the constant extracted from the greatest common divisor of the elements of row `i`.

Line 19 defines the repetition statement to access all the elements in row `i`.

Line 20 tests for non-zero elements in row `i`. If the test is true the element is divided by the polynomial factor.

Line 21 terminates the repetition statement.

Line 22 terminates the row repetition statement.

Line 23 returns the matrix with the row factors removed.

Line 24 terminates the procedure.

6.3.11 The varfac1 Procedure

The Function:

Syntax: `varfac1(`p`,`x`);`

Parameters: p - a polynomial,
x - an indeterminate of the polynomial p.

The result of the function is a list of polynomial factors of p that are polynomial in the indeterminate x only. The function factors the polynomial p and then tests each factor for the required form.

Procedure Code:

```
1 varfac1:=proc(poly,var)
2       local i,p,r,j;
3         r:=factor(poly);
4         if type(r,'*')=true then
5         elif type(r,polynom,var,numeric)=true and type(r,numeric)=false
6           then RETURN([r]) else RETURN([])
7         fi;
8         j:=0;
9         p:=[];
10        for i to nops(r) do
11          if type(op(i,r),polynom,var,numeric)=true
```

```
12              and type(op(i,r),numeric)=false
13              then j:=j+1;p[j]:=op(i,r)
14            fi
15          od;
16          if p=[] then RETURN(p) else RETURN(op(p)) fi
17        end;
```

Formal Parameters:

`poly` - a polynomial;

`var` - an unevaluated name.

Local Variables:

`i` - iteration index;

`j` - an integer;

`p` - the polynomial factors in the indeterminate `var`;

`r` - the factors of the polynomial `poly`.

Line 1 defines `varfac1` as the name of the procedure.

Line 2 declares the local variables of the procedure.

Line 3 assigns `r` the product of the factors of `poly`.

Line 4 tests for `r` to be a product of terms if this is false then lines 5-7 are executed otherwise line 8 is the next line. If `r` contains only one factor its type is not '*' thus only that one term is tested for the required form. If `r` is of the form '*' then each factor is tested separately.

Line 5 tests for the single factor `r` to be polynomial in the indeterminate `var` with coefficients that are numbers, i.e. not polynomials in any other indeterminates, and is not purely a number.

Line 6 if the test on line 5 is true then `r` is returned as the value of the function. Otherwise `poly` contains no factor of the required form and the empty list is returned.

Line 7 terminates the conditional statement.

Line 8 assigns zero to `j`.

Line 9 assigns `p` the empty list.

Line 10 defines the repetition statement to access each factor of the polynomial `r`.

Lines 11-12 tests for the `i`th factor of `r` to be polynomial in the indeterminate `var` with coefficients that are numbers, i.e. not polynomials in any other indeterminates, and is not purely a number.

Line 13 if the test on lines 11-12 is true then the counter `j` is incremented by one and the `j`th element of `p` is assigned the `i`th factor of `r`.

Line 14 terminates the conditional statement.

Line 15 terminates the repetition statement.

Line 16 if there exists no factors of r that are of the required form the empty list is returned as the value of the procedure. Otherwise the elements of the list p are returned as the value of the procedure.

Line 17 terminates the procedure.

6.3.12 The zerow Procedure

The Function:

Syntax: `zerow(`A`,`i`);`

Parameters: A - a matrix,
i - an integer.

The result of the procedure is `true` if row i has zero as every element in the row and `false` otherwise.

Procedure Code:

```
1 zerow:=proc(A,row)
2       local i;
3         for i to coldim(A) do
4           if radsimp(A[row,i])<>0 then RETURN(false) fi
5         od;
6         RETURN(true)
7       end;
```

Formal Parameters:

`A` - a two dimensional array;

`row` - an integer.

Local Variables:

`i` - an iteration index.

Line 1 defines `zerow` as the name of the procedure.

Line 2 declares the local variables of the procedure.

Line 3 defines the repetition statement to scan each element of row `row` in the matrix `A`.

Line 4 if element (`row`,`i`) of `A` is non-zero then then the value of the procedure is `false` and the procedure is terminated. The function `radsimp` is used to simplify radicals that occur in the matrix elements.

Line 5 terminates the repetition statement. The column index is increased by one and the next element is tested.

Line 6 returns the value `true`. This statement is only executed if all the elements in row `row` are zero, i.e. `RETURN(false)` has not been executed on line 4.

Line 7 terminates the procedure.

6.4 The Modified Hermite Procedures

This set of procedures is used in two different ways. The first is for singular matrices in one indeterminate, invoked in the primitive factorisation procedures; the second is for the 2-D Hermite form of a full rank matrix, in Step 4 of the GCD Algorithm. The procedures can be used in this way for the following three reasons.

1. The latter is polynomial in two indeterminates whilst the former is polynomial in only one indeterminate. However, the MAPLE procedure only requires one of the indeterminates to be explicitly specified, therefore the matrix argument in the procedure may contain any number of unevaluated names (indeterminates) in its elements.
2. The latter performs the triangularisation using the pseudo-division algorithm and the elementary operations over a generalised polynomial ring, such as $F[z_1][z_2]$ whilst the former uses the division algorithm and elementary operations over a Euclidean ring, such as $F[z_2]$. These differences do not affect the MAPLE procedure because the division algorithm over a Euclidean ring is a special case of the pseudo-division algorithm and the elementary operations are effectively equivalent (in the first case the coefficient field is taken to be $F[z_1]$ and in the second to be F).
3. By the discussion in Chapter 4 the modified Hermite algorithm and the 2-D Hermite algorithm are equivalent for full rank matrices, irrespective of the number of indeterminates.

For these reasons, only one MAPLE procedure is required to compute both the 2-D Hermite form and the modified Hermite form.

The name used in the MAPLE procedure to compute the modified Hermite form is `shiftherm` for which two parameters are necessary. The first is a matrix and the second an indeterminate of the matrix. A third parameter may also be specified, which is assigned the elementary operations used to effect the modified Hermite form. This third parameter is necessary for the primitive factorisation procedure. The two matrices are stored at each stage of the `shiftherm` procedure as a list; this allows both matrices to be passed between procedures as one parameter. Initially the pair consists of the matrix argument and the unit matrix, of appropriate dimensions. Thus the elementary operations used to derive the modified Hermite form are stored by performing the same operations on the unit matrix.

One further feature of the modified Hermite MAPLE procedure concerns the representation of complex numbers, `a+Ib`. These may be introduced in the calculation of

the primitive factorisation. This causes problems in some of the predefined functions because I is not treated on the same basis as other unevaluated names due to the property that $I^2 = -1$. For example, the predefined function to calculate the rank of a matrix, `rank`, returns an error message when the elements of the matrix involve complex coefficients. Therefore a new procedure, `Rank`, has been defined to calculate the rank of a matrix based on minor expansion, i.e. the rank of a matrix, r, is defined to be the largest integer such that there exists a non-zero rth order minor but all $r + 1$th order minors are zero. To overcome the seemingly unpredictable properties of complex numbers, I is substituted for the unevaluated name `compl` and is subsequently treated as an indeterminate. However, this does not increase the complexity of the algorithm since rational expressions in `compl` are permitted. When the desired modified Hermite form has been computed the unevaluated name `compl` is substituted for the complex number I and each element of the matrix is brought to standard complex number notation (`a+Ib`) by using the complex number simplifier, `evalc`; thus the rational expressions in `compl` are eliminated.

By using the pseudo-division algorithm and elementary operations the upper triangular modified Hermite procedure is derived by firstly reducing the matrix to upper triangular form (`gaussredS`) and then forcing the column properness condition. For singular matrices the definitions of upper triangular and column properness have to be modified to accommodate the zero pivotal columns, i.e. columns in which a pivot element failed to be located. Recall that the positions occupied by the first non-zero element in each row is termed the quasi-principal diagonal. A quasi-upper triangular matrix is then defined to be the matrix with elements that are zero below the quasi-principal diagonal and the quasi-column degree condition is defined by the elements on the quasi-principal diagonal having degree greater than any other element in the column. Therefore the zero pivotal columns are effectively ignored both in the derivation of the upper triangular form and also in the column properness condition. The code that effects the computation is described below.

6.4.1 The shiftherm Procedure

The Function:

Syntax: `shiftherm(`A`,`x`,`U`);`

Parameters: A - a matrix in at least one indeterminate,
x - an indeterminate of the matrix A,
U - (optional) an unevaluated name.

The result of the function is the modified Hermite form over $F[x]$, where the coefficient field F may be polynomial. If a third actual parameter is specified it is assigned the elementary operations stored as a matrix.

Procedure Code:

```
1 shiftherm:=proc(A,var,U)
2       local compl,mx;
3         mx:=[subs(I=compl,op(A)),unitmx(rowdim(A))];
4         mx:=gaussredS(mx,var);
5         mx:=colredS(mx,var);
6         if nargs>2 then U:=emxc(subs(compl=I,mx[2])) fi;
7         emxc(subs(compl=I,mx[1]))
8       end;
```

Formal Parameters:

`A` - a two dimensional array in at least one unevaluated name;
`var` - an unevaluated name of `A`;
`U` - an unevaluated name.

Local Variables:

`compl` - an unevaluated name used to substitute for the complex number `I`;
`mx` - a list that contains `A` as its first element and the elementary operations matrix as its second.

Line 1 defines `shiftherm` as the name of the procedure.

Line 2 declares the local variables of the procedure.

Line 3 assigns to the name `mx` a list of two matrices. The first matrix is `A`, with the complex number `I` substituted for the unevaluated name `compl`, and the second matrix is the unit matrix with the same number of rows as possessed by `A`, this will become the elementary operations matrix.

Line 4 `mx` is reassigned as a list with two elements, which are the result of the procedure `gaussredS`.

Line 5 `mx` is reassigned the result of the procedure `colredS`.

Line 6 assigns to U the elementary operations matrix, with unevaluated name compl substituted for the complex number I and then simplified by using the procedure emxc, so that any complex numbers contained in U are brought to their simplest form.

Line 7 the result of the procedure is the matrix obtained as the first element of the list mx, with the unevaluated name compl substituted for the complex number I and simplified using the procedure emxc.

Line 8 terminates the procedure.

6.4.2 The gaussredS Procedure

The Function:

Syntax: gaussredS(a,x);

Parameters: a - a list with two matrix elements,
x - an indeterminate of the first element of a.

The two matrices contained in the list a are, firstly, the matrix that is to be reduced and, secondly, the matrix that is to contain the elementary operations. The result of the procedure is a list with two matrix elements. The first of which is quasi-upper triangular and the second is a matrix that stores the elementary operations.

Procedure Code:

```
1 gaussredS:=proc(mx,var)
2       local j,cdim,rdim,MX,rk,shift;
3         MX:=copy(mx);
4         rdim:=rowdim(MX[1]);
5         cdim:=coldim(MX[1]);
6         rk:=Rank(MX[1]);
7         shift:=0;
8         for j to max(cdim,rdim) do
9           if rk-shift<j then break fi;
10          if upperS(MX[1],j,j+shift-1)=true then shift:=shift-1 fi;
11          while upperS(MX[1],j,j+shift)=false do
12            MX:=initialpivotS(MX,var,j,j+shift,rdim);
13            MX:=lowerzerosS(MX,var,j,j+shift,rdim)
14          od
15        od;
16        MX
17      end;
```

Formal Parameters:

`mx` - a two element list of matrices;

`var` - an indeterminate of the first element of `mx`.

Local Variables:

`j` - an index of iteration;

`rk` - the rank of `mx[1]`;

`cdim` - the number of columns of `mx[1]`;

`rdim` - the number of rows of `mx[1]`;

`shift` - a negative number with modulus the number of zeros on the pseudo-principal diagonal (Chapter 4);

`MX` - the intermediate results of the procedure.

Line 1 defines `gaussredS` as the name of the procedure.

Line 2 declares the local variables of the procedure.

Line 3 assigns `MX` as a copy of `mx`.

Line 4 assigns `rdim` the number of rows of the first element of the list `MX`.

Line 5 assigns `cdim` the number of columns of the first element of the list `MX`.

Line 6 assigns the rank of the matrix `mx[1]` to `rk`.

Line 7 assigns `shift` the number `0`. Initially the number of elements on the pseudo-principal diagonal is zero.

Line 8 defines the repetition statement. The index `j` accesses each column until either the iteration terminates by exhausting all the iterates or the break on line 8 is executed.

Line 9 a break is executed if the number of rows plus the number of zeros on the pseudo-principal diagonal exceeds the column index there it is not necessary to continue and a `break` is executed to exit from the iterative statement.

Line 10 determines if all the elements below and on the pseudo-diagonal are zero. If this is true the counter `shift` is decreased by one (decrementation is used because the counter `shift` is a negative number).

Lines 11-14 executes the two statements on lines 12-132 until all the elements below the pseudo-principal diagonal are zero. Line 12 finds a suitable pivot and line 12 reduces the degree, in `var`, of all the elements below the quasi-principal diagonal.

Line 15 increments `j` by one and continues with the next iteration.

Line 16 returns the list of matrices as the result to the procedure.

Line 17 terminates the procedure.

6.4.3 The initialpivotS procedure

The Function:

Syntax: `initialpivotS(`a`,`x`,`c`,`k`,`r`);`

Parameters:
- a - a list of two matrices, say $[A, V]$,
- x - an indeterminate of the matrix A,
- c - a column number of the matrix A,
- k - an integer denoting a row of the matrix A,
- r - an integer denoting the number of rows in A.

The procedure scans column c from rows k to r, inclusive, for the element with lowest degree in the indeterminate x and then swaps that row with row k. The matrix V contains the elementary operations.

Procedure Code:

```
1 initialpivotS:=proc(mx,var,col,k,rdim)
2      local i,row,testelem,pivot,M,V;
3        M:=copy(mx[1]);
4        V:=copy(mx[2]);
5        row:=0;
6        for i from k to rdim do
7          testelem:=deg(M[i,col],var);
8          if row=0 and testelem<>-1 then
9            pivot:=testelem;
10           row:=i;
11         fi;
12         if testelem>=0 and testelem<pivot then
13           pivot:=testelem;
14           row:=i;
15         fi;
16       od;
17       if row>k then
18         M:=swaprow(M,row,k);
19         V:=swaprow(V,row,k)
20       fi;
21       [op(M),op(V)]
22     end;
```

Formal Parameters:

mx - a list of two matrices;
var - an unevaluated name;
col - an integer;
k - an integer;
rdim - an integer.

Local Variables:

i - an index of iteration;
row - an integer denoting the row number of the row containing the element with least degree in var.
testelem - an integer denoting the degree of the element under test at the current iteration.
pivot - an integer denoting the least degree in var of the elements scanned.
M - part of the result of the procedure; it has the element in column col with least degree, in var, in the pivotal position.
V - the elementary operation of swapping the rows.

Line 1 defines initialpivotS as the name of the procedure.
Line 2 declares the local variables.
Line 3 assigns M the first matrix in the list mx.
Line 4 assigns V the second matrix in the list mx.
Line 5 assigns row the integer zero.
Line 6 defines the repetition statement to scan the rows k to rdim of column col.
Line 7 assigns testelem the degree of the polynomial in position (i,col) with respect to the indeterminate var.
Line 8 tests for row being zero, i.e. no pivot element has been found, and the element being tested is non-zero. If the tests are both true then line 9 is the next line otherwise the statements from line 12 are executed.
Line 9 pivot is assigned the value of the variable testelem.
Line 10 row is assigned the integer corresponding to the row number of the current test element.
Line 11 terminates the conditional statement.
Line 12 tests for the current test element being non-zero and smaller than the current pivot value.
Line 13 pivot is assigned the value of the variable testelem.
Line 14 row is assigned the integer corresponding to the row number of the current test element.
Line 15 terminates the conditional statement.

Line 16 terminates the repetition statement and increments the iteration index by one, thus using the next element in the column as the test element.

Lines 17-20 if a pivot element has been found that does not lie in the pivotal position an elementary operation is used to effect a row permutation. The operation is stored in the matrix `V`.

Line 21 returns the list of the matrices `M` and `V` as the value of the procedure.

Line 22 terminates the procedure.

6.4.4 The `lowerzerosS` Procedure

The Function:

Syntax: `lowerzerosS(`a`,`x`,`j`,`k`,`r`);`

Parameters:
- a - a list of two matrices $[A, V]$,
- x - an indeterminate of the matrix A,
- j - an integer denoting a column of the matrix A,
- k - an integer denoting a row of the matrix A,
- r - an integer denoting the number of rows in A.

The procedure performs elementary operations on the matrix A to reduce the degree, in x, of all the elements below the kth element in column j to below the degree, in x, of the element in position (k,j). The elementary operations used to reduce the matrix A are also applied to the matrix V, the elementary operations matrix.

Procedure Code:

```
1 lowerzerosS:=proc(mx,var,j,k,rdim)
2       local i,m,q,M,V;
3         M:=copy(mx[1]);
4         V:=copy(mx[2]);
5         for i from k+1 to rdim do
6           if deg(M[k,j],var)<=deg(M[i,j],var) then
7             sprem(M[i,j],M[k,j],var,'m','q');
8             M:=addrowm(M,k,i,m,q);
9             V:=addrowm(V,k,i,m,q)
10            fi
11        od;
12        [op(M),op(V)]
13      end;
```

Formal Parameters:

`mx` - a list containing two matrix elements;

`var` - an unevaluated name;

`j` - an integer;

`k` - an integer;

`rdim` - an integer.

Local Variables:

`i` - an index of iteration;

`m` - the multiplier m in the pseudo-division algorithm;

`q` - the pseudo-quotient in the pseudo-division algorithm;

`M` - the first matrix element of the list a;

`V` - the second matrix element of the list a.

Line 1 defines `lowerzerosS` as the name of the procedure.

Line 2 declares the local variables.

Line 3 assigns to `M` the first element of the list `mx`.

Line 4 assigns to `V` the second element of the list `mx`.

Line 5 defines the repetition statement so that each element below the quasi-principal diagonal is accessed.

Line 6 compares the degrees, in `var`, of the test element (in position `(i,j)`) and the pivot element (in position `(k,j)`). If the latter is less than the former the degree of the former is reduced to below that of the pivot (using the following three lines).

Line 7 performs the pseudo-division algorithm between the pivot element and the test element. The multiplier m is assigned to `m` and the pseudo-quotient q is assigned to `q`.

Line 8 reassigns row `i` of `M` as `m` times row `k` minus `q` times row `i`.

Line 9 reassigns row `i` of `V` as `m` times row `k` minus `q` times row `i`. Thus storing the elementary operation in the matrix `V`.

Line 10 terminates the conditional statement.

Line 11 terminates the repetition statement and increments the index `i` by one. Lines 6-10 are executed for each valid `i` until `i` exceeds `rdim`.

Line 12 the value of the procedure is the two element list consisting of the matrix `M` and the matrix `V`.

Line 13 terminates the procedure.

6.4.5 The colredS Procedure

The Function:

Syntax: colredS(a,x);

Parameters: a - a list of two matrices,

x - an indeterminate of the matrices in a.

The first matrix in the list a is a pseudo-upper triangular matrix and the second matrix is a store for the ensuing elementary operations. The result of the procedure is a quasi-upper triangular matrix with the elements on the quasi-principal diagonal having greatest degree, in x, in their column.

Procedure Code:

```
1 colredS:=proc(mx,var)
2       local i,j,m,q,rk,dega,degb,rdim,cdim,shift,M,V;
3         M:=copy(mx[1]);
4         V:=copy(mx[2]);
5         rdim:=rowdim(M);
6         cdim:=coldim(M);
7         rk:=rank(M);
8         shift:=0;
9         for j from 1 to max(rdim,cdim) do
10          if rk-shift<j then break fi;
11          if M[j+shift,j]=0 then shift:=shift-1 fi;
12          for i from j-1+shift by -1 to 1 do
13            dega:=deg(M[i,j],var);
14            degb:=deg(M[j+shift,j],var);
15            if dega>=0 and degb>=0 and degb<=dega then
16              sprem(M[i,j],M[j+shift,j],var,'m','q');
17              M:=addrowm(M,j+shift,i,m,q);
18              V:=addrowm(V,j+shift,i,m,q)
19            fi
20          od
21        od;
22        RETURN(op(M),op(V))
23      end;
```

Formal Parameters:

mx - a two element list of two dimensional arrays;

`var` - an unevaluated name.

Local Variables:

`i` - an index of iteration;

`j` - an index of iteration;

`m` - the multiplier m in the pseudo-division algorithm;

`q` - the pseudo-quotient in the pseudo-division algorithm;

`rk` - the rank of the matrix `A`;

`dega` - an integer denoting the degree of a polynomial in the indeterminate `var`;

`degb` - an integer denoting the degree of a polynomial in the indeterminate `var`.

`cdim` - an integer denoting the number of columns in `M`;

`rdim` - an integer denoting the number of rows in `M`;

`shift` - an integer denoting the number of zeros on the pseudo-principal diagonal;

`M` - the first matrix element of the list a;

`V` - the second matrix element of the list a.

Line 1 defines `colredS` as the name of the procedure.

Line 2 declares the local variables.

Line 3 assigns `M` as a copy of the first element of the list `mx`.

Line 4 assigns `V` as a copy of the second element of the list `mx`.

Line 5 assigns `rdim` the number of rows of `M`.

Line 6 assigns `cdim` the number of columns of `M`.

Line 7 assigns `rk` as the rank of the matrix `A`.

Line 8 assigns the counter `shift` to zero.

Line 9 defines the repetition statement so that each column of `M` is accessed.

Line 10 a break is executed if the number of rows plus the number of zeros on the pseudo-principal diagonal exceeds the column index then it is not necessary to proceed further and a `break` is executed to exit from the iterative statement.

Line 11 determines if the pseudo-diagonal element is zero. If this is true the counter `shift` is decreased by one (decrementation is used because the counter `shift` is a negative number).

Line 12 defines the repetition statement that controls the row number of the element to be degree reduced. The degree reduction is carried out by starting with the element just above the quasi-principal diagonal and then proceeds with subsequently higher ones until the first row is reached.

Line 13 `dega` is assigned the value of the degree, in the indeterminate `var`, (as defined by `deg` and not the standard library function `degree`) of the element to be degree reduced.

Line 14 degb is assigned the value of the degree, in the indeterminate var, (as defined by deg and not the standard library function degree) of the element on the quasi-principal diagonal.

Line 15 If the two elements defined in lines 13-14 are both non-zero and the degree of the off quasi-principal diagonal element is greater than or equal to the degree of the quasi-principal diagonal element the pseudo-division algorithm is performed. Then an elementary operation is performed to reduce the degree of the off quasi-principal diagonal element.

Line 16 computes the pseudo-quotient, q, and the polynomial multiplier, m, as defined by the pseudo-division algorithm.

Line 17 performs the elementary operation of redefining row i as m row i minus q row j+shift. This reduces the degree of the off quasi-principal diagonal element (and also alters other elements in row i).

Line 18 performs the same elementary operation on the matrix V: thus storing the elementary operations as a matrix.

Line 19 terminates the pseudo-division algorithm for valid polynomials as defined in line 14.

Line 20 terminates the inner repetition statement and causes the next iteration to begin, i.e. the row number is reduced by one, and the process defined by lines 15-19 is repeated if the polynomials are of the form defined by line 15.

Line 21 terminates the outer repetition statement and causes the procedure to move to the next column, so that the degree reduction may be performed on that column. In this way all of the columns are degree reduced.

Line 22 the elements of the two matrices are returned as a list.

Line 23 terminates the procedure.

6.4.6 The addrowm Procedure

The Function:

Syntax: addrowm(A,r_1,r_2,m,q);

Parameters: A - a matrix in two indeterminates,
r_1 - a row number of the matrix A,
r_2 - a row number of the matrix A,
m - a polynomial,
q - a polynomial

Performs the elementary operation of subtracting q times row r_1 from m times r_2 and puts the result in row r_2 of A.

Procedure Code:

```
1 addrowm:=proc(A,r1,r2,m,q)
2       local k,M;
3         M:=copy(A);
4         for k to coldim(M) do
5           M[r2,k]:=expand(m*M[r2,k]-q*M[r1,k])
6         od;
7         op(M)
8       end;
```

Formal Parameters:

`A` - a two dimensional array in two unevaluated names;

`r1` - an integer;

`r2` - an integer;

`m` - a polynomial in the indeterminates of `A`;

`q` - a polynomial in the indeterminates of `A`.

Local Variables:

`k` - an index of iteration.

`M` - the result of the procedure.

Line 1 defines `addrowm` as the name of the procedure.

Line 2 declares the local variables.

Line 3 assigns `M` as a copy of `A`, so that when `M` is altered `A` is not also altered.

Line 4 defines a repetition statement to access every element in a row.

Line 5 assigns the `k`th element of row `r2` as the polynomial obtained when `q` times the `k`th element of row `r1` is subtracted from `m` times the `k`th element of row `r2`. The function `expand` is used so that like terms are combined and expressions that evaluate to zero assigned the value zero

Line 6 terminates the repetition statement and increments `k` by 1. The loop is repeated until `k=coldim(M)`.

Line 7 returns the elements of `M` as the value of the procedure.

Line 8 terminates the procedure.

6.4.7 The deg Procedure

The Function:

Syntax: `deg(`a`,`x`);`

Parameters: a - a polynomial,
x - an indeterminate.

The result of the procedure is the degree of the polynomial a in the indeterminate x. If a is the zero polynomial the value is -1; if a does not contain the indeterminate x (and is non-zero) the value is zero.

Note that the degree of the zero polynomial is usually given the value minus infinity ($-\infty$) but here this function is only used for pure polynomials and not rational polynomials so that any negative number could be used: all that is required is that for the degree of the zero polynomial to be less than the degree of any other expression.

Procedure Code:

```
1 deg:=proc(a,var)
2       local d;
3          if a=0 then d:=-1; RETURN(d) fi;
4          if degree(a,var)=FAIL then d:=0 else d:=degree(a,var) fi;
5          d
6       end;
```

Formal Parameters:

`a` - an expression;

`var` - an indeterminate or set or list of indeterminates.

Local Variables:

`d` - an integer; the result of the procedure.

Line 1 defines `deg` as the name of the procedure.

Line 2 declares the local variables.

Line 3 decides if the expression `a` is zero; if so then -1 is returned as the value of the procedure via the explicit return function.

Line 4 decides if the expression `a` contains the indeterminate(s) `var`. If this is true `d` is assigned the value of the degree of the expression `a` in the indeterminate(s) `var`. If not the value zero is assigned to `d`. Note that the zero expression also returns FAIL from the function `degree` but if the expression is zero this line is never executed, thus there is no danger of the zero expression being given the degree value zero.

Line 5 the value of `d` is returned as the value of the procedure.

Line 6 terminates the procedure.

6.4.8 The emxc Procedure

The Function:

Syntax: emxc(*A*);

Parameters: *A* - a matrix.

The result of the function is the matrix *A* in which all elements are expressed in the form $a + Ib$, where a and b may be zero, numeric or polynomial. Thus the matrix *A* is simplified over the complex field.

Procedure Code:

```
1 emxc:=proc(A)
2       local i,j,rdim,cdim,B;
3         cdim:=coldim(A);
4         rdim:=rowdim(A);
5         B:=array(1..rdim,1..cdim);
6         for i to rdim do
7           for j to cdim do
8             B[i,j]:=evalc(A[i,j])
9           od
10          od;
11        op(B)
12      end;
```

Formal Parameters:

A - a two dimensional array.

Local Variables:

i - an index of iteration;

j - an index of iteration;

cdim - an integer denoting the number of columns in M;

rdim - an integer denoting the number of rows in M;

B - a two dimensional array that contains the result of the simplified form of the matrix *A*.

Line 1 defines the name of the procedure as emxc.

Line 2 declares the local variables.

Line 3 assigns rdim the number of rows of A.

Line 4 assigns cdim the number of columns of A.

Line 5 declares B as a two dimensional array with rdim rows and cdim columns. This declaration is necessary otherwise B would have the data type table and a

two dimensional table is not recognised as a matrix, unlike a two dimensional array. An alternative to declaring B as an array would be to let MAPLE treat it as a table and then convert it to an array using `convert(B,array)`.

Line 6 defines the repetition statement so that all rows of the matrix A are accessed.

Line 7 defines the repetition statement so that all columns of the matrix A are accessed.

Line 8 evaluates element (i,j) over the complex field, i.e. expresses each element in the form $a + Ib$.

Line 9 terminates the column repetition statement.

Line 10 terminates the row repetition statement.

Line 11 returns the elements of B as the result of the procedure.

Line 12 terminates the procedure.

6.4.9 The unitmx Procedure

The Function:

Syntax: `unitmx(`n`);`

Parameter: n - an integer.

The result of the function is a matrix with ones along the principal diagonal and zeros in all other positions, i.e the unit matrix with n rows and columns.

Procedure Code:

```
 1 unitmx:=proc(n)
 2       local i,j,E;
 3         E:=array(1..n,1..n);
 4         for i to n do
 5           for j to n do
 6              if i=j then E[i,j]:=1 else E[i,j]:=0 fi
 7           od
 8         od;
 9         op(E)
10       end;
```

Formal Parameters:

n - an integer.

Local Variables:

i - an index of iteration;

j - an index of iteration;

E - a two dimensional array.

Line 1 defines `unitmx` as the name of the procedure.

Line 2 declares the local variables.

Line 3 declares `E` as a two dimensional array of dimension `n`×`n`, i.e. a matrix with `n` rows and `n` columns.

Line 4 defines the repetition statement to access each row in turn from `1` to `n`.

Line 5 defines the repetition statement to access each column in turn form `1` to `n`.

Line 6 assigns `1` to those elements on the principal diagonal, i.e. row number equal to the column number, and zero is assigned to all other elements.

Line 7 terminates the repetition statement on line 5. Increments the column index `j` by one and returns the procedure to line 6.

Line 8 terminates the repetition statement on line 4. Increments the row `i` index by one and returns the procedure to line 5.

Line 9 returns the elements of the array `E` as the value of the procedure.

Line 10 terminates the procedure.

6.4.10 The upperS Procedure

The Function:

Syntax: `upperS(`A`,`j`,`k`);`

Parameters: A - a matrix,
j - a column number of the matrix A,
k - a row number of the matrix A.

The result of the function is `true` if all the elements in column j below row k are zero.

Procedure Code:

```
1 upperS:=proc(A,j,k)
2       local i;
3         for i from k+1 to rowdim(A) do
4           if A[i,j]<>0 then RETURN(false) fi
5         od;
6         RETURN(true)
7       end;
```

Formal Parameters:

`A` - a two dimensional array;

`j` - an integer;

`k` - an integer.

Local Variables:

`i` - an index of iteration.

Line 1 defines `upperS` as the name of the procedure.

Line 2 declares the local variables.

Line 3 defines the repetition statement that accesses each of the elements in position `(k+1,j)` to `(m,j)`, where `m` is the number of rows in `A`.

Line 4 tests each element in turn. If it is non-zero `false` is returned immediately.

Line 5 terminates the repetition statement. The row index is incremented by one so that the next element in the list can tested.

Line 6 the value of the procedure is `true` if all of the elements tested are zero. The explicit return is not accessed if all the elements are zero in the repetition statement.

Line 7 terminates the procedure.

Chapter 7

Evaluation and Concluding Discussion

7.1 Introduction

The purpose of this final chapter is to evaluate the MAPLE procedures detailed in the previous chapter. This is achieved by testing the program for a range of examples. The examples have been constructed in such a way as to test each procedure as fully as possible, by knowing the factorisation of each matrix at the outset. Thus the examples may seem slightly artificial in the context of control systems. The starting point for each example is a coprime matrix to which polynomial matrix factors are multiplied, each possessing a different property; thus the factorisation is known at the outset and therefore the types of factors that should be the result of each procedure

step. For example, suppose that a coprime matrix is multiplied by a polynomial matrix factor with its determinant polynomial in one indeterminate; therefore by favouring this indeterminate the factor is calculated in Step 1 of the algorithm (Line 4 of the procedure code). In addition to this final evaluation of the program, each procedure was individually tested at the time of construction.

The motivation for the program was the impracticability of performing manually the GCD Algorithm due to the types of factor possessed by matrices in two indeterminates. A further motivation is demonstrated by considering the amount of time taken for each of the calculations. Seemingly simple examples take many seconds of CPU time to be calculated. This is further investigated in Section 7.3.

As a result of the testing of the program it was found that certain examples caused an error to occur. The reasons for these failures are investigated and certain remedies are proposed.

7.2 Test Examples

The examples documented are considered in four main sets. The first set considers matrices with low dimensions, i.e. 3×2, to demonstrate the range of different factors that a matrix can possess. The second set considers three square polynomial matrices, one each of the types described in Section 4.5.4. These are multiplied together in all six combinations and then pre-multiplied by a coprime matrix to form six non-square polynomial matrices with non-trivial greatest common divisor. The GCD program is then used on each of the matrices with respect to the two indeterminates in turn, and the results compared. The third set of examples more fully demonstrates the flow of the program by considering two matrices and displaying arguments for each program statement and the results returned. Finally the fourth set demonstrates the program for larger dimensioned matrices. However, these are limited by the power and storage capacity of the host computer.

Example 7.1: Consider a matrix with one factor constructed by

$$B_1 = \begin{pmatrix} 1 & y \\ y & x \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x & 0 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} x+y & y \\ xy+x & x \\ x+1 & 1 \end{pmatrix}$$

The high-order minors are $x(x-y^2)$, $x(1-y)$, $x(y-x)$. Thus B_1 is primitive over $\mathbb{R}[y][x]$ but not primitive over $\mathbb{R}[x][y]$.

To calculate the greatest common right divisor of B_1 either of the indeterminates may be specified in the function call. Firstly consider performing the calculation by favouring the x indeterminate, i.e. the command `gcrd(`B_1`,`x`,'`U_x`');`

Thus `var` :=x and `var1` :=y. The first call to `rprimfac` on line 4 performs the primitive factorisation $B_1 \equiv \mathtt{A} = \mathtt{M} \times \mathtt{R0}$ over $\mathbb{R}[\mathtt{var}][\mathtt{var1}]$, i.e. over $\mathbb{R}[x][y]$, and since B_1 is not primitive over this ring `R0` is returned as a non-unit matrix containing the primitive factor of B_1.

$$\mathtt{R0} = \begin{pmatrix} 1 & 1 \\ 0 & x \end{pmatrix} \qquad \mathtt{M} = \begin{pmatrix} x+y & -1 \\ xy+x & -y \\ x+1 & -1 \end{pmatrix}$$

The second call to `rprimfac` on line 5 performs the primitive factorisation of `M` over $\mathbb{R}$[var1][var], i.e. over $\mathbb{R}[y][x]$, but `M` (calculated on line 4) is primitive over this ring, thus `R1` is returned as the unit matrix.

$$\mathtt{R1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \mathtt{M} = \begin{pmatrix} x+y & -1 \\ xy+x & -y \\ x+1 & -1 \end{pmatrix}$$

Next the `shiftherm` procedure is performed on the matrix `M` (the result from line 5) to give

$$\mathtt{H} = \begin{pmatrix} -y^3+y & 0 \\ 0 & -1-y \\ 0 & 0 \end{pmatrix}$$

and therefore the test for coprimeness performed in lines 7–10 terminates the procedure returning `R1` × `R0` as the result. Thus D_x =`gcrd(`B_1`,`x`,'`U_x`');` yields

$$D_x = \begin{pmatrix} 1 & 1 \\ 0 & x \end{pmatrix}$$

$$U_x = \begin{pmatrix} x+y & -1 \\ xy+x & -y \\ x+1 & -1 \end{pmatrix}$$

By design, this example is primitive over $\mathbb{R}[y][x]$ but is not primitive over $\mathbb{R}[x][y]$; thus testing the two possible outcomes of the primitive factorisation procedure, namely a non-unimodular polynomial matrix when the matrix argument is not primitive (`rprimfac(`B_1`,`x`,'R0');` on line 4) or the unit matrix when the matrix argument is primitive (`rprimfac(`B_1`,`y`,'R1');` on line 5). After the two primitive factorisations have been performed the resulting matrix `M` is primitive with respect to both indeterminates. In addition to this the matrix `M`, computed on line 5, is coprime; thus the expressions on lines 7–10 terminate the procedure and deliver the greatest common right divisor as the result.

Consider now favouring the y indeterminate. It is clear from the form of the GCRD Algorithm that the result will be identical to the result obtained by favouring x. In effect by favouring y Steps 1 and 2 are interchanged and since one of these steps returns the unit matrix the product of `R1` and `R0` is commutative. Hence

D_y =gcrd(B_1,y,'U_y'); yields

$$D_y = \begin{pmatrix} 1 & 1 \\ 0 & x \end{pmatrix}$$

$$U_y = \begin{pmatrix} x+y & -1 \\ xy+x & -y \\ x+1 & -1 \end{pmatrix}$$

□

Example 7.2: Consider the matrix B_2 with primitive factors over both $\mathbb{R}[x][y]$ and $\mathbb{R}[y][x]$, defined by

$$B_2 = \begin{pmatrix} 1 & y \\ y & x \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} y & 0 \\ 1 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} xy+y+1 & y+1 \\ xy^2+y+x & x+y \\ xy+2 & 2 \end{pmatrix}$$

For this matrix the primitive factorisation procedure delivers non-unit matrices to both invocations:

M:=rprimfac(B_2,x,'R0'); yields

$$\mathtt{R0} = \begin{pmatrix} 1 & 1 \\ 0 & x \end{pmatrix} \qquad \mathtt{M} = \begin{pmatrix} xy+y+1 & -y \\ xy^2+x+y & -y^2 \\ xy+2 & -y \end{pmatrix}$$

M:=rprimfac(M,y,'R1'); yields

$$\mathtt{R1} = \begin{pmatrix} 1 & 0 \\ 0 & y \end{pmatrix} \qquad \mathtt{M} = \begin{pmatrix} xy+y+1 & -1 \\ xy^2+x+y & -y \\ xy+2 & -1 \end{pmatrix}$$

The third step of the algorithm computes the modified Hermite form. This is achieved by the procedure **shiftherm**:

H:=shiftherm(M,x); yields

$$\mathtt{H} = \begin{pmatrix} -y^5+y^4-y^3+y & 0 \\ 0 & y^2-y \\ 0 & 0 \end{pmatrix}$$

The number of indeterminates in this expression is one, therefore the coprimeness test on lines 7-10 terminates the procedure and the greatest common right divisor D is delivered by

`D:=evalm(R1 & * R0);` yielding

$$\mathtt{D} = \begin{pmatrix} 1 & 1 \\ 0 & xy \end{pmatrix}$$

and the coprime matrix resulting from removing this greatest common divisor is given by

`op(`U_x`);` yielding

$$U_x = \begin{pmatrix} xy+y+1 & -1 \\ xy^2+x+y & -y \\ xy+2 & -1 \end{pmatrix}$$

Favouring the y indeterminate interchanges Steps 1 and 2 of the GCRD Algorithm and the result is exactly the same as the result obtained by favouring x, i.e.

D_y `=gcrd(`B_2`,`y`,'`U_y`');` yields

$$D_y = \begin{pmatrix} 1 & 1 \\ 0 & xy \end{pmatrix}$$

$$U_y = \begin{pmatrix} xy+y+1 & -1 \\ xy^2+x+y & -y \\ xy+2 & -1 \end{pmatrix}$$

However, the product of the matrices R1 and R0 is non-commutative and by examining the individual steps of the procedure it is seen that the result is obtained in an entirely different manner:

`M:=rprimfac(`B_2`,`y`,'R0');` yields

$$\mathtt{R0} = \begin{pmatrix} 1 & 1 \\ 0 & y \end{pmatrix} \qquad \mathtt{M} = \begin{pmatrix} xy+y+1 & -x \\ xy^2+x+y & -xy \\ xy+2 & -x \end{pmatrix}$$

`M:=rprimfac(M,`y`,'R1');` yields

$$\mathtt{R1} = \begin{pmatrix} 1 & 0 \\ 0 & x \end{pmatrix} \qquad \mathtt{M} = \begin{pmatrix} xy+y+1 & -1 \\ xy^2+x+y & -y \\ xy+2 & -1 \end{pmatrix}$$

`H:=shiftherm(M,y);` yields

$$\mathtt{H} = \begin{pmatrix} -1 - x + 17x^2 + 32x^3 - 101x^4 & \\ -326x^5 + 95x^6 + 1430x^7 + 1465x^8 & \\ -2150x^9 - 6067x^{10} - 3142x^{11} & \\ +6384x^{12} + 11979x^{13} + 6148x^{14} & 0 \\ -5144x^{15} - 10582x^{16} - 7126x^{17} & \\ -716x^{18} + 2818 + x^{19} + 2819x^{20} & \\ +1511x^{21} + 523x^2 2 + 118x^{23} & \\ +16x^{24} + x^{25} & \\ & -1 - x + 7x^2 + 13x^3 \\ 0 & -7x^4 - 37x^5 - 32x^6 \\ & +4x^7 + 26x^8 + 20x^9 + 7x^{10} \\ 0 & 0 \end{pmatrix}$$

This example also demonstrates the assertion made in Section 6.2 that the degree in the non-favoured indeterminate rapidly increases when performing the 2-D Hermite form. Thus demonstrating the necessity of the `rowfactors` procedure, although the procedure is not invoked here because the coprimeness test terminates the procedure.

□

Example 7.3: Consider the matrix B_3 that is primitive with respect to both indeterminates formed by

$$\begin{aligned} B_3 &= \begin{pmatrix} 1 & y \\ y & x \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x+y & 0 \\ 1 & 1 \end{pmatrix} \\ &= \begin{pmatrix} x+2y & y \\ xy+y^2+x & x \\ x+y+1 & 1 \end{pmatrix} \end{aligned}$$

Hence the GCRD Algorithm delivers unit matrices from the primitive factorisations in Steps 1 and 2 (lines 4 and 5 of the `gcrd` code) but the test for coprimeness will fail so that lines 11-19 are executed. Explicitly the intermediate results of `gcrd(`B_3`,`x`,'`U_x`');` are:

`M:=rprimfac(`B_3`,`x`,'R0');` yields

$$\mathtt{R0} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \mathtt{M} = \begin{pmatrix} x+2y & y \\ xy+y^2+x & x \\ x+y+1 & 1 \end{pmatrix}$$

`M:=rprimfac(M,`y`,'R1');` yields

$$\mathtt{R1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \mathtt{M} = \begin{pmatrix} x+2y & y \\ xy+y^2+x & x \\ x+y+1 & 1 \end{pmatrix}$$

`H:=shiftherm(M,`x`);` yields

$$\mathtt{H} = \begin{pmatrix} 2y-y^2-y^3 & 2y-y^2-y^3 \\ 0 & xy-y^2-y-x \\ 0 & 0 \end{pmatrix}$$

`H:=rowfactors(H,`y`);` yields

$$\mathtt{H} = \begin{pmatrix} -1 & -1 \\ 0 & x+y \end{pmatrix}$$

Therefore the greatest common right divisor is

$$\begin{pmatrix} -1 & -1 \\ 0 & x+y \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} -1 & -1 \\ 0 & x+y \end{pmatrix}$$

and

$$U_x = \begin{pmatrix} -x-2y & -1 \\ -xy-y^2-x & -y \\ -x-y-1 & -1 \end{pmatrix}$$

To verify the result compute the greatest common right divisor with respect to y, i.e. `gcrd(`B_3`,`y`,'`U_x`');`. The intermediate results are:

`M:=rprimfac(`B_3`,`y`,'R0');` yields

$$\mathtt{R0} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \mathtt{M} = \begin{pmatrix} x+2y & y \\ xy+y^2+x & x \\ x+y+1 & 1 \end{pmatrix}$$

`M:=rprimfac(M,`x`,'R1');` yields

$$\mathtt{R1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \mathtt{M} = \begin{pmatrix} x+2y & y \\ xy+y^2+x & x \\ x+y+1 & 1 \end{pmatrix}$$

`H:=shiftherm(M,`y`);` yields

$$\mathtt{H} = \begin{pmatrix} 262144x^5 - 851968x^6 + 1081344x^7 & 262144x^5 - 851968x^6 + 1081344x^7 \\ -696320x^8 + 250880x^9 & -696320x^8 + 250880x^9 \\ -51456x^{10} + 5632x^{11} - 256x^{12} & -51456x^{10} + 5632x^{11} - 256x^{12} \\ 0 & -256x^3 + 384x^4 - 144x^5 + 16x^6 \\ & -256x^2y + 384x^3y - 144x^4y + 16x^5y \\ 0 & 0 \end{pmatrix}$$

`H:=rowfactors(H,x);` yields

$$\begin{pmatrix} -1 & -1 \\ 0 & x+y \\ 0 & 0 \end{pmatrix}$$

Therefore the greatest common right divisor over the ring $\mathbb{R}[x][y]$ is given by

$$\begin{pmatrix} -1 & -1 \\ 0 & x+y \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} -1 & -1 \\ 0 & x+y \end{pmatrix}$$

and the coprime matrix is given by

$$U_y = \begin{pmatrix} -x-2y & -1 \\ -xy-y^2-x & -y \\ -x-y-1 & -1 \end{pmatrix}$$

As expected the results of the right primitive factorisation procedures on lines 4 and 5 are equivalent, because the matrix B_3 is primitive over both $\mathbb{R}[x][y]$ and $\mathbb{R}[y][x]$. The results of the `shiftherm` procedure demonstrates the inequality of the 2-D Hermite form and also the necessity of removing factors common to each element of a particular row before computing the left primitive factorisation on line 16. □

Example 7.4: Consider the matrix B_4 with complex and real primitive roots.

$$\begin{aligned} B_4 &= \begin{pmatrix} 1 & y \\ y & x \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x^2+x+1 & 0 \\ 1 & y+1 \end{pmatrix} \\ &= \begin{pmatrix} x^2+x+1+y & y(y+1) \\ yx^2+xy+x+y & x(y+1) \\ x^2+x+1 & y+1 \end{pmatrix} \end{aligned}$$

Thus the primitive roots over $\mathbb{R}[x][y]$ are

$$x = -1 + i\sqrt{3}, -1 - i\sqrt{3}$$

and over $\mathbb{R}[y][x]$ are

$$y = -1$$

Consider firstly favouring x the results of the intermediate steps are:

`rprimfac(`B_4`,`x`,'R0');` yields

$$\mathtt{R0} = \begin{pmatrix} 1 & y^2+1 \\ 0 & x^2+x+1 \end{pmatrix} \qquad \mathtt{M} = \begin{pmatrix} x^2+x+1+y & -1-y^2 \\ x^2y+xy+y+x & -y(y^2+1) \\ x^2+x+2 & -1-y^2 \end{pmatrix}$$

`rprimfac(M,`y`,'R1');` yields

$$\mathtt{R1} = \begin{pmatrix} -y-1 & y^2 \\ x+1 & -(-1+y)(x+1) \end{pmatrix} \qquad \mathtt{M} = \begin{pmatrix} x^2+x+1+y & -1 \\ x^2y+xy+x+y & -y \\ x^2+x+2 & -1 \end{pmatrix}$$

Therefore result of the command `gcrd(`B_4`,`x`,`U_x`);` is

$$\begin{pmatrix} 1 & y+1 \\ 0 & (y+1)(x^2+x+1) \end{pmatrix}$$

$$U_x = \begin{pmatrix} x^2+x+1+y & -1 \\ x^2y+xy+x+y & -y \\ x^2+x+2 & -1 \end{pmatrix}$$

Secondly favouring y gives:

`rprimfac(`B_4`,`y`,'R0');` yields

$$\mathtt{R0} = \begin{pmatrix} 1 & 0 \\ 0 & y+1 \end{pmatrix} \qquad \mathtt{M} = \begin{pmatrix} x^2+x+1+y & y \\ x^2y+xy+y+x & x \\ x^2+x+2 & 1 \end{pmatrix}$$

`rprimfac(M,`x`,'R1');` yields

$$\mathtt{R1} = \begin{pmatrix} 1 & 0 \\ 0 & x^2+x+1 \end{pmatrix} \qquad \mathtt{M} = \begin{pmatrix} x^2+x+1+y & -1 \\ x^2y+xy+y+x & -y \\ x^2+x+2 & -1 \end{pmatrix}$$

`shiftherm(M,`y`);` yields

$$\mathtt{H} = \begin{pmatrix} -x^7-2x^6-6x^5-4x^4-5x^3+6x^2+4x+8 & 0 \\ 0 & -x^5-x^4-3x^3+x^2+4 \\ 0 & 0 \end{pmatrix}$$

Therefore the result to `gcrd(`B_4`,`y`,'`U_y`')`; is

$$\begin{pmatrix} 1 & y+1 \\ 0 & (x^2+x+1)(1+y) \end{pmatrix}$$

$$U_y = \begin{pmatrix} x^2+x+1+y & -1 \\ x^2y+xy+y+x & -y \\ x^2+x+2 & -1 \end{pmatrix}$$

Thus the routine can deal with matrices possessing complex roots. □

The following example has been presented in the introduction to Chapter 4 to demonstrate the motivation for the GCD program. It is reconstructed here to show that the secondary matrices do not cause the program to fail.

Example 7.5: Consider the matrix, B_5, formed by

$$B_5 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & x+1 \end{pmatrix} \begin{pmatrix} 1-xy & -y^2 \\ x^2 & 1+xy \end{pmatrix}$$

$$= \begin{pmatrix} 1-xy & -y^2 \\ x^2 & 1+xy \\ 1-xy & -y^2 \end{pmatrix}$$

The primitive root of B_5 is given by $x = -1$ therefore line 4 of the `gcrd` procedure (with x favoured) effectively delivers the greatest common right divisor, as the matrix B_5 does not possess primitive roots in the y indeterminate and the `shiftherm` procedure delivers a matrix independent of x. Thus `gcrd(`B_5`,`x`,`U_x`)`; yields

$$\begin{pmatrix} -y-1 & y^2 \\ x+1 & -(y-1)(x+1) \end{pmatrix}$$

$$U_x = \begin{pmatrix} y-xy^2-1+xy^2-y & -y^3 \\ -x^3-x^2+x^3y+2x^2y+x+1+xy & x^2y^2+xy^2+xy+y+1 \\ -1+xy+y-xy^2-y^2 & -y^3 \end{pmatrix}$$

Also `gcrd(`B_5`,`y`,`U_y`)`; yields

$$\begin{pmatrix} -y-1 & y^2 \\ x+1 & -(y-1)(x+1) \end{pmatrix}$$

$$U_y = \begin{pmatrix} y-xy^2-1+xy^2-y & -y^3 \\ -x^3-x^2+x^3y+2x^2y+x+1+xy & x^2y^2+xy^2+xy+y+1 \\ -1+xy+y-xy^2-y^2 & -y^3 \end{pmatrix}$$

Example 7.6: As a further example involving this secondary matrix consider the matrix B_6 that is primitive over both $\mathbb{R}[x][y]$ and $\mathbb{R}[y][x]$

$$B_6 = \begin{pmatrix} 1 & y \\ y & x \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x+y & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1-xy & -y^2 \\ x^2 & 1+xy \end{pmatrix}$$

$$= \begin{pmatrix} x+2y-2xy^2 & -2y^3+y \\ xy-x^2y^2+y^2-y^3x+x-yx^2+x^3 & -y^3x-y^4-xy^2+x^2+yx \\ x-x^2y+y-xy^2+1-xy+x^2 & -xy^2-y^3-y^2+1+xy \end{pmatrix}$$

Here the results obtained by favouring first x and secondly y are not the same. However, since they are both greatest common right divisors they are linked by a unimodular matrix V.

`gcrd(`B_6`,`x`,'`U_x`');` yields

$$\begin{pmatrix} 2y^2+1 & 1-2y^2 \\ -8y-8x & 8y+8x \end{pmatrix}$$

$$U_x = \begin{pmatrix} \frac{x}{2}+\frac{3y}{2}-xy^2-y^3 & \frac{-(2y^2-1)^2}{16} \\ \frac{xy}{2}-\frac{x^2y^2}{2}+\frac{y^2}{2}-y^3x & \frac{x^2y^2}{8}-\frac{x^2}{16}+\frac{xy^2}{16}-\frac{xy^4}{8} \\ +x+\frac{x^3}{2}-\frac{y^4}{2}-\frac{xy^2}{2} & +\frac{3xy}{16}-\frac{y^3x}{8}-\frac{y^5}{8}-\frac{y}{16}+\frac{y^3}{16} \\ \frac{x}{2}-\frac{yx^2}{2}+\frac{y}{2}-xy^2 & \frac{xy}{16}-\frac{x}{16}+\frac{xy^2}{8}-\frac{y^3x}{8}-\frac{y^4}{8} \\ +1+\frac{x^2}{2}-\frac{y^3}{2}-\frac{y^2}{2} & +\frac{y^2}{16}-\frac{1}{16}+\frac{3y}{16}-\frac{y^3}{8} \end{pmatrix}$$

`gcrd(`B_6`,`y`,'`U_y`');` yields

$$\begin{pmatrix} -2x^2-1 & 2x^2-1 \\ -8y-8x & 8y+8x \end{pmatrix}$$

$$U_y = \begin{pmatrix} \frac{-x}{2}-\frac{3y}{2}+xy^2+y^3 & \frac{x^2}{8}-\frac{x^2y^2}{4}+\frac{xy}{4}-\frac{y^2}{8}-\frac{1}{16} \\ \frac{x^2y^2}{2}-\frac{xy}{2}-\frac{y^2}{2}+y^3x-x & \frac{x^4}{8}-\frac{x^3y^2}{8}-\frac{x^3y}{8}+\frac{yx^2}{8} \\ -\frac{x^3}{2}+\frac{y^4}{2}+\frac{xy^2}{2} & -\frac{x^2y^3}{8}+\frac{3x^2}{16}-\frac{xy}{16}+\frac{xy^2}{16}-\frac{y}{16}-\frac{y^3}{16} \\ \frac{yx^2}{2}-\frac{x}{2}-\frac{y}{2}+xy^2-1 & \frac{x^3}{8}-\frac{x^3y}{8}+\frac{x^2}{8}-\frac{x^2y^2}{8} \\ -2x^2+2y^3+\frac{y^2}{2} & -\frac{yx^2}{8}+\frac{xy}{16}+\frac{3x}{16}-\frac{y}{16}-\frac{1}{16}-\frac{y^2}{16} \end{pmatrix}$$

and the connection is given by

$$\begin{pmatrix} -2x^2-1 & 2x^2-1 \\ -8y-8x & 8y+8x \end{pmatrix} = \underbrace{\begin{pmatrix} -1 & \frac{x^2-y^2}{4x+4y} \\ 0 & 1 \end{pmatrix}}_{V} \begin{pmatrix} 2y^2+1 & 1-2y^2 \\ -8y-8x & 8y+8x \end{pmatrix}$$

□

The following example demonstrates that the order in which the factors are multiplied together affects the form in which the result is given, but the results are all related by unimodular matrices.

Example 7.7: Consider the matrices formed from the following

$$A_0 = \begin{pmatrix} 1 & y \\ y & x \\ 1 & 1 \end{pmatrix}$$

$$A_1 = \begin{pmatrix} x & 0 \\ 1 & 1 \end{pmatrix}$$

$$A_2 = \begin{pmatrix} y & 0 \\ 1 & 1 \end{pmatrix}$$

$$A_3 = \begin{pmatrix} x+y & 0 \\ 1 & 1 \end{pmatrix}$$

Thus there are six ways of combining the factors

$$C_1 = A_0A_1A_2A_3 = \begin{pmatrix} x^2y + 2xy^2 + y^3 + xy + y^2 + y & y \\ x^2y^2 + y^3x + yx^2 + xy^2 + x^2 + xy + x & x \\ x^2y + xy^2 + xy + y^2 + x + y + 1 & 1 \end{pmatrix}$$

$$C_2 = A_0A_1A_3A_2 = \begin{pmatrix} x^2y + 2xy^2 + y^3 + y^2 + y & y \\ x^2y^2 + y^3x + yx^2 + xy^2 + xy + x & x \\ x^2y + xy^2 + xy + y^2 + y + 1 & 1 \end{pmatrix}$$

$$C_3 = A_0A_2A_1A_3 = \begin{pmatrix} 2x^2y + 2xy^2 + xy + y^2 + y & y \\ x^2y^2 + y^3x + x^3 + x^2 + yx^2 + xy^2 + xy + x & x \\ x^2y + xy^2 + xy + x + x^2 + y + 1 & 1 \end{pmatrix}$$

$$C_4 = A_0A_2A_3A_1 = \begin{pmatrix} 2x^2y + 2xy^2 + xy + y & y \\ x^2y^2 + y^3x + x^3 + x^2 + yx^2 + xy^2 + x & x \\ x^2y + xy^2 + xy + x + x^2 + 1 & 1 \end{pmatrix}$$

$$C_5 = A_0A_3A_2A_1 = \begin{pmatrix} x^2y + 2xy^2 + xy + y & y \\ x^2y^2 + y^3x + yx^2 + x^2 + x & x \\ x^2y + xy^2 + xy + x + 1 & 1 \end{pmatrix}$$

$$C_6 = A_0A_3A_1A_2 = \begin{pmatrix} x^2y + 2xy^2 + y^2 + y & y \\ x^2y^2 + y^3x + yx^2 + yx^2 + xy + x & x \\ x^2y + xy^2 + xy + y + 1 & 1 \end{pmatrix}$$

$D_{1(x)} = \texttt{gcrd}(C_1, x, U_{1(x)})$; yields

$$D_{1(x)} = \begin{pmatrix} y^2 + y + 1 + xy + x & 1 \\ (x+y)yx & 0 \end{pmatrix}$$

$$U_{1(x)} = \begin{pmatrix} y & 1 \\ x & y \\ 1 & 1 \end{pmatrix}$$

$D_{1(y)}$ =gcrd($C_1, y, U_{1(y)}$); yields

$$D_{1(y)} = \begin{pmatrix} -x-1-y-y^2-xy & -1 \\ -(x+y)xy & 0 \end{pmatrix}$$

$$U_{1(y)} = \begin{pmatrix} -y & -1 \\ -x & -y \\ -1 & -1 \end{pmatrix}$$

$D_{2(x)}$ =gcrd($C_2, x, U_{2(x)}$); yields

$$D_{2(x)} = \begin{pmatrix} y^2+y+1+xy & 1 \\ (x+y)yx & 0 \end{pmatrix}$$

$$U_{2(x)} = \begin{pmatrix} y & 1 \\ x & y \\ 1 & 1 \end{pmatrix}$$

$D_{2(y)}$ =gcrd($C_2, y, U_{2(y)}$); yields

$$D_{2(y)} = \begin{pmatrix} -y^2-y-1-xy & -1 \\ (x+y)xy & 0 \end{pmatrix}$$

$$U_{2(y)} = \begin{pmatrix} -y & 1 \\ -x & y \\ -1 & 1 \end{pmatrix}$$

$D_{3(x)}$ =gcrd($C_3, x, U_{3(x)}$); yields

$$D_{3(x)} = \begin{pmatrix} y+1+x^2+xy+x & 1 \\ (x+y)yx & 0 \end{pmatrix}$$

$$U_{3(x)} = \begin{pmatrix} y & 1 \\ x & y \\ 1 & 1 \end{pmatrix}$$

$D_{3(y)}$ =gcrd($C_3, y, U_{3(y)}$); yields

$$D_{3(y)} = \begin{pmatrix} -x^2-x-1-y-xy & -1 \\ -(x+y)xy & 0 \end{pmatrix}$$

$$U_{3(y)} = \begin{pmatrix} -y & -1 \\ -x & -y \\ -1 & -1 \end{pmatrix}$$

$D_{4(x)}$ =gcrd(C_4, x, $U_{4(x)}$); yields

$$D_{4(x)} = \begin{pmatrix} x^2+x+1+xy & 1 \\ -(x+y)xy & 0 \end{pmatrix}$$

$$U_{4(x)} = \begin{pmatrix} y & -1 \\ x & -y \\ 1 & -1 \end{pmatrix}$$

$D_{4(y)}$ =gcrd(C_4, y, $U_{4(y)}$); yields

$$D_{4(y)} = \begin{pmatrix} -x^2-x-1-xy & -1 \\ -(x+y)xy & 0 \end{pmatrix}$$

$$U_{4(y)} = \begin{pmatrix} -y & -1 \\ -x & -y \\ -1 & -1 \end{pmatrix}$$

$D_{5(x)}$ =gcrd(C_5, x, $U_{5(x)}$); yields

$$D_{5(x)} = \begin{pmatrix} -xy-x-1 & -1 \\ 1-x^2-y & 1-y-x \end{pmatrix}$$

$$U_{5(x)} = \begin{pmatrix} -x-2y+1 & 1 \\ -xy-x-y^2+y & y \\ -x-y & 1 \end{pmatrix}$$

$D_{5(y)}$ =gcrd(C_5, y, $U_{5(y)}$); yields

$$D_{5(y)} = \begin{pmatrix} -x-1-xy & -1 \\ -(x+y)xy & 0 \end{pmatrix}$$

$$U_{5(y)} = \begin{pmatrix} -y & -1 \\ -x & -y \\ -1 & -1 \end{pmatrix}$$

$D_{6(x)}$ =gcrd(C_6, x, $U_{6(x)}$); yields

$$D_{6(x)} \begin{pmatrix} xy+y+1 & 1 \\ (x+y)yx & 0 \end{pmatrix}$$

$$U_{6(x)} = \begin{pmatrix} y & 1 \\ x & y \\ 1 & 1 \end{pmatrix}$$

$D_{6(y)}$=gcrd($C_6, y, U_{6(y)}$); yields

$$D_{6(y)} = \begin{pmatrix} -xy-y-1 & -1 \\ y^2-1+x & x-1+y \end{pmatrix}$$

$$U_{6(y)} = \begin{pmatrix} -x-2y+1 & -1 \\ -xy-x-y^2+y & -y \\ -x-y & -1 \end{pmatrix}$$

The GCDs computed with respect to x and with respect to y above are linked by unimodular matrices in a similar way to Example 7.6:

$$D_{i(x)} = U_i D_{i(y)} \quad \text{for} \quad i = 1,2,3,4,5,6$$

where

$$U_1 = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$U_2 = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$U_3 = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$U_4 = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$U_5 = \begin{pmatrix} 1 & 0 \\ x-1+y & -1 \end{pmatrix}$$

$$U_6 = \begin{pmatrix} -1 & 0 \\ 1-x-y & -1 \end{pmatrix}.$$

The following two extended examples show the precise details of the greatest common divisor procedures by displaying the parameters at each of the notable points in the procedures. Thus demonstrating the flow of the program as executed by the MAPLE kernel.

Example 7.8: The first of the extended examples demonstrates the `gcld` and the `shiftherm` procedures. The major intermediate results of the procedure are given together with the intermediate results of the of the `shiftherm` procedure. The first

two primitive factorisations are redundant as the matrix does not contain any primitive factors in x or y. Thus it is the `shiftherm` procedure that performs most of the calculations and to a lesser degree the `lprimfac` procedure.

$$D_1 = \begin{pmatrix} x+y & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & y & 1 \\ y & x & 1 \end{pmatrix}$$
$$= \begin{pmatrix} x+y & y(x+y) & x+y \\ 1+y & x+y & 2 \end{pmatrix}$$

`gcld(`D_1`,`x`,'`U`');` yields
→ enter `gcld` args = D_1, x, U
→ enter `transpose` args = D_1
← exit `transpose`

$$D_1^T := \begin{pmatrix} x+y & 1 \\ y(x+y) & x+y \\ x+y & 2 \end{pmatrix}$$

→ enter `gcrd` args = D_1^T, x, U
→ enter `otherindet` args = D_1^T, x
← exit `otherindet`

$$var1 := y$$

→ enter `rprimfac` args = D_1^T, x, $R0$
← exit `rprimfac`

$$M := \begin{pmatrix} x+y & 1 \\ y(x+y) & x+y \\ x+y & 2 \end{pmatrix}$$

$$R0 := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

→ enter `rprimfac` args = M, y, $R1$
← exit `rprimfac`

$$M := \begin{pmatrix} x+y & 1 \\ y(x+y) & x+y \\ x+y & 2 \end{pmatrix}$$

$$R1 := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

→ enter shiftherm args = M, x

$$mx := \left[\begin{pmatrix} x+y & 1 \\ y(x+y) & x+y \\ x+y & 2 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \right]$$

→ enter gaussredS args = mx, x

← exit gaussredS

$$mx := \left[\begin{pmatrix} x+y & 1+y \\ 0 & 1-y \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ -y+y^2-xy+x & 1-2y+y^2 & -y^3+y^2+xy-x \end{pmatrix} \right]$$

→ enter colredS args = mx, x

← exit colredS

$$M := \left[\begin{pmatrix} x+y-y^2-xy & 0 \\ 0 & 1-y \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 2 & 0 & -y-1 \\ -1 & 0 & 1 \\ -y+y^2-xy+x & 1-2y+y^2 & -y^3+y^2+xy-x \end{pmatrix} \right]$$

← exit shiftherm

$$H := \begin{pmatrix} x+y-y^2-xy & 0 \\ 0 & 1-y \\ 0 & 0 \end{pmatrix}$$

→ enter rowdim args = H

← exit rowdim

$$rdim := 3$$

→ enter coldim args = H

← exit coldim

$$cdim := 2$$

$$m := 2$$

→ enter `rowfactors` args = H, y

← exit `rowfactors`

$$H := \begin{pmatrix} -x-y & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

→ enter `submatrix` args = H, $1\ldots2$, $1\ldots2$

← exit `submatrix`

$$R := \begin{pmatrix} -x-y & 0 \\ 0 & 1 \end{pmatrix}$$

→ enter `lprimfac` args = R, y

← exit `lprimfac`

$$R_{prim} := \begin{pmatrix} -x-y & 0 \\ 0 & 1 \end{pmatrix}$$

→ enter `evalm` args = R_{prim}, $R1$, $R0$

← exit `evalm`

$$D := \begin{pmatrix} -x-y & 0 \\ 0 & 1 \end{pmatrix}$$

$$U := \begin{pmatrix} -1 & 1+y \\ -y & x+y \\ -1 & 2 \end{pmatrix}$$

← exit `gcrd`

$$D := \begin{pmatrix} -x-y & 0 \\ 0 & 1 \end{pmatrix}$$

$$U := \begin{pmatrix} -1 & 1+y \\ -y & x+y \\ -1 & 2 \end{pmatrix}$$

→ enter `transpose` args = U

← exit `transpose`

$$U := \begin{pmatrix} -1 & -y & -1 \\ 1+y & x+y & 2 \end{pmatrix}$$

→ enter `transpose` args = D
← exit `transpose`

$$D := \begin{pmatrix} -x-y & 0 \\ 0 & 1 \end{pmatrix}$$

← exit `gcld`

$$\begin{pmatrix} -x-y & 0 \\ 0 & 1 \end{pmatrix}$$ □

Example 7.9: The second of the extended examples principally demonstrates the `lprimfac` and the `shiftherm` procedures. The matrix has one primitive factor in y, thus the first primitive factorisation with respect to x does not extract any factors. However the second primitive factorisation, with respect to y, does extract a factor. When the `shiftherm` procedure is performed it is discovered that the coprime test is true, thereby terminating the procedure.

$$D_2 = \begin{pmatrix} 1 & y \\ y & x \\ 1 & 1 \end{pmatrix} \begin{pmatrix} xy+1 & -y \\ 1 & 0 \end{pmatrix}$$
$$= \begin{pmatrix} xy+y+1 & -y \\ xy^2+x+y & -y^2 \\ xy+2 & -y \end{pmatrix}$$

`gcrd(`D_2`,`x`,'`U`');` yields
→ enter `gcrd` args = D_2, x, U
→ enter `otherindet` args = D_2, x
← exit `otherindet`

$$var1 := y$$

→ enter `rprimfac` args = D_2, x, $R0$
← exit `rprimfac`

$$M := \begin{pmatrix} xy+y+1 & -y \\ xy^2+x+y & -y^2 \\ xy+2 & -y \end{pmatrix}$$

$$R0 := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

→ enter `rprimfac` args = D_2, y, $R1$
→ enter `transpose` args = D_2
← exit `transpose`

$$D_2^T := \begin{pmatrix} xy+y+1 & xy^2+x+y & xy+2 \\ -y & -y^2 & -y \end{pmatrix}$$

→ enter `lprimfac` args = D_2^T, y, S
→ enter `rowdim` args = D_2^T
← exit `rowdim`

$$rdim := 2$$

→ enter `coldim` args = D_2^T
← exit `coldim`

$$cdim := 3$$

→ enter `copy` args = D_2^T
← exit `copy`

$$M := \begin{pmatrix} xy+y+1 & xy^2+x+y & xy+2 \\ -y & -y^2 & -y \end{pmatrix}$$

→ enter `unitmx` args = 2
← exit `unitmx`

$$Q := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

→ enter `primroots` args = M, y
← exit `primroots`

$$s := [0]$$

→ enter `otherindet` args = M, y
← exit `otherindet`

$$var1 := x$$

→ enter subs args = M, $y = 0$
← exit subs

$$B := \begin{pmatrix} 1 & x & 2 \\ 0 & 0 & 0 \end{pmatrix}$$

→ enter emxc args = B
← exit emxc

$$B := \begin{pmatrix} 1 & x & 2 \\ 0 & 0 & 0 \end{pmatrix}$$

→ enter shiftherm args = B, x, U
← exit shiftherm

$$B := \begin{pmatrix} 1 & x & 2 \\ 0 & 0 & 0 \end{pmatrix}$$

$$U := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

→ enter multiply args = U, M
← exit multiply

$$M := \begin{pmatrix} xy+y+1 & xy^2+x+y & xy+2 \\ -y & -y^2 & -y \end{pmatrix}$$

→ enter factorow args = M, y
← exit factor

$$row := 2$$

→ enter unitmx args = 2
← exit unitmx

$$C := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$C := \begin{pmatrix} 1 & 0 \\ 0 & y \end{pmatrix}$$

→ enter evalm args = Q, U^{-1}, C
← exit evalm

$$Q := \begin{pmatrix} 1 & 0 \\ 0 & y \end{pmatrix}$$

→ enter emxc args = Q
← exit emxc

$$Q := \begin{pmatrix} 1 & 0 \\ 0 & y \end{pmatrix}$$

← exit lprimfac

$$Q := \begin{pmatrix} xy+y+1 & xy^2+x+y & xy+2 \\ -1 & -y & -1 \end{pmatrix}$$

$$R := \begin{pmatrix} 1 & 0 \\ 0 & y \end{pmatrix}$$

→ enter transpose args = Q
← exit transpose

$$Q := \begin{pmatrix} xy+y+1 & -1 \\ xy^2+x+y & -y \\ xy+2 & -1 \end{pmatrix}$$

← exit rprimfac

$$M := \begin{pmatrix} xy+y+1 & -1 \\ xy^2+x+y & -y \\ xy+2 & -1 \end{pmatrix}$$

→ enter shiftherm args = M, x
← exit shiftherm

$$H := \begin{pmatrix} 1-y^2 & 0 \\ 0 & 1-y \\ 0 & 0 \end{pmatrix}$$

→ enter indets args = op(H)
← exit indets

$$\texttt{indets} = y$$

$$U := \begin{pmatrix} xy+y+1 & -1 \\ xy^2+x+y & -y \\ xy+2 & -1 \end{pmatrix}$$

← exit gcrd

$$\begin{pmatrix} 1 & 0 \\ 0 & y \end{pmatrix}$$ □

The examples upto this point have demonstrated the important features of the procedures but have been relatively simple. This has been necessary so that the examples can easily be checked, by hand, and also for conciseness. The following examples are slightly larger matrices. However, the power and capacity of the host computer has prevented larger examples being tested.

Example 7.10: Consider the 5×4 matrix defined by

$$F_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x & 0 & 0 & 0 \\ 1 & y+2 & 0 & 0 \\ 1 & 1 & x+y & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & y+2 & 0 & 0 \\ 2 & y+3 & x+y & 0 \\ 1 & 1 & 1 & 1 \\ x+2 & 2 & x+y+1 & 1 \end{pmatrix}$$

gcrd(F_1,x,'U_x'); yields

$$\begin{pmatrix} 1 & y+2 & 0 & 0 \\ -2-2y & xy+2x-4y-3-y^2 & -y-xy-y^2-x & 0 \\ 1 & 1 & 1 & 1 \\ 2(x+y)(y+2) & (x+y)(y+2)(y+3) & (x+y)(y+2)y & 0 \end{pmatrix}$$

$$U_x = \begin{pmatrix} -(2y^2+6y+3)x & y(y+2) & 0 & 1+y \\ 1 & 0 & 0 & 0 \\ 4x+2xy & -y-1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ x-1-4xy-2xy^2 & y^2+y-1 & 1 & y \end{pmatrix}$$

gcrd(F_1,y,'U_y'); yields

$$\begin{pmatrix} \frac{1}{8} & \frac{1}{8}y+\frac{1}{4}+\frac{1}{8}x & \frac{1}{8}xy+\frac{1}{8}x^2 & 0 \\ \frac{1}{2} & \frac{1}{4}y+\frac{3}{4}+\frac{1}{4}x & \frac{1}{4}xy+\frac{1}{4}y+\frac{1}{4}x+\frac{1}{4}x^2 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & \frac{1}{16}x(x+y) & \frac{1}{16}x(x+y)(x-2) & 0 \end{pmatrix}$$

$$U_y = \begin{pmatrix} -24x & 8x & 0 & 16 \\ 8-16x & 4x & 0 & 16 \\ -16x & 4+4x & 0 & 16 \\ 0 & 0 & 1 & 0 \\ -24x-8 & 8x+4 & 1 & 16 \end{pmatrix} \qquad \square$$

Example 7.11: Consider the matrix F_2 formed by

$$F_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x+3 & 1 & 1 & 1 \\ 0 & y & 1 & 1 \\ 0 & 0 & x^2+y & 1 \\ 0 & 0 & 0 & x^2 \end{pmatrix}$$

$$= \begin{pmatrix} x+3 & 1 & 1 & 1 \\ 0 & y & 1 & 1 \\ 0 & y & 1+x^2+y & 2 \\ 0 & 0 & 0 & x^2 \\ x+3 & 1 & 1+x^2+y & 2+x^2 \end{pmatrix}$$

`gcrd(`F_2`,`x`,'`U_x`');` yields

$$\begin{pmatrix} \frac{1}{6} & 0 & \frac{1}{3}y+\frac{1}{6}xy+\frac{1}{6}x^2+\frac{1}{3}x & 0 \\ -1 & -\frac{1}{2} & -\frac{3}{2}y-\frac{1}{2}xy-\frac{3}{2}x-\frac{1}{2}x^2 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & -\frac{1}{24}(x+y)x(x+2) & 0 \end{pmatrix}$$

$$U_x = \begin{pmatrix} 6x & 0 & 0 & 24 \\ -18-12x & -2x-4 & 0 & -24 \\ -24-12x & -2x-6 & 0 & -24 \\ 0 & 0 & 1 & 0 \\ 6x-6 & -2 & 1 & 24 \end{pmatrix}$$

`gcrd(`F_2`,`y`,'`U_y`');` yields

$$\begin{pmatrix} \frac{1}{6} & 0 & \frac{1}{3}y+\frac{1}{6}xy+\frac{1}{6}x^2+\frac{1}{3}x & 0 \\ 1 & \frac{1}{2} & \frac{3}{2}y+\frac{1}{2}xy+\frac{3}{2}x+\frac{1}{2}x^2 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & \frac{1}{24}(x+y)x(x+2) & 0 \end{pmatrix}$$

$$U_y = \begin{pmatrix} 6x & 0 & 0 & -24 \\ -18-12x & 2x+4 & 0 & 24 \\ -24-12x & 2x+6 & 0 & 24 \\ 0 & 0 & 1 & 0 \\ 6x-6 & 2 & 1 & -24 \end{pmatrix} \qquad \square$$

It may be argued that these are not large matrices, but to perform the GCD algorithm manually with matrices of this size is very time consuming.

7.3 CPU Time Considerations

It has been seen in the previous section that the documented program delivers the expected results. It is the purpose of this section to consider the time taken for such calculations, expressed in the number of seconds used by the CPU (Central Processing Unit).

It is instructive to consider the examples in three sets, due to the similarity of dimensions. Firstly the matrices $B_1, B_2, \ldots, B_6$, secondly matrices $C_1, C_2, \ldots, C_6$ and thirdly matrices F_1, F_2. Arguably the first two sets may be compared together but the second set is composed from the same three matrix factors. To compare the times more reliably a number of calculations were performed and the average taken: typically the calculations were performed twenty times.

The times recorded for the $B_1, B_2, \ldots, B_6$ are as follows.

A	`gcrd(`A`,`x`,'`U_x`');`	`gcrd(`A`,`y`,'`U_y`');`
B_1	7.33	6.00
B_2	9.70	9.82
B_3	8.83	10.48
B_4	19.50	20.57
B_5	7.33	8.03
B_6	49.23	52.43

Figure 7.2: CPU time in seconds

To explain the results consider the GCRD Algorithm and the number of steps in which the procedures perform the calculations. Thus comparing B_1 and B_2. The former only possess one primitive root with respect to one indeterminate and the second possesses primitive roots in both indeterminates therefore as expected the results for B_2 take longer than those for B_1. The calculations for matrix B_3 are performed in the `shiftherm` and `rowfactors` procedures therefore comparing the times with B_1 and B_2 is fairly meaningless. The matrix B_4 not only contains primitive roots in both indeterminates but some are complex thereby explaining the increased time taken.

A seemingly surprising result is obtained by considering the time taken for the calculations using matrices B_5 and B_6. Both of these possess a secondary matrix but the time taken for B_6 is almost seven times as long as the time taken for B_5. This discrepancy can be explained by considering the 2-D Hermite form: for B_5 `shiftherm` delivers a matrix with polynomial elements with degrees less than ten, however, for B_6 the elements have degree upto twenty-five and additionally have six digit integer coefficients. Therefore the extra time is used in computing the 2-D Hermite form.

From the above discussion it appears that the most time consuming process is the computation of the 2-D Hermite form. The reason for this is that the degrees of the polynomial elements of the matrix in the 2-D Hermite procedure are relatively large compared to the degrees in other procedures. Therefore the number of elementary operations and invocations of the pseudo-division algorithm is relatively high.

The second set of examples are derived by multiplying a coprime matrix, A_0, by three factors corresponding to the three types of matrix factor discussed in Section 4.5.4 (GCD modifications).

A	gcrd(A,x,'U_x');	gcrd(A,y,'U_y');
C_1	15.39	15.29
C_2	14.90	16.24
C_3	14.75	15.48
C_4	15.81	15.04
C_5	19.81	15.51
C_6	14.60	39.20

Figure 7.3: CPU time in seconds

Thus in the first four cases the time taken for each calculation is similar, but it is more efficient to perform the calculation with respect to y in the fifth matrix, C_5. A more marked difference is seen between the calculation of C_6 firstly with respect to x and secondly with respect to y (the latter takes over twice as long to perform the calculation). By examining the individual steps of the procedure it is discovered that the 2-D Hermite procedure delivers a matrix with polynomial elements upto degree 37 and containing nine digit integer coefficients; the highest polynomial degree in the other 2-D Hermite forms is seven (in C_5 with respect to x) with single digit coefficients.

Thus once again the 2-D Hermite form accounts for the increased time taken.

Finally the timestaken for the larger examples are:

A	gcrd(A,x,'U_x');	gcrd(A,y,'U_y');
F_1	44.98	42.17
F_2	43.8	46.88

Figure 7.4: CPU time in seconds

Thus, although the matrices have more elements than those considered in Figures 7.2 and 7.3, the time taken to compute the GCDs are not overwhelmingly different. This is because of the relatively simple nature of the matrix structure. However, more complicated examples have been prohibited by the power and storage capacity of the host machine.

7.4 Further Discussion

In the previous section numerous examples of matrices with various sizes have been tested and a greatest common divisor has been returned as the result of the procedure. However, the picture is not yet complete. Consider computing the greatest common right divisor of the matrix, $A(x,y)$, via `gcrd(A,x);` where

$$A = \begin{pmatrix} x^3y + xy + y + 1 & 1 \\ x^4 + x^2 + 2x & x \\ x^3 + x + y + 1 & y \end{pmatrix}$$

The result of this command is the error message:

```
error in sprem, invalid arguments to divide
```

The reason for this failure can be traced to the simplification rules used by MAPLE to simplify radicals, i.e. rational powers of expressions. The problem is hi-lighted by considering the polynomial $a(x) = x^3 + x + 1$. The zeros of this polynomial, as calculated by MAPLE using the command:

`solve(a=0,x);`

are

$$s_1 \equiv -\left(\frac{1}{2} - \frac{\sqrt{31}}{6\sqrt{3}}\right)^{\frac{1}{3}} - \left(\frac{1}{2} + \frac{\sqrt{31}}{6\sqrt{3}}\right)^{\frac{1}{3}} \tag{284}$$

$$s_2 \equiv \frac{1}{2}\left[\left(\frac{1}{2} - \frac{\sqrt{31}}{6\sqrt{3}}\right)^{\frac{1}{3}} + \left(\frac{1}{2} + \frac{\sqrt{31}}{6\sqrt{3}}\right)^{\frac{1}{3}} + \left(\left(\frac{1}{2} + \frac{\sqrt{31}}{6\sqrt{3}}\right)^{\frac{1}{3}} - \left(\frac{1}{2} - \frac{\sqrt{31}}{6\sqrt{3}}\right)^{\frac{1}{3}}\right) i\sqrt{3}\right], \tag{285}$$

$$s_3 \equiv \frac{1}{2}\left[\left(\frac{1}{2} - \frac{\sqrt{31}}{6\sqrt{3}}\right)^{\frac{1}{3}} + \left(\frac{1}{2} + \frac{\sqrt{31}}{6\sqrt{3}}\right)^{\frac{1}{3}} - \left(\left(\frac{1}{2} + \frac{\sqrt{31}}{6\sqrt{3}}\right)^{\frac{1}{3}} - \left(\frac{1}{2} - \frac{\sqrt{31}}{6\sqrt{3}}\right)^{\frac{1}{3}}\right) i\sqrt{3}\right], \tag{286}$$

However, when these roots are substituted into the polynomial a and

`simplify(a,radical);`

is executed MAPLE delivers the following result

$$\frac{-1}{2 \times 3^{\frac{2}{3}} 6^{\frac{1}{3}}}\left[\begin{array}{c}(3\sqrt{3} - \sqrt{31})^{\frac{2}{3}}(3\sqrt{3} + \sqrt{31})^{\frac{1}{3}} 6^{\frac{1}{3}} 3^{\frac{1}{6}} + (3\sqrt{3} - \sqrt{31})^{\frac{1}{3}}(3\sqrt{3} + \sqrt{31})^{\frac{2}{3}} 6^{\frac{1}{3}} 3^{\frac{1}{6}} \\ + 2\sqrt{3}(3\sqrt{3} - \sqrt{31})^{\frac{1}{3}} + 2\sqrt{3}(3\sqrt{3} + \sqrt{31})^{\frac{1}{3}}\end{array}\right] \tag{287}$$

This expression is in fact zero, but not recognised as such by MAPLE. The crucial result appears to be the difference of two squares when raised to a fractional power, i.e.

$$(a+b)^{\frac{1}{3}}(a-b)^{\frac{1}{3}} = (a^2 - b^2)^{\frac{1}{3}} \tag{288}$$

using this result in (288) the following factored form is derived

$$\frac{\left((-4)^{\frac{1}{3}}6^{\frac{1}{3}}3^{\frac{1}{6}}+2\sqrt{3}\right)\left[(3\sqrt{3}-\sqrt{31})^{\frac{1}{3}}+(3\sqrt{3}+\sqrt{31})^{\frac{1}{3}}\right]}{2\times 3^{\frac{2}{3}}6^{\frac{1}{3}}} \tag{289}$$

Now

$$(-4)^{\frac{1}{3}}6^{\frac{1}{3}}3^{\frac{1}{6}}=(-24)^{\frac{1}{3}}(\sqrt{3})^{\frac{1}{3}}=-(\sqrt{1728})^{\frac{1}{3}}=-\sqrt{1728^{\frac{1}{3}}}=-\sqrt{12}=-2\sqrt{3} \tag{290}$$

Thus the first bracket in (289) is zero. Hence the expression (287) is zero.

The failure of MAPLE to simplify expressions of this kind causes the primitive factorisation algorithm to fail when calculating the modified Hermite form of the matrix, formed by substitution of an indeterminate by its primitive root. In particular the last row of the modified Hermite form does not become identically equal to zero using MAPLE's simplification rules and indeed may cause division by zero if all elements are evaluated fully.

Clearly the primitive factorisation procedure needs to be modified so that examples involving complicated primitive roots such as those described above do not fail to be computed.

One solution to this problem may be achieved by replacing the polynomial by a product of its linear factors. Consider this procedure in the context of the above discussion.

The roots of $a(x) = x^3 + x + 1$ are given by s_1, s_2, s_3 as given in (284)–(286), thus every occurrence of $x^3 + x + 1$ could be replaced by its linear factorisation $(x - s_1)(x - s_2)(x - s_3)$. This could be achieved by the following.

```
a:=x^3+x+1;
s:=solve(a=0,x);
M:=simplify(op(A),{a=(x-s[1])*(x-s[2])*(x-s[3])});
```

where A is the matrix in which the substitution is to be made, op is used to indicate that it is the operands, i.e. the elements, of A that are to have the substitution performed on and not the name A. M then contains the all occurrences of a replaced by its linear factorisation.

This process can be achieved in general for the polynomial poly by the procedure linearfac below.

```
linearfac:=proc(poly)
local s,m;
```

```
s:=solve(poly);
m:=map(proc(x,y) ;y-x end,
[s],
op(indets(poly)));
convert(m,'*')
end;
```

Thus returning to the above polynomial $x^3 + x + 1$

```
linearfac(x^3+x+1);
```

delivers

$$(x - s_1)(x - s_2)(x - s_3)$$

where s_1, s_2, s_3 are given by (284)–(286). Occurrences of the polynomial $x^3 + x + 1$ are replaced by the linear factorisation $(x - s_1)(x - s_2)(x - s_3)$ by executing the commands:

```
a:=x^3+x+1;
simplify(A,{a=linearfac(a)});
```

However, this solution does not rectify the problem for all cases because the polynomial that defines the primitive roots is not necessarily present as a factor of the matrix elements. Consider forming the left primitive factorisation of the matrix

$$C(x, y) = \begin{pmatrix} x^3y & x+1 \\ -y & 1 \end{pmatrix}$$

over the ring $\mathbb{R}[x][y]$. The determinant of the matrix is given by

$$\det C(x, y) = x^3y + (x + 1)y = y(x^3 + x + 1)$$

Therefore the primitive roots are given by the solution to the equation

$$(x^3 + x + 1) = 0$$

i.e. the three numbers given by (284)–(286). Thus the primitive roots are defined by the zeros of the polynomial $x^3 + x + 1$. However, the procedure described above does not have any effect on the elements of the matrix since $x^3 + x + 1$ does not occur within any of the matrix elements.

Thus the procedure of replacing the primitive factor defining polynomial by its linear factorisation has limited benefits but does allow the greatest common divisor of a larger class of matrices to be computed. A longer term solution is provided by extracting the primitive factor defining polynomials without computing the roots of

the polynomial. Additionally, a procedure of this type would remove the necessity of using complex numbers when deriving the primitive factorisation. The theoretical existence of such a procedure is provided by [35].

The philosophy of the procedure is to remove the irreducible factors of the primitive root-defining-polynomial one at a time by working with the matrix modulo the primitive factors and performing a similar type of reduction as described in the primitive factorisation algorithm. Therefore the coefficient field of the polynomials does not have to be algebraically closed. For example, if the coefficient field of the polynomial matrix elements is the reals, i.e. the matrix elements belong to $\mathbb{R}[z_1, z_2]$, all calculations may be performed using only real numbers. The process used here is to calculate all the primitive roots and then to remove them one at a time; thus some of these roots may involve complex numbers. As it has been noted complex numbers appear to be slightly unpredicable when used in predefined functions in the `linalg` package. If the primitive roots of a matrix are all real the two algorithms are essentially identical.

Thus the algorithm of Guiver and Bose [35] has addressed the problem of using complex numbers but by requiring the polynomial to be expressed in terms of irreducible factors it does not address the real problem of radical expressions, i.e. terms to a fractional power. Therefore it is believed that the additional effort involved in the coding of this algorithm is not justified.

7.5 Application to Coprime MFD

Recall the definition of a matrix fraction description (MFD) and the method of formation as proposed in Chapter 2. Suppose that $G_{p\times q}(z_1, z_2)$ is a matrix with rational elements in the two indeterminates z_1 and z_2. A matrix fraction description is given by

$$G(z_1, z_2) = D_1^{-1}(z_1, z_2)N_1(z_1, z_2) \qquad \text{left MFD} \qquad (291)$$

$$= N_2(z_1, z_2)D_2^{-1}(z_1, z_2) \qquad \text{right MFD} \qquad (292)$$

where $N_1(z_1, z_2)$, $N_2(z_1, z_2)$, $D_1(z_1, z_2)$ and $D_2(z_1, z_2)$ are polynomial matrices with dimensions $p \times q$, $p \times q$, $p \times p$ and $q \times q$, respectively.

A coprime MFD is defined by insisting that the polynomial matrices in (291) and (292) form a coprime pair; thus defining two types of coprime MFD corresponding to the two notions of coprimeness, namely minor or zero coprime. Explicitly $D_1(z_1, z_2)$, $N_1(z_1, z_2)$ are said to form a minor (left) coprime MFD (zero (left) coprime MFD) of $G(z_1, z_2)$ if

$$G(z_1, z_2) = D_1^{-1}(z_1, z_2)N_1(z_1, z_2)$$

and additionally $D_1(z_1, z_2)$, $N_1(z_1, z_2)$ form a minor (left) coprime (zero (left) coprime) pair of matrices. Similar statements can be made for coprime right MFDs by insisting that $D_2(z_1, z_2)$, $N_2(z_1, z_2)$ form a minor (right) coprime (zero (right) coprime) pair of matrices.

The process by which a left coprime MFD can be formed is given by the procedure detailed in Algorithm 2.1 and by incorporating the program documented in Chapter 6, the following is obtained.

Algorithm 7.1: (*Left coprime MFD*)
Let $G_{p\times q}(z_1, z_2)$ be the matrix defined by

$$G_q(z_1, z_2) = \begin{pmatrix} g_{11}(z_1, z_2) & g_{12}(z_1, z_2) & \cdots & g_{1q}(z_1, z_2) \\ g_{21}(z_1, z_2) & g_{22}(z_1, z_2) & \cdots & g_{2q}(z_1, z_2) \\ \vdots & \vdots & \ddots & \vdots \\ g_{p1}(z_1, z_2) & g_{p2}(z_1, z_2) & \cdots & g_{pq}(z_1, z_2) \end{pmatrix}$$

where the elements $g_{ij}(z_1, z_2)$ for $i = 1, 2, \ldots, p$ and $j = 1, 2, \ldots, q$ are rational functions in the two indeterminates z_1 and z_2.

Step 1: Form $d_i(z_1, z_2)$, the least common denominator of the ith row of $G_{p\times q}(z_1, z_2)$, for $i = 1, 2, \ldots, p$.

Step 2: Construct the diagonal matrix $D_{p\times p}(z_1, z_2)$,

$$D(z_1, z_2) = \begin{pmatrix} d_1(z_1, z_2) & 0 & \cdots & 0 \\ 0 & d_2(z_1, z_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_p(z_1, z_2) \end{pmatrix}$$

Step 3: Form the polynomial matrix $N_{p\times q}(z_1, z_2) = D_{p\times p}(z_1, z_2)G_{p\times q}(z_1, z_2)$.

Step 4: Define the matrix $A(z_1, z_2)$ by augmenting the matrices $D(z_1, z_2)$ and $N(z_1, z_2)$ as derived in Steps 2 and 3. Thus

$$A(z_1, z_2) = [\, D(z_1, z_2) \quad N(z_1, z_2) \,]$$

Step 5: Compute the greatest common left divisor, $Q(z_1, z_2)$, by executing the command

$$\texttt{gcld}(A, var, \texttt{'U'});$$

where *var* is either z_1 or z_2.

Step 6: The coprime MFD is given by a suitable partition of the matrix U, i.e. define $\bar{D}(z_1, z_2)$ as the first p columns of U and $\bar{N}(z_1, z_2)$ as the last q columns of U. Then $\bar{D}(z_1, z_2)$ and $\bar{N}(z_1, z_2)$ form a coprime pair of matrices and by construction

$$[\, D(z_1, z_2) \quad N(z_1, z_2) \,] = Q(z_1, z_2)\, [\, \bar{D}(z_1, z_2) \quad \bar{N}(z_1, z_2) \,].$$

Then

$$\begin{aligned} G(z_1, z_2) &= D^{-1}(z_1, z_2)N(z_1, z_2) \\ &= (\bar{D}(z_1, z_2)^{-1}Q^{-1}(z_1, z_2))(Q(z_1, z_2)\bar{N}(z_1, z_2)) \\ &= \bar{D}(z_1, z_2)^{-1}\bar{N}(z_1, z_2). \end{aligned}$$

Thus by construction $\bar{D}(z_1, z_2)$, $\bar{N}(z_1, z_2)$ necessarily form a minor (left) coprime MFD of $G(z_1, z_2)$. Additionally, by Theorem 2.2, $\bar{D}(z_1, z_2)$, $\bar{N}(z_1, z_2)$ may also be zero (left) coprime, depending on the nature of $G(z_1, z_2)$. □

The following example illustrates the algorithm defined above.

Example 7.12: Consider forming the left coprime MFD of the rational matrix $G(z_1, z_2)$ defined by

$$G(z_1, z_2) = \begin{pmatrix} z_1 & z_2 + 1 \\ \dfrac{z_1 + z_2}{z_2} & \dfrac{z_2}{z_1} \\ \dfrac{z_1}{z_2} & 1 \end{pmatrix}$$

Step 1: The least common denominators are given by

$$d_1(z_1, z_2) = 1, \qquad d_2(z_1, z_2) = z_1 z_2, \qquad d_3(z_1, z_2) = z_2$$

Step 2: The diagonal matrix $D(z_1, z_2)$ is thus defined as

$$D(z_1, z_2) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & z_1 z_2 & 0 \\ 0 & 0 & z_2 \end{pmatrix}$$

Step 3: The polynomial matrix $N(z_1, z_2) = D(z_1, z_2)G(z_1, z_2)$ is therefore defined by

$$N(z_1, z_2) = \begin{pmatrix} z_1 & z_2 + 1 \\ z_1(z_1 + z_2) & z_2^2 \\ z_1 & z_2 \end{pmatrix}$$

Step 4: The polynomial matrix $A(z_1, z_2)$ is defined by

$$A(z_1, z_2) = \begin{pmatrix} 1 & 0 & 0 & z_1 & z_2 + 1 \\ 0 & z_1 z_2 & 0 & z_1(z_1 + z_2) & z_2^2 \\ 0 & 0 & z_2 & z_1 & z_2 \end{pmatrix}$$

Step 5: The result of the MAPLE statement `gcld(A,x,'U');` is

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & x & y \\ 0 & 1 & 0 \end{pmatrix}$$

Step 6: By partitioning the matrix $U(z_1, z_2)$ the coprime MFD is given by

$$G_{p\times q}(z_1, z_2) = \bar{D}^{-1}_{p\times p}(z_1, z_2) N_{p\times q}(z_1, z_2) = \begin{pmatrix} 1 & 0 & -z_2 \\ 0 & 0 & z_2 \\ 0 & z_1 & -z_1 \end{pmatrix}^{-1} \begin{pmatrix} 0 & 1 \\ z_1 & z_2 \\ z_1 & z_2 - z_1 \end{pmatrix} \qquad \square$$

Algorithm 7.1 may be automated in MAPLE by defining the procedure `lmfd`.

The Function:

Syntax: `lmfd(`G`,`x`,'`D`');`

Parameters: G - a two dimensional array,
x - an unevaluated name,
D - a place holder for the denominator matrix.

The matrix G is a rational polynomial matrix in two indeterminates, one of which is x. The result of the function is the numerator matrix of a coprime MFD of G. The denominator matrix is returned in D. Thus $G = D^{-1}N$, where N is the result of the function.

Procedure Code:

```
 8      lmfd:=proc(G,var,D)
 9        local i,j,d,n,s,rdim,cdim,M,N,U;
10          rdim:=rowdim(G);
11          cdim:=coldim(G);
12          d:=array(diagonal,1..rdim,1..rdim);
13          for i to rdim do
14            for j to cdim do
15              s[j]:=denom(G[i,j])
16            od;
17          s:=convert(s,set);
18          d[i,i]:=lcm(op(s));
19          od;
20          n:=multiply(d,G);
21          M:=array(1..rdim,1..rdim+cdim);
22          for i to rdim do
23            for j to rdim+cdim do
24              if j<=rdim then M[i,j]:=d[i,j] else
25              M[i,j]:=n[i,j-rdim] fi
26            od
27          od;
28          if gcdminors(M)=1 then D:=copy(d); RETURN(op(n)) fi;
29          gcld(M,var,'U');
30          D:=submatrix(U,1..rdim,1..rdim);
31          N:=submatrix(U,1..rdim,rdim+1..rdim+cdim)
32        end;
```

Formal Parameters:

`G` - a two dimensional array;

`var` - an unevaluated name;

`D` - a place holder to store the denominator matrix.

Local Variables:

`i` - an index of iteration;

`j` - an index of iteration;

`d` - a two dimensional array, containing the non-coprime denominator matrix;

`n` - a two dimensional array, containing the non-coprime numerator matrix;

`s` - a one dimensional array, containing the denominator of a particular row of `G`;

`rdim` - the number of rows in `G`;

`cdim` - the number of columns in `G`;

`M` - a two dimensional array, containing the augmented matrices `d` and `n`;

`N` - the coprime numerator matrix;

`U` - the coprime matrix resulting from the `gcld` procedure.

Line 1 defines the `lmfd` as the procedure name.

Line 2 declares the local variables.

Line 3 assigns `rdim` the number of rows in `G`.

Line 4 assigns `cdim` the number of columns in `G`.

Line 5 assigns `d` as a square matrix with elements not on the principal diagonal identically zero.

Lines 6-12 Commutes the least common denominator of each row of `G` by finding the lowest common multiple of the denominators, expressed as a set, `s`.

Line 13 assigns `n` as the polynomial matrix resulting from the multiplication of `d` by `G`.

Line 14 assigns `M` as a two dimensional array with `rdim` rows and `rdim+cdim` columns.

Lines 15-20 assigns the two dimensional array `M` formed by augmenting the matrix `n` to `d`.

Line 21 decides if the matrix `M` is coprime; if so this matrix is returned as the denominator and numerator matrices.

Line 22 computes the greatest common left divisor of the matrix `M` with respect to `var` and assigns `U` as the coprime matrix resulting from extracting the GCD.

Lines 23-24 assigns `D` and `N` as the denominator and numerator matrices respectively, with the numerator matrix being the result of the procedure.

Line 25 terminates the procedure.

A right coprime MFD may be calculated by transposing the matrix G and then performing a left MFD, using `lmfd`. The right coprime MFD is the transpose of the result obtained from the `lmfd` procedure. This may be defined as:

The Function:

Syntax: `rmfd(`G`,`x`,'`D`');`

Parameters: G - a two dimensional array,
x - an unevaluated name,
D - a place holder for the denominator matrix.

The matrix G is a rational polynomial matrix in two indeterminates, one of which is x. The result of the function is the numerator matrix of a coprime MFD of G. The denominator matrix is returned in D. Thus $G = ND^{-1}$, where N is the result of the function.

Procedure Code:

```
rmfd:=proc(G,var,D)
      local H,N;
        H:=transpose(G);
        N:=lmfd(H,var,D);
        D:=transpose(D);
        N:=transpose(N)
      end;
```

Formal Parameters:

`G` - a two dimensional array;
`var` - an unevaluated name;
`D` - a place holder to store the denominator matrix.

Local Variables:

`H` - the transpose of the matrix `G`;
`N` - the numerator matrix of the MFD.

Line 1 defines `rmfd` as the name of the procedure.
Line 2 declares the local variables.
Line 3 transposes the matrix `G`.
Line 4 assigns `N` as the numerator matrix of the left MFD of `H`.
Line 5 transposes the denominator matrix of the left coprime MFD.
Line 6 transposes the numerator matrix computed in line 4: this is the result of the procedure.
Line 7 terminates the procedure.

Thus using this on the above example

`lmfd(`G`,`x`,'D');` yields

$$\begin{pmatrix} 1 & 0 & -z_2 \\ 0 & 0 & z_2 \\ 0 & z_1 & -z_1 \end{pmatrix}$$

$$D = \begin{pmatrix} 0 & 1 \\ z_1 & z_2 \\ z_1 & z_2 - z_1 \end{pmatrix}$$

□

7.6 Conclusions

In Part Two of this thesis a program has been documented that automatically calculates the greatest common divisor of a polynomial matrix. This program is based on certain theoretical ideas by [33], [34]. However, the algorithms as described within those works are unsuitable for implementation into an algebraic manipulation language for two reasons. Firstly, the algorithms are fundamentally theoretically deficient and secondly practical considerations suggest modifications to increase the efficiency of the procedure.

The one major efficiency improvement is the test for coprimeness after the 2-D Hermite formation. A test for coprimeness at an earlier stage is prohibited by the number of calculations that are necessary. The evidence provided by the examples demonstrate that the formation of the 2-D Hermite form would result in an efficiency loss due to the necessary number of elementary operations and use of the pseudo-division algorithm. This is attributable to the high degrees of the polynomial elements in the 2-D Hermite form. The alternative method of checking for a coprime matrix by computing the high-order minors is equally prohibitive due to the number of calculations required in computing a number of determinants and then checking for factors of the polynomials so formed.

Thus the program can only be made more efficient by improving the individual procedures that constitute the GCRD algorithm, namely the modified Hermite procedure and the primitive factorisation procedure, and also the procedures contained there-in.

From the evidence provided by the examples it is an improvement in the speed of the 2-D Hermite form that would contribute the most to an improvement in the overall program. However, the speed of the 2-D Hermite form is governed almost exclusively by the speed that elementary operations can be performed and the calculation of the pseudo-division algorithm. The procedure that effects these calculations are predefined MAPLE procedures. Thus an increase in efficiency of the 2-D Hermite form can only be achieved by rewriting the predefined procedures in MAPLE.

The main theoretical contributions of this work are two fold. The first contribution is the rigorous definition via a constructive algorithm of the Hermite form of a matrix that does not have full rank. This seems to have been overlooked in the literature and certainly not accounted for in the original formulation ideas [33], [34]. The second contribution is the development of a simple and efficient coprimeness test arising from the 2-D Hermite form. Of course, these theoretical contributions are in addition to

the major practical contribution of forming the GCD of a 2-D polynomial matrix: a process that is very lengthy and complicated to apply without the aid of a symbolic manipulator or some other method of manipulating 2-D polynomial matrices. One of the main applications of this program is to the realisation of transfer function matrices to form coprime MFDs. The automatic procedure for achieving these MFDs has been given in Section 7.5.

It has been seen from the test examples presented in Section 7.2 that the program is limited by the nature of the primitive roots possessed by the matrices. The program fails to complete the automatic procedure due to the type of numbers contained in the calculations, namely integers to fractional powers (termed radicals). These numbers arise from a need to work with exact numbers; the rounding errors introduced by decimal notation are clearly unacceptable in this type of computation. However, the size of the matrices are limited by the power storage capacity of the host computer.

The solution to the failure of the primitive factorisation procedure problem is not entirely clear, but by considering the MAPLE internal representation of polynomial roots either additional rules need to be defined in order to deal with examples of the type detailed in Section 7.4 or the radicals must be expressed in an alternative form. MAPLE provides a data structure called `RootOf` which is used for the internal representation of algebraic numbers, i.e. solutions of polynomials. For example, the internal representation of the solutions to $x^3 + x + 1 = 0$ are

$$\texttt{RootOf(\_Z}^3\texttt{+\_Z+1)}$$

The actual algebraic number representation of these results is obtained by using the function `allvalues`: as used by the `roots` procedure. The simplification rules of MAPLE are defined to recognise `RootOf` expressions thus the type of problems encountered in the simplification of expressions involving radicals would not arise. However, it does introduce the problem of working with expressions that contain `RootOf`.

It is the author's belief that the solution to this problem lies with using the `RootOf` notation and not with a new procedure to calculate the primitive factorisation. Although improvements in efficiency may be gained by a new procedure.

As a final note to this program documentation a reference is made to an alternative symbolic manipulator called *Mathematica*. In light of the problems encountered in MAPLE one of the first considerations to be made when selecting a symbolic

manipulator for the calculation of a greatest common divisor is the computation and representation of polynomial roots. As demonstrated above MAPLE can not always simplify expressions involving radicals, such as those given in Section 7.4. This problem is not encountered when using *Mathematica*. For example the roots of the polynomial $x^3 + x + 1$ are s_1, s_2, s_3 as given by (284)–(286). The *Mathematica* command to compute these roots is given by

```
Solve[x^3+x+1==0,x]
```

If subsequently these roots are substituted into the polynomial and the expression simplified using the command `Simplify` the result is zero.

Thus *Mathematica* appears to be the better symbolic manipulator for producing the primitive factorisation of a polynomial matrix. However, the procedure code used to implement the algorithms in MAPLE is not directly transferable to *Mathematica* due to the two packages using different programming languages. Perhaps the best short term answer is to perform the GCD calculations in MAPLE and then pass the resulting matrix to *Mathematica* for simplification at the relevant stage of the primitive factorisation procedure. If *Mathematica* is installed in a UNIX environment it possesses the capability of calling external programs that exist within the UNIX environment. Thus the MAPLE procedures may be called from *Mathematica*.

A more economical and efficient solution is to translate the MAPLE code into *Mathematica* code using a translation program. This is theoretically possible because the C programming language is the basis of both the MAPLE and *Mathematica* kernels. If such a translator is not available the last resort is to recode the algorithms in *Mathematica*'s programming language.

Thus recommendations for future work is three-fold. Firstly to link *Mathematica* and MAPLE in a UNIX environment. Secondly, to encode the algorithms in *Mathematica* either by translating the MAPLE program or by rewriting the code. Thirdly, to modify the MAPLE program documented here using the `RootOf` notation.

References

[1] Gantmakher, F.R., *Theory of Matrices*, Chelsea Publishing Company: New York, 1959.

[2] MacDuffee, C.C., *The Theory of Matrices*, Chelsea Publishing Company: New York, 1946.

[3] van der Waerden, B., *Modern Algebra*, Fredrick Ungar, 1953.

[4] Bocher, M., *Introduction to Higher Algebra*, Macmillan: New York, 1907.

[5] Hungerford, T.W., *Algebra*, Holt Rinehart and Winston, 1974.

[6] Birkhoff, G., MacLane, S., *A Survey of Modern Algebra*, Macmillan: New York, 1941.

[7] Albert, A.A., *Modern Higher Algebra*, University of Chicago, 1937.

[8] MacDuffee, C.C., *Introduction to Abstract Algebra*, New York: Wiley, 1940.

[9] Atiyah, M.F., MacDonald, I.G., *Introduction to Commutative Algebra*, Reading, Mass.:Addison-Wesley, 1969.

[10] Zariski, O., Samuel, P., *Commutative Algebra*, 2 Volumes, New York: Springer-Verlag, 1976.

[11] Sontag, E.D., "Linear systems over commutative rings: A survey", Richerche di Automatica, Vol. 7, No. 1, pp1–34, 1976.

[12] Rosenbrock, H.H., *State-space and Multivariable Theory*, Nelson, 1970.

[13] Wolovich, W.A., *Linear Multivariable Systems*, New York: Springer-Verlag, 1974.

[14] Kailath, T., *Linear Systems*, Englewood Cliffs, N.J., Prentice Hall, 1980.

[15] Bose, N.K., "New results and techniques in multidimensional problems", Journal of Franklin Institute, Vol. 301, No. 1,2, pp83–101, 1976.

[16] Bose, N.K., "Problems and progress in multidimensional systems theory", Proceedings of I.E.E.E., Vol. 65, No. 6, pp824–840, 1977.

[17] Osgood, W.F., *Topics in the Theory of Functions of Several Complex Variables*, Dover publications, 1966.

[18] Walker, R.J., *Algebraic Curves*, Dover Publications, 1950.

[19] Fulton, W., *Algebraic Curves*, W.A. Benjamin: New York, Inc., 1969.

[20] Hodge, W.V.D., Pedoe, D., *Methods of Algebraic Geometry*, 3 Volumes, Cambridge University Press: Cambridge, England, 1947.

[21] Shafarevich, I.R., *Basic Algebraic Geometry*, Springer-Verlag: Berlin, 1974.

[22] Lefschetz, S. *Algebraic Geometry*, Princeton University Press: Princeton, N.J, 1963.

[23] Coolidge, J.L., *Algebraic Plane Curves*, Oxford, 1931.

[24] Hutchins, H.C., *Examples of Commutative Rings*, Polygonal Publishing House: N.J., 1981.

[25] Šebek, M., "One more counter-example in n-D systems theory–Unimodular matrices versus elementary operations", IEEE Transactions on Automatic Control, Vol. 33, No. 5, pp502–503, May 1988.

[26] Rosenbrock, H.H., Storey, C., *Mathematics of Dynamical Systems*, Nelson, 1970.

[27] Kučera, V. *Discrete Linear Control*, John Wiley & Sons: Prague, 1979.

[28] Šebek, M, "n-D polynomial matrix equations, IEEE Transactions on Automatic Control, Vol. 33, No. 5, pp499–502, May 1988.

[29] Šebek, M., "2-D Polynomial Equations", Kybernetica, Vol. 19, No. 3, 1983

[30] Kaczorek, T., "Algorithm for solving 2-D polynomial matrix equations", Bull. Pol. Acad. Sci., Vol. 31, pp51–57, 1983

[31] Kaczorek, T., "Polynomial matrix equations in two indeterminates", Bull. Pol. Acad. Sci., Vol. 30, pp39–44, 1982

[32] Kaczorek, T. *Two dimensional linear systems*, Springer-Verlag: Berlin, 1985

[33] Morf, M., Lévy, B.C., Kung, S-Y., "New results in 2-D system theory, Part I: 2-D polynomial matrices, factorisation and corpimeness", Proceedings of I.E.E.E., Vol. 65, No. 6, pp861–872, June 1977.

[34] Lévy, B.C., "2-D polynomial and rational matrices and theor applications for the modelling of 2-D dynamical systems", Technical Report Number M735-11, Ph.D. Thesis, Stanford Univeristy, 1981.

[35] J.P. Guiver and N.K. Bose, "Polynomial matrix primitive factorisation over arbitrary coefficient field and related results", IEEE Trans. Circuits and Systems, Vol. CAS-29, No. 10, Oct. 1982.

[36] Frost, M.G., Storey, C., "Equivalence of a matrix over $\mathbb{R}[s,z]$ with its Smith form", International Journal of Control, Vol. 28, No. 5, pp665–671, 1978.

[37] Frost, M.G., Storey, C., "Equivalence of matrices over $\mathbb{R}[s,z]$: Counter-example", International Journal of Control, Vol. 34, No. 6, pp1225–1226, 1981.

[38] Bose, N.K., "A criterion to determine if two multivariable polynomials are relatively prime", Proceedings of the I.E.E.E., Vol. 60, No. 1, pp134–135, Jan. 1972.

[39] Emre, E., Huseyin, O., "Relative primeness of multivariate polynomials", I.E.E.E. Transactions on Circuits and Systems, Vol. 22, p56, 1975.

[40] Youla, D.C., Gnavi, G., "Notes on n dimensional systems", IEEE Transactions on Circuits and Systems, Vol. CAS-26, No. 2, pp105–111, 1979.

[41] Quillen, D., "Projective modules over a polynomial ring", Invent. Math., Vol. 36, pp167–171, 1976

[42] Suslin, A.A., "Projective modules over a polynomial ring are free", Dokl. Adad. Nauk. S.S.S.R., Vol. 229, 1976. (English translation-Soviet Math. Dokl., MR 57, No. 9685, Vol. 17, pp1160–1164, 1976).

[43] Zak, S.H., Lee, E.B., "The simplified derivation of the completion theorem to a unimodular matrix over $\mathbb{R}[z_1,z_2]$", IEEE Transactions on Automatic Control, Vol. AC-30, No. 2, pp161-162. 1985

[44] Nobuyama, E., Shin, S., Okubo, S., Kitamori, T., "Bézout identities and common minors zeros of 2-D polynomial matrices and applications to time-delay systems", International Journal of Control, Vol. 52, No. 6, pp1311–1326, 1990.

[45] Lin, Z., "On Matrix Fraction Descriptions of Multivariable Linear n-D Systems", I.E.E.E. Transactions on Circuits and Systems, Vol. 35, No. 10, pp1317–1322, 1988.

[46] Pugh, A.C., Hayton, G.E., Walker, A.B., "System matrix characterisation of input-output equivalence", International Journal of Control, Vol. 51, No. 6, pp1319–1326, 1990.

[47] Fuhrmann, P.A., "On strict system equivalence and similarity", International Journal of Control, Vol. 25, No. 1, pp5–10, 1977.

[48] Morf, M., "Extended system and transfer function matrices and system equivalence definition", in Proceedings of the 1977 IEEE Conference on Decision and Control, New Orleans, pp795–800, 1977.

[49] Pernebo, L., "Notes on strict system equivalence", International Journal of Control, Vol. 25, No. 1, pp21–38, 1977.

[50] Pugh, A.C., Shelton, A.K. "On a new definition of strict system equivalence", International Journal of Control, Vol. 27, pp657–672, 1978.

[51] Rosenbrock, H.H., "Structural properties of linear dynamical systems", International Journal of Control, Vol. 20, pp191–202, 1974.

[52] Smith, M.C., "Matrix fractions and strict system equivalence", International Journal of Control, Vol. 34, pp869–883, 1981.

[53] Verghese, G.C., "Infinite frequency behaviour in generalised dynamical systems", Ph.D. dissertation, Stanford University, California, U.S.A., 1978

[54] Luenberger, D.G., "Time-invariant descriptor systems", Automatica, Vol. 14, No. 3, pp473–480, 1978.

[55] Bosgra, O.H., van der Weiden, A.J.J., "Realisations in generalised state-space form for polynomial system matrices and the definition of poles, zeros and decoupling zeros at infinity, International Journal of Control, Vol. 33, pp393–411, 1981

[56] Hayton, G.E., Walker, A.B., Pugh, A.C., "Matrix pencil equivalents of a general polynomial matrix", International Journal of Control, Vol. 49, pp1979–1987, 1989.

[57] Pugh, A.C., Ratcliff, "The infinite zeros of a rational matrix", Report CT5, Mathematics Department, Plymouth Polytechnic, Plymouth, U.K., 1979.

[58] Pugh, A.C., Hayton, G.E., Fretwell, P., "On transformations of matrix pencils and implications in linear systems theory", International Journal of Control, Vol. 45, pp529–548, 1987.

[59] Jones, E.R.LL., "The general pole placement problem in singular systems", Ph.D., Thesis, Loughborough University of Technology, Leicestershire, U.K., 1991.

[60] Hayton, G.E., Walker, A.B., Pugh, A.C., "A unification of system transformations", Proceedings ACC-88, Atlanta Georgia, Vol. 1, pp47-48, 1988

[61] Zhang, S-Y, "Generalised proper inverse of polynomial matrices and the existence of infinite decoupling zeros", IEEE Transactions on Automatic Control, AC-34, pp743–745. 1989

[62] Pugh, A.C., Ratcliffe, P.A., "On the zeros and poles of a rational matrix", International Journal of Control, Vol. 30, No. 2, pp213–226, 1979.

[63] Pugh, A.C., Ratcliffe, P.A., "Infinite frequency interpretations of minimal bases", International Journal of Control, Vol. 32, No. 4, pp581–588, 1980.

[64] Hayton, G.E., Pugh, A.C., Fretwell, P., "Infinite elementary divisors of a matrix polynomial and implications", International Journal of Control, Vol. 47, No. 1, pp53–64 , 1988.

[65] Pugh, A.C. Hayton, G.E., "Transformation issues in linear system theory", Proceedings MTNS-89, 1989.

[66] Pugh, A.C., "The McMillan degree of a polynomial system matrix", International Journal of Control, Vol. 24, No. 1, pp129–135, 1976.

[67] Pugh, A.C., Karampetakis, N.P., Vardularkis, A.I.G., Hayton, G.E., "A fundamental notion of equivalence for linear multivariable systems", Mathematical Sciences Report A153, Loughborough University of Technology, Leics, LE11 3TU, 1992.

[68] Pugh, A.C., "The occurrence of non-properness in closed-loop systems and some implications", Tzafestas, S.G., Multivariable Control, 43–63, Reidel Publishing Company, 1984.

[69] Pugh, A.C., Jones, E.R.LL., Demianczuk, O., Hayton, G.E., "Infinite frequency structure and a certain Laurent expansion", International Journal of Control, Vol. 50, pp1793–1805. 1989

[70] Anderson, B.D.O., Coppel, W.A., Cullen, D.J., "Strong system equivalence I", Journal of the Australian Mathematical Society, Ser. B27, pp194–222, 1985.

[71] Coppel, W.A., Cullen, D.J., "Strong system equivalence II", Journal of the Australian Mathematical Society, Ser. B27, pp223–237, 1985.

[72] Youla, D.C., Pickel, P.F., "The Quillen-Suslin theorem and structure of n-D elementary polynomial matrices", IEEE Transactions on Circuits and Systems, Vol. CAS-31, No. 6, pp513–517, 1984.

[73] Edwards, J.B., Owens, D.H., *Analysis and Control of Multipass Processes*, New York:Wiley.

[74] Rogers, E., Owens, D.H., "Modelling and stability analysis for a class of industrial repetitive processes", International Journal of Control, Vol. 52, No. 2, pp265–278, 1990.

[75] Owens, D.H., "Stability of multipass processes", Proceeding of the Institute of Electrical Engineers, Part D, Vol. 124, pp1079–1082, 1977.

[76] Rogers, E., Owens, D.H., "2-D transfer functions and stability tests for differential non-unit memory linear multipass processes", International Journal of Control, Vol. 50, No. 2, pp651–666, 1989.

[77] Solak, M.K., "A note on the Wolovich method of extraction of the GCD of two polynomial matrices", I.E.E.E. Transactions on Automatic Control, Vol. 30, No. 10, pp1032–1033, 1985.

[78] B.W. Char, K.O. Geddes, G.H. Gonnet, M.B. Monagan and S.M. Watt, *MAPLE Reference Manual*, 5th Edition, WATCOM Publications Limited, Waterloo, Canada, March 1988.

Index of Definitions

Index of Theorems

Index of Examples

Index of Algorithms and Lemmas

Index of Procedures

Publications

1. D.S. Johnson, A.C. Pugh and G.E. Hayton, "On the structure of polynomial models of 2-D systems", Proceedings of the Tenth IASTED International Symposium on Modelling, Identification and Control, Innsbruck Austria, February 1991, pp231–234. ISBN:0-88986-182-x.

2. D.S. Johnson, A.C. Pugh and G.E. Hayton, "Symbolic computation of the greatest common divisor of polynomial matrices in two indeterminates", Proceedings of the Eleventh IASTED International Conference on Modelling, Identification and Control, Innsbruck Austria, February 1992, pp124–125. ISBN:3-7153-0002-7.

3. D.S. Johnson, A.C. Pugh and G.E. Hayton, "On n-D matrix fraction descriptions", Proceedings ACC-'92, Vol. 1, pp357–358, June 1992.

4. A.C. Pugh, D.S. Johnson and G.E. Hayton, "On conditions guaranteeing two polynomial matrices possess identical zero structures", IEEE Transactions on Automatic Control, Vol. 37, No. 9, pp1383–6, September 1992.

5. A.C. Pugh, D.S. Johnson, G.E. Hayton, "2-D system structure and applications", IFAC Workshop on System Structure and Control, Prague, September 3-5, 1992.

6. D.S. Johnson, A.C. Pugh and G.E. Hayton, "Symbolic computation of the greatest common divisor of 2-D polynomial matrices", IMA C-MCI'92, Control: Modelling, Computation, Information, UMIST, England, September 2-4, 1992.