ARISTOTLE UNIVERSITY OF THESSALONIKI
DEPARTMENT OF MATHEMATICS
MASTER'S PROGRAMME
"THEORETICAL INFORMATICS & SYSTEMS AND CONTROL THEORY"

Specific Topics on Multivariable Systems and applications in Wolfram Mathematica

M.Sc. Thesis

Michailidis G.Michail

Thesis Supervisor: A.I.G. Vardulakis

Professor , Aristotle University of Thessaloniki

Thessaloniki , September 2014

ARISTOTLE UNIVERSITY OF THESSALONIKI
DEPARTMENT OF MATHEMATICS
MASTER'S PROGRAMME
"THEORETICAL INFORMATICS & SYSTEMS AND CONTROL THEORY"

Specific Topics on Multivariable Systems and applications in Wolfram Mathematica

M.Sc. Thesis

Michailidis G.Michail

Thesis Supervisor: A.I.G. Vardulakis

Professor , Aristotle University of Thessaloniki

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την

.....	.....	.....
Α.Ι.Γ. Βαρδουλάκης	Ν. Καραμπετάκης	Ε. Αντωνίου
Καθηγητής Α.Π.Θ.	Καθηγητής Α.Π.Θ.	Επικ. Καθηγητής ΑΤΕΙΘ

Thessaloniki , September 2014

ARISTOTLE UNIVERSITY OF THESSALONIKI
DEPARTMENT OF MATHEMATICS
MASTER'S PROGRAMME
"THEORETICAL INFORMATICS & SYSTEMS AND CONTROL THEORY"

Περιγραφή και Ανάλυση αλγορίθμων για πολυωνυμικούς πίνακες και Υλοποίηση μέσω Wolfram Mathematica

Μεταπτυχιακή Διπλωματική Εργασία

Μιχαηλίδης Γ. Μιχαήλ

Επιβλέπων : Α.Ι.Γ. Βαρδουλάκης

Καθηγητής , Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την

.....	.....	.....
Α.Ι.Γ. Βαρδουλάκης	Ν. Καραμπετάκης	Ε. Αντωνίου
Καθηγητής Α.Π.Θ.	Καθηγητής Α.Π.Θ.	Επικ. Καθηγητής ΑΤΕΙΘ

Thessaloniki , September 2014

……..

Μιχαηλίδης Γ. Μιχαήλ

Πτυχιούχος Μαθηματικός Α.Π.Θ.

Στην αξιολάτρευτη οικογένειά μου

Hey Brother , there is an endless road to (re)discover….

Avicii

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία επικεντρώνεται στον σχεδιασμό απλών δυναμικών γραφικών διασυνδέσεων χρήστη(user interface) που υλοποιούν κάποιους συγκεκριμένους αλγορίθμους για πολυωνυμικούς πίνακες , στο πεδίο της Θεωρίας Πινάκων ειδικότερα , που είναι άμεσα συνδεδεμένοι με το ευρύτερο πεδίο των Πολυμεταβλητών Συστημάτων.

Έχουμε δημιουργήσει μια καινούρια συνάρτηση στο προγραμματιστικό περιβάλλον του Mathematica για κάθε αλγόριθμο που αναλύουμε παρακάτω. Ένα στοιχειώδες user interface δέχεται τα δεδομένα του χρήστη , καλεί την αντίστοιχη συνάρτηση και έπειτα τυπώνει ένα πλαίσιο διαλόγου που περιέχει τα αποτελέσματα. Ο κύριος στόχος αυτής της εργασίας , όμως , είναι να σχεδιαστεί ένα πρόγραμμα το οποίο λύνει Διοφαντικές Εξισώσεις Πολυωνυμικών Πινάκων , προσδιορίζοντας παράλληλα την παραμετρική οικογένεια των κανονικών αντισταθμιστών που δίνουν κλειστά συστήματα με κάποιον επιθυμητό πίνακα παρονομαστή. Έτσι , οι παράγραφοι 3-9 μπορούν να θεωρηθούν ως ενδιάμεσα βήματα για την επίτευξη αυτού του στόχου. Η παράγραφος 11 καταδεικνύει μια αδυναμία εφαρμογής του αλγόριθμου αναγωγής πολυωνυμικού πίνακα σε κανονικό ως προς τις στήλες , ο οποίος οφείλεται στον Κύριο Βαρδουλάκη αλλά δεν δημοσιεύτηκε. Προτείνεται η χρήση του αντίστοιχου αλγορίθμου των **W.H.L. Neven** και **C.Praagman** , μια παραλλαγή του αρχικού αλγορίθμου του **Wolovich** , καθώς πιστεύουμε ότι λειτουργεί με πιο σαφή και εύληπτο τρόπο. Τέλος , στην παράγραφο 12 περιέχονται όλοι οι κώδικες που χρησιμοποιήθηκαν για την σχεδίαση των προγραμμάτων.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ

Διοφαντικές εξισώσεις πολυωνυμικών πινάκων ,γραμμικός πολυμεταβλητός έλεγχος , αναγωγή πίνακα σε κανονικό ως προς τις γραμμές – στήλες , διαίρεση πινάκων , Smith – McMillan μορφή , αριστερός-δεξιός μέγιστος κοινός διαιρέτης , αριστερή-δεξιά κλασματική περιγραφή, μοντελοποίηση μέσω Mathematica.

ABSTRACT

The current thesis focuses on the designation of dynamic user interfaces that implement some specific algorithms for polynomial matrices , Matrix Theory in general , which are closely related to the field of Multivariable Systems.

We have created a new , user – defined Mathematica function for each one of the algorithmic procedures that are illustrated . A dynamic user interface reads in the data inserted by the user , calls this function , and displays a dialog box containing the results. The major objective of this study , however , is to develop a program that solves polynomial matrix Diophantine equations and thus determines the class of proper compensators that yield closed – loop systems with a desired denominator. Therefore , sections 3 – 9 can be considered as intermediate steps in order to achieve this goal. Section 11 states an inaccuracy that was spotted about the column proper reduction algorithm informally developed by the Thesis Supervisor , Professor **Vardulakis**. In turn , we suggest the use of the column – proper algorithm by **W.H.L. Neven** and **C. Praagman** , a variation of the first algorithm originally developed by Wolovich , as we think it works in a more straightforward and direct fashion. In conclusion , section 12 contains all the Mathematica codes that were used to develop the interfaces and the programs.

KEYWORDS

Diophantine equations, linear multivariable control, row – column proper reduction, matrix division, Smith – McMillan form, left - right matrix gcd, left – right matrix fraction description, modeling via Mathematica.

TABLE OF CONTENTS

CHAPTER 1

DIVISION OF POLYNOMIAL MATRICES

Let us first consider the following sets :

$$\mathbb{R}(s)=\left\{t(s)\middle|t(s)=\frac{n(s)}{d(s)},n(s),d(s)\in\mathbb{R}[s],d(s)\neq 0\right\}$$

$$\mathbb{R}_{pr}(s)=\left\{t(s)\middle|t(s)=\frac{n(s)}{d(s)},n(s),d(s)\in\mathbb{R}[s],d(s)\neq 0,\deg d(s)\geq\deg n(s)\right\}$$

$$\mathbb{R}_{sp}(s)=\left\{t(s)\middle|t(s)=\frac{n(s)}{d(s)},n(s),d(s)\in\mathbb{R}[s],d(s)\neq 0,\deg d(s)>\deg n(s)\right\}$$

It is quite obvious that : $\mathbb{R}_{sp}(s)\subset\mathbb{R}_{pr}(s)\subset\mathbb{R}(s)$.

Theorem 1.1 , [1]

Let $t(s)\in\mathbb{R}(s)$. Then $t(s)$ is written uniquely in the form $t(s)=t_1(s)+q(s)$, where $t_1(s)\in\mathbb{R}_{sp}(s)$ and $q(s)\in\mathbb{R}[s]$.

Proof

Suppose that $t(s)=\frac{n(s)}{d(s)}$, $n(s),d(s)\in\mathbb{R}[s]$.

- If $t(s)\in\mathbb{R}[s]$, then $t_1(s)=0$ and $q(s)=t(s)$.
- If $t(s)\in\mathbb{R}_{sp}(s)$, then $t_1(s)=t(s)$ and $q(s)=0$.
- If $t(s)\in\mathbb{R}(s)$, that is if $\deg n(s)\geq\deg d(s)$, then by Euclid division there exist $q(s)\in\mathbb{R}[s]$, $r(s)\in\mathbb{R}[s]$ such that $n(s)=d(s)q(s)+r(s)$, where $\deg r(s)<\deg d(s)$.

This also implies that $t(s)=\frac{n(s)}{d(s)}=\frac{r(s)}{d(s)}+q(s)$ with $t_1(s)=\frac{r(s)}{d(s)}\in\mathbb{R}_{sp}(s)$.

▲

A generalization of the above theorem is basically the theory that we will use to divide two polynomial matrices.

Given two polynomial matrices $N(s)\in\mathbb{R}[s]^{p\times m}$, $D(s)\in\mathbb{R}[s]^{m\times m}$ we want to determine two polynomial matrices $Q(s),R(s)\in\mathbb{R}[s]^{p\times m}$ such that :

$$N(s)=Q(s)D(s)+R(s)\Rightarrow N(s)D(s)^{-1}=Q(s)+R(s)D(s)^{-1} \quad (1.1)$$

, where $R(s)D(s)^{-1}\in\mathbb{R}_{sp}(s)^{p\times m}$ and $Q(s)\in\mathbb{R}[s]^{p\times m}$. The matrices Q,R are called **right quotient** and **right remainder** respectively . Moreover , the method that will be described is the **right division** of the polynomial matrices N,D.

Algorithm for the right division , [1]

Let $T(s)=N(s)D(s)^{-1}\in\mathbb{R}(s)^{p\times m}$ and $t_{ij}(s)=\dfrac{n_{ij}(s)}{d_{ij}(s)}\in\mathbb{R}(s)$ the elements of T.

Using Theorem 1.1 we write each t_{ij} in the form : $t_{ij}(s)=t'_{ij}(s)+q_{ij}(s)$, where $t_{ij}'\in\mathbb{R}_{sp}(s)$ and $q_{ij}\in\mathbb{R}[s]$. This also means that $T(s)=T_{sp}(s)+T_{pol}(s)$, with $T_{sp}(s)\in\mathbb{R}_{sp}(s)^{p\times m}$ and $T_{pol}(s)\in\mathbb{R}[s]^{p\times m}$. The polynomial matrices Q,R that we seek are given by the following formulas :

- $Q(s)=T_{pol}(s)$ (1.2)
- $R(s)=N(s)-T_{pol}(s)D(s)=T_{sp}(s)D(s)$ (1.3)

Proof

It holds that $T(s)=N(s)D(s)^{-1}=T_{sp}(s)+T_{pol}(s)$. The right multiplication of the previous relation with $D(s)$ results in : $T(s)D(s)=N(s)=T_{sp}(s)D(s)+T_{pol}(s)D(s)$

It follows that $T_{sp}(s)D(s)=N(s)-T_{pol}(s)D(s)=R(s)$.Finally , we have :

$$R(s)D(s)^{-1}=\left[N(s)-T_{pol}(s)D(s)\right]D(s)^{-1}=N(s)D(s)^{-1}-T_{pol}(s)=T(s)-T_{pol}(s)=T_{sp}(s)$$

▲

Remark 1.1

- The matrix $D(s)$, also referred to as **denominator matrix** , has to be square and invertible($Det\left[D(s)\right] \neq 0$) . Its dimension must also be equal to the number of columns of the **numerator matrix ,** $N(s)$.

Following the same procedure with a few minor adjustments , we can use the **left division** algorithm for two polynomial matrices $N(s), D(s)$.

Brief Explanation of the algorithm used in Mathematica

- Firstly , we ask the user to insert the numerator matrix and the denominator matrix using the command **Input .** We then save the dimensions of each matrix using the command **Dimensions.**
- The algorithm then checks if either the algorithm for the right division or the algorithm for the left division can be applied using an **If** command , and then checks if the denominator matrix is invertible. If not, it displays an appropriate message to the user. It then proceeds to the main part of the division algorithm , which is the analysis of the matrix T , and the calculation of the matrices $T_{sp}(s), T_{pol}(s)$. We remark that if the denominator matrix is invertible and if the numerator matrix is a square one , then both of the algorithms can be applied.
- Afterwards , we calculate the matrix T and the matrices $T_{sp}(s), T_{pol}(s)$ by using three **If** commands , which perform the analysis of each element of the matrix T as described above.
- Finally we print the quotient , the remainder, and the division formula of the given matrices.

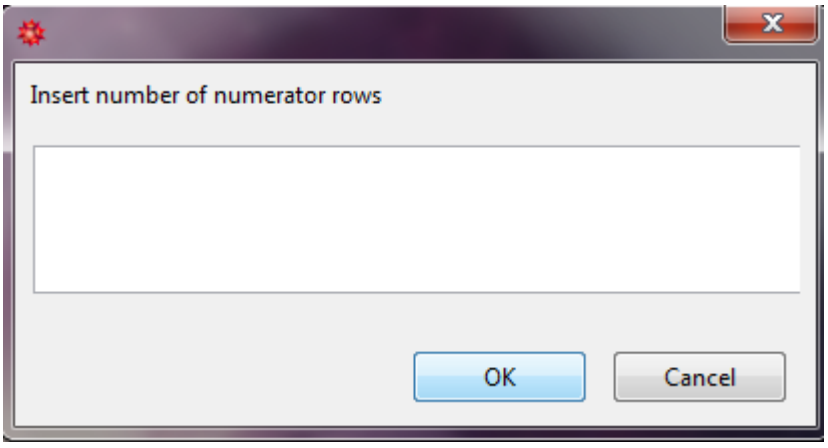
Commands and Syntax in Mathematica

- Input[*prompt*] $\rightarrow$ Requests input , displaying *prompt.*

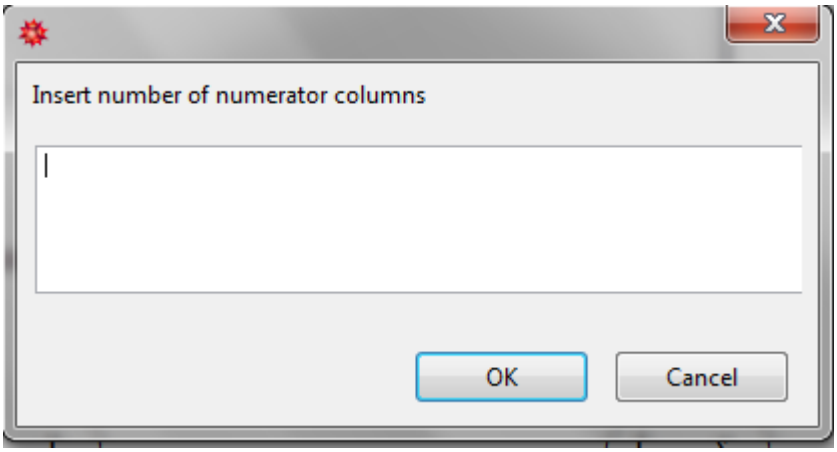
- For[*start , test , incr , body*] $\rightarrow$Executes *start ,* then repeatedly evaluates *body* and *incr* until *test* fails to become a true expression.
- If[*condition , t*] $\rightarrow$ Returns *t* if *condition* is a true expression.
- Dimensions[x] $\rightarrow$ Returns a list of the dimensions of the matrix x . We save the dimensions of the matrices $N(s), D(s)$ using the variables **dim** and **dimen** . For instance , **dim[[1]]** represents the number of rows of the numerator matrix.
- Det[x] $\rightarrow$ Returns the determinant of the matrix x.
- Exponent[*expr,form*] $\rightarrow$ Gives the maximum power with which *form* appears in the expanded form of *expr .*
- Numerator[*expr*] $\rightarrow$ Gives the numerator of the *expr .*
- Denominator[*expr*] $\rightarrow$ Gives the denominator of the *expr.*
- ConstantArray[0, {p,m}] $\rightarrow$ Generates a $p \times m$ matrix containing copies of the element 0.
- PolynomialQuotient[p,q,x] $\rightarrow$ Gives the quotient of p and q , treated as polynomials in x .
- PolynomialRemainder[p,q,x] $\rightarrow$ Gives the remainder of p and q , treated as polynomials in x.
- m[[i,j]] $\rightarrow$ Denotes the m_{ij} element of a matrix m.
- Exponent[0,x] $\rightarrow$ Mathematica returns $-\infty$. We mention this fact because in several cases we check whether the exponent of a specific element is equal to $-\infty$.
- Together[*expr*] $\rightarrow$ Puts terms in a sum over a common denominator. After the simplifications during the calculation of the matrix T , it may happen that an element is equal to a sum of terms. This will cause the **Denominator** command to malfunction regarding this specific element. For example , the command **Denominator[**$1+\frac{1}{s^2}$**]** will give 1 as an output. This is not the desired result and that is why we use the command **Together**.

Examples of running the program

The user interface that allows the user to insert data dynamically is :

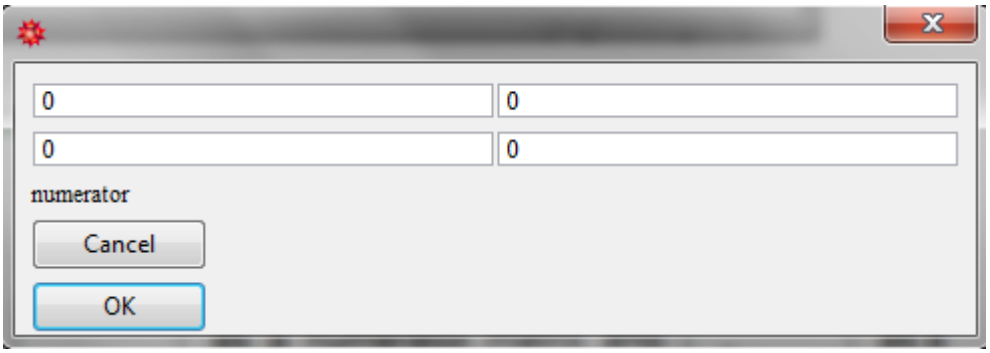


(Figure 1.1)



(Figure 1.2)

For instance , if the user inserts 2 and 2 respectively , then :



(Figure 1.3)

And the same applies to the denominator matrix. This is the general pattern that we use to create the dynamic user interface for all the algorithms that follow. So , we won't mention it again in the subsequent chapters.

When the user presses the button ok , an appropriate result is displayed by calling the respective function.

Let us now demonstrate the use of the created Mathematica function.

If we insert $\begin{pmatrix} s^2 & 1 \\ s & s+1 \end{pmatrix}$ as a numerator matrix and $\begin{pmatrix} 1 & s \\ s^3 & s-1 \end{pmatrix}$ as a denominator matrix , then the program yields :

```
MatrixDivision[{{s^2, 1}, {s, s + 1}}, {{1, s}, {s^3, s - 1}}]
```

The right quotient of the division is the matrix $\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$

The right remainder of the division is the matrix $\begin{pmatrix} s^2 & 1 \\ -1+s & 1 \end{pmatrix}$

The right division formula for the given matrices is : $\begin{pmatrix} s^2 & 1 \\ s & 1+s \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & s \\ s^3 & -1+s \end{pmatrix} + \begin{pmatrix} s^2 & 1 \\ -1+s & 1 \end{pmatrix}$

The left quotient of the division is the matrix $\begin{pmatrix} 0 & 0 \\ s & 0 \end{pmatrix}$

The left remainder of the division is the matrix $\begin{pmatrix} 0 & 1 \\ -(-2+s)\,s & 1+s \end{pmatrix}$

The left division formula for the given matrices is : $\begin{pmatrix} s^2 & 1 \\ s & 1+s \end{pmatrix} = \begin{pmatrix} 1 & s \\ s^3 & -1+s \end{pmatrix} \begin{pmatrix} 0 & 0 \\ s & 0 \end{pmatrix} + \begin{pmatrix} 0 & 1 \\ -(-2+s)\,s & 1+s \end{pmatrix}$

(Figure 1.4)

If we insert $\begin{pmatrix} s^2 & s \\ 1 & s-1 \\ s^2-2s & s+1 \end{pmatrix}$ as a numerator matrix and $\begin{pmatrix} s & s-1 \\ s^2 & s^3-1 \end{pmatrix}$ as a denominator matrix the outcome will be :

```
MatrixDivision[{{s^2, s}, {1, s - 1}, {s^2 - 2 s, s + 1}}, {{s, s - 1}, {s^2, s^3 - 1}}]
```

The right quotient of the division is the matrix $\begin{pmatrix} 1+s & 0 \\ 0 & 0 \\ -1+s & 0 \end{pmatrix}$

The right remainder of the division is the matrix $\begin{pmatrix} -s & 1+s-s^2 \\ 1 & -1+s \\ -s & -(-3+s)\,s \end{pmatrix}$

The right division formula for the given matrices is : $\begin{pmatrix} s^2 & s \\ 1 & -1+s \\ -2s+s^2 & 1+s \end{pmatrix} = \begin{pmatrix} 1+s & 0 \\ 0 & 0 \\ -1+s & 0 \end{pmatrix} \begin{pmatrix} s & -1+s \\ s^2 & -1+s^3 \end{pmatrix} + \begin{pmatrix} -s & 1+s-s^2 \\ 1 & -1+s \\ -s & -(-3+s)\,s \end{pmatrix}$

(Figure 1.5)

If the user inserts a denominator matrix that is not invertible , then the program will display the following message :

"*The determinant of the denominator matrix must not be equal to zero. Please insert another matrix."*

Finally , if the user inserts a denominator matrix that is not a square one , or if the dimensions of the denominator matrix are not suitable for the execution of the division algorithm , then we print the message :

"The denominator matrix needs to be a square matrix with its dimension equal to the number of rows or the number of columns of the numerator matrix. "

CHAPTER 2

SMITH – MCMILLAN FORM OF A PROPER RATIONAL MATRIX

Our first goal is to describe the steps required to find the **Smith – Form** of a polynomial matrix $T(s)\in\mathbb{R}[s]^{p\times m}$ with $rank_{\mathbb{R}(s)}T(s)=r\leq\min\{p,m\}$.

Let $m_i(s)\in\mathbb{R}[s]$, $i=1,2,\ldots r$ be the greatest common divisor of all the $i\times i$ minors of the matrix $T(s)$.

We also consider the following polynomials :

$$\varepsilon_i(s)=\frac{m_i(s)}{m_{i-1}(s)}\ ,\ i=1,2,\ldots r \text{ with } m_0(s)=1.$$

Then the **Smith – Form** of the polynomial matrix $T(s)$ is the $p\times m$ matrix :

$$S^{\mathbb{C}}_{T(s)}=\begin{pmatrix}\varepsilon_1(s) & 0 & \ldots & 0 & 0 & \ldots & 0\\ 0 & \varepsilon_2(s) & \ldots & 0 & 0 & \ldots & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots\\ 0 & 0 & \ldots & \varepsilon_r(s) & 0 & \ldots & 0\\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0\\ \vdots & \vdots & & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0\end{pmatrix}$$

The polynomials $\varepsilon_i(s)$ are called **invariant polynomials** and satisfy $\varepsilon_i\,|\,\varepsilon_{i+1},\forall 1\leq i<r$.

There is also another way to obtain the **Smith – Form** of a polynomial matrix , but we will give a detailed description in the next chapter.

Now , suppose that the matrix $T(s)\in\mathbb{R}(s)^{p\times m}$ is not necessarily a polynomial one, in other words $T(s)$ is a proper rational matrix.

If $d(s)$ is the least common multiple of the denominators of the elements $t_{ij}(s)=\frac{n_{ij}(s)}{d_{ij}(s)}$

$i=1,2,\dots p$, $j=1,2,\dots m$ of $T(s)$, then $T(s)$ is written in the form :

$T(s)=\frac{1}{d(s)}N(s)$, $N(s)\in\mathbb{R}[s]^{p\times m}$. Let's suppose that the Smith – Form of the matrix $N(s)$ is the following :

$$S_{N(s)}^{\mathbb{C}}=\begin{pmatrix} n_1(s) & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & n_2(s) & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & n_r(s) & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \end{pmatrix}$$

Now let $\frac{n_i(s)}{d(s)}=\frac{\varepsilon_i(s)}{\psi_i(s)}$, $i=1,2,\dots r$.

The **Smith – McMillan** Form of the matrix $T(s)$ is the matrix :

$$S_{T(s)}^{\mathbb{C}}=\begin{pmatrix} \frac{\varepsilon_1(s)}{\psi_1(s)} & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \frac{\varepsilon_2(s)}{\psi_2(s)} & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \frac{\varepsilon_r(s)}{\psi_r(s)} & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \end{pmatrix}\in\mathbb{R}(s)^{p\times m}$$

The polynomials ε_i,ψ_i satisfy :

- $\varepsilon_i\,|\,\varepsilon_{i+1},\forall 1\le i<r$
- $\psi_{i+1}\,|\,\psi_i,\forall 1\le i<r$

Brief Explanation of the algorithm used in Mathematica

- We ask the user to insert a proper rational matrix and we save the dimensions and the rank .
- Using the variables **d , i** we save the invariant polynomials of $N(s)$ and we create a matrix **smith** which will be the Smith form of the matrix $N(s)$ that we want to determine.
- We enter a loop to find the least common multiple of the denominators of the matrix $T(s)$ which is saved using the variable **lcm ,** and then we calculate the matrix $N(s)$, represented by the variable **b .**
- For $k=1,2,\ldots r$ we save the $k\times k$ minors of $N(s)$ using the matrix **mat** and then we calculate the greatest common divisor of all the elements of the matrix **mat .** This divisor is saved in the element d[[1,k]].
- For $k=1,2,\ldots r$ we can now find the invariant polynomials of $N(s)$, that are saved in the element i[[1,k]]. The Smith Form of the matrix $N(s)$ can now be easily calculated.
- By multiplying the matrix $N(s)$ with $\frac{1}{least common multiple}$, we find the Smith – McMillan form of the given matrix $T(s)$.
- We print the final result , displaying an appropriate message to the user.

Commands and Syntax in Mathematica

- MatrixRank[x] $\rightarrow$ Gives the rank of the matrix x.
- PolynomialLCM[$x_1, x_2, \ldots$] $\rightarrow$ Displays the least common multiple of the polynomials x_i
- Minors[m,k] $\rightarrow$ Gives the k^{th} minors of m.
- PolynomialGCD[$x_1, x_2, \ldots$] $\rightarrow$ Displays the greatest common divisor of the polynomials x_i
- Factor[x] $\rightarrow$ Factors a polynomial x over the integers .

Examples of running the program

- Square Matrices

If the matrix $\begin{pmatrix} \frac{1}{s+1} & 0 \\ \frac{1}{s+2} & \frac{1}{s+3} \end{pmatrix}$ is inserted , then the program yields the following **Smith – McMillan** form :

```
SmithMcMillan[{{1 / (s + 1), 0}, {1 / (s + 2), 1 / (s + 3)}}]
```

```
The Smith-McMillan form of the matrix {{1/(1+s), 0}, {1/(2+s), 1/(3+s)}} is the matrix {{1/((1+s) (2+s) (3+s)), 0}, {0, 2 + s}}
```

(Figure 2.1)

- Random Matrices

If the matrix $\begin{pmatrix} \frac{1}{s+1} & 0 & \frac{s+3}{(s+1)(s+2)} \\ \frac{1}{s+2} & \frac{1}{s+3} & \frac{s+3}{(s+2)^2} \end{pmatrix}$ is inserted , then the program prints the following **Smith – McMillan** form :

```
SmithMcMillan[{{1 / (s + 1), 0, (s + 3) / ((s + 1) (s + 2))}, {1 / (s + 2), 1 / (s + 3), (s + 3) / (s + 2)^2}}]
```

```
The Smith-McMillan form of the matrix {{1/(1+s), 0, (3+s)/((1+s) (2+s))}, {1/(2+s), 1/(3+s), (3+s)/(2+s)^2}} is the matrix {{1/((1+s) (2+s)^2 (3+s)), 0, 0}, {0, 2 + s, 0}}
```

(Figure 2.2)

- Polynomial Matrices

If the matrix $\begin{pmatrix} s^3-1 & s & s-1 \\ s^2-1 & s^2-2s & s+2 \\ s & s & s-4 \end{pmatrix}$ is inserted , then we obtain the **Smith – McMillan** form :

```
SmithMcMillan[{{s^3 - 1, s, s - 1}, {s^2 - 1, s^2 - 2 s, s + 2}, {s, s, s - 4}}]
```

```
The Smith-McMillan form of the matrix {{-1 + s^3, s, -1 + s}, {-1 + s^2, -2 s + s^2, 2 + s}, {s, s, -4 + s}} is the matrix {{1, 0, 0}, {0, 1, 0}, {0, 0, s (-9 + 7 s + 6 s^2 + 5 s^3 - 7 s^4 + s^5)}}
```

(Figure 2.3)

CHAPTER 3

SMITH DECOMPOSITION – SMITH FORM OF A POLYNOMIAL MATRIX USING ELEMENTARY OPERATIONS

As we mentioned in the previous section , there is also another way to obtain the **Smith – Form** of a polynomial matrix . This method will be analyzed in the current chapter. Suppose that we have a polynomial matrix $T(s) \in \mathbb{R}[s]^{p\times m}$.

We define the elementary operations that can be performed on $T(s)$.

1. Row or column switching. A row or column within the matrix can be switched with another row or column. $\left(R_i \leftrightarrow R_j, C_i \leftrightarrow C_j\right)$
2. Row or column multiplication . Each element in a row or column can be multiplied by a non – zero constant. $\left(R_i \to kR_i, C_i \to kC_i, k \neq 0\right)$
3. Row or column addition . A row or column can be replaced by the sum of that row or column and a multiple of another row or column. $\left(R_i \to R_i + kR_j, C_i \to C_i + kC_j, i \neq j\right)$

We now present the algorithm , [1] & [3], that can be applied to the matrix $T(s)$, in order to put this particular matrix into Smith – Form.

1. We locate an element $t_{ij}(s)$ of the matrix $T(s)$ that has the minimum degree $\left(t_{ij}(s) \neq 0\right)$, and using row and column switching we transfer it to the $(1,1)$ position .
2. We consider the Euclid division of the element $t_{12}(s)$ with $t_{11}(s)$, so that $t_{12}(s) = t_{11}(s)q_{12}(s) + r_{12}(s)$. If $r_{12}(s) \neq 0$, we subtract the 1st column multiplied by $q_{12}(s)$ from the 2nd and thus we replace the element $t_{12}(s)$ with the element $r_{12}(s)$. The point is that $\deg r_{12}(s) < \deg t_{12}(s)$. After that , we switch the columns 1 and 2 , thus reducing the degree of the $(1,1)$ element , and we repeat the previous procedure till $r_{12}(s) = 0$.

3. For $k=2,3,\ldots m$ we repeat step 2 for all the elements $t_{1k}(s)$ of the 1st row of $T(s)$.
4. When steps 2 and 3 are finished we subtract the 1st column multiplied by $q_{1k}(s)$ from the k^{th} column , for $k=2,3,\ldots m$. In this way , we transform the initial matrix $T(s)$ into $\begin{pmatrix} t_{11}(s) & 0 & \ldots & 0 \\ t_{21}(s) & t_{22}(s) & \ldots & t_{2m}(s) \\ \vdots & \vdots & \ddots & \vdots \\ t_{p1}(s) & t_{p2}(s) & \ldots & t_{pm}(s) \end{pmatrix}$.
5. We consider the Euclid division of the element $t_{21}(s)$ with $t_{11}(s)$, so that $t_{21}(s)=t_{11}(s)q_{21}(s)+r_{21}(s)$. If $r_{21}(s)\neq 0$, we subtract the 1st row multiplied by $q_{21}(s)$ from the 2nd and thus we replace the element $t_{21}(s)$ with the element $r_{21}(s)$. The point is that $\deg r_{21}(s)<\deg t_{21}(s)$. After that , we switch the rows 1 and 2 , thus reducing the degree of the $(1,1)$ element , and we repeat the previous procedure till $r_{21}(s)=0$.
6. For $k=2,3,\ldots p$ we repeat step 5 for all the elements $t_{k1}(s)$ of the 1st column of $T(s)$.
7. When steps 5 and 6 are finished we subtract the 1st row multiplied by $q_{k1}(s)$ from the k^{th} row , for $k=2,3,\ldots p$. In this way , we transform the matrix $T(s)$ into $\begin{pmatrix} t_{11}(s) & 0 & \ldots & 0 \\ 0 & t_{22}(s) & \ldots & t_{2m}(s) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & t_{p2}(s) & \ldots & t_{pm}(s) \end{pmatrix}$.
8. If $t_{11}(s)$ divides $(remainder\equiv 0)$, all the elements of the matrix that is generated if we eliminate the 1st row and the 1st column of $T(s)$, then we proceed to step 9. Otherwise , if $t_{xy}(s)$, $x=2,3,\ldots p$, $y=2,3,\ldots m$ is such an element , we add the y^{th} column to the first one and we return to step 2.
9. The element $t_{11}(s)$ is the first of the invariant polynomials. To determine the rest of them and to obtain the Smith – Form we repeat the previous steps for the submatrix :

$$\begin{pmatrix} t_{22}(s) & \dots & t_{2m}(s) \\ \vdots & \ddots & \vdots \\ t_{p2}(s) & \dots & t_{pm}(s) \end{pmatrix}.$$

10. Finally , if $S^{\mathbb{C}}_{T(s)} = \begin{pmatrix} \varepsilon_1(s) & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \varepsilon_2(s) & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \varepsilon_r(s) & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \end{pmatrix}$ is the Smith – Form then we can multiply each row with an appropriate number in order to transform the invariant polynomials $\varepsilon_i(s)$ into polynomials whose leading coefficient (the non zero coefficient of highest degree) is equal to 1.

The Smith Form can also be obtained from the original matrix by multiplying on the left and right by unimodular ($T(s)$ unimodular $\Leftrightarrow Det\left[T(s)\right] = c \in \mathbb{R}, c \neq 0$) square matrices $T_L(s) \in \mathbb{R}[s]^{p\times p}$, $T_R(s) \in \mathbb{R}[s]^{m\times m}$ so that :

$$S^{\mathbb{C}}_{T(s)} = T_L(s)T(s)T_R(s) \qquad (3.1)$$

The matrices $T_L(s), T_R(s)$ can be found by starting out with identity matrices of the appropriate size , and modifying $T_L(s)$ each time a row operation is performed on $T(s)$ in the algorithm by the same row operation , and similarly modifying $T_R(s)$ for each column operation performed.

We will give an example to indicate how the algorithm works.

Suppose that we have the polynomial matrix : $\begin{pmatrix} s+1 & s-1 & s^2-1 \\ s^2-2s & s^2 & s^2+2s \\ s & s & s+1 \end{pmatrix}$.

$$\begin{pmatrix} s+1 & s-1 & s^2-1 \\ s^2-2s & s^2 & s^2+2s \\ s & s & s+1 \end{pmatrix} \underset{\rightarrow}{R_1 \leftrightarrow R_3} \begin{pmatrix} s & s & s+1 \\ s^2-2s & s^2 & s^2+2s \\ s+1 & s-1 & s^2-1 \end{pmatrix} \underset{\rightarrow}{C_3 \rightarrow C_3 - 1\cdot C_1} \begin{pmatrix} s & s & 1 \\ s^2-2s & s^2 & 4s \\ s+1 & s-1 & s^2-s-2 \end{pmatrix}$$

$$C_1 \leftrightarrow C_3 \to \begin{pmatrix} 1 & s & s \\ 4s & s^2 & s^2-2s \\ s^2-s-2 & s-1 & s+1 \end{pmatrix} \begin{matrix} C_2 \to C_2 - s\cdot C_1 \\ \to \\ C_3 \to C_3 - s\cdot C_1 \end{matrix} \begin{pmatrix} 1 & 0 & 0 \\ 4s & -3s^2 & -s(2+3s) \\ s^2-s-2 & -s^3+s^2+3s-1 & -s^3+s^2+3s+1 \end{pmatrix}$$

$$\begin{matrix} R_2 \to R_2 - 4s\cdot R_1 \\ \to \\ R_3 \to R_3 - (s^2-s-2)\cdot R_1 \end{matrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -3s^2 & -s(2+3s) \\ 0 & -s^3+s^2+3s-1 & -s^3+s^2+3s+1 \end{pmatrix} \begin{matrix} C_3 \to C_3 - 1\cdot C_2 \\ \to \end{matrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -3s^2 & -2s \\ 0 & -s^3+s^2+3s-1 & 2 \end{pmatrix}$$

$$C_3 \leftrightarrow C_2 \to \begin{pmatrix} 1 & 0 & 0 \\ 0 & -2s & -3s^2 \\ 0 & 2 & -s^3+s^2+3s-1 \end{pmatrix} C_3 \to C_3 - \frac{3s}{2}\cdot C_2 \to \begin{pmatrix} 1 & 0 & 0 \\ 0 & -2s & 0 \\ 0 & 2 & -s^3+s^2-1 \end{pmatrix} R_2 \leftrightarrow R_3 \to \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & -s^3+s^2-1 \\ 0 & -2s & 0 \end{pmatrix}$$

$$C_3 \to C_3 - \left(\frac{-s^3+s^2-1}{2}\right)\cdot C_2 \to \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & -2s & -s+s^3-s^4 \end{pmatrix} R_3 \to R_3 - (-s)\cdot R_2 \to \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -s+s^3-s^4 \end{pmatrix} \begin{matrix} R_2 \to \frac{1}{2}\cdot R_2 \\ \to \\ R_3 \to (-1)\cdot R_3 \end{matrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s-s^3+s^4 \end{pmatrix}$$

The final matrix is the Smith – Form of the matrix that was given.

<u>Brief Explanation of the algorithm used in Mathematica</u>

- We ask the user to insert a polynomial matrix and we save its dimensions.
- The arrays **aleft , aright** are created to save the row and column operations respectively , which will be used to form the matrices $T_L(s), T_R(s)$. The variables **row , column** count the number of each operation performed on $T(s)$.
- The first loop `While[condition < 1, ⋮ ];` includes the steps 1 – 9 of the algorithm.
- We then use another loop to scan the matrix $T(s)$ for an element that has the minimum degree , and we save the position of it using the variables **x ,y.**
- Mathematica has no fixed way of performing row or column operations. To select the i^{th} row of a matrix a we can type : $a[[i]]$. Similarly , to select the i^{th} column we type : $a[[All, i]]$. We can now switch rows i, j by typing : $help = a;$ $a[[i]] = help[[j]];$. If we want to add the row j to the row i we can type : $a[[j]] = help[[i]];$

$help = a;$
$a[[i]] = help[[i]] + help[[j]];$. Using this pattern we can perform row or column operations.

- We now transfer the element (x, y) to the $(1,1)$ position of the matrix $T(s)$, thus finishing step 1 of the algorithm. We can notice that every time a row or column operation is performed , this operation is saved using **aleft** or **aright** accordingly.

$While[\ firstcondition < 1,$
- The second loop $\vdots$ includes the steps 2 – 8. The loop $];$

is terminated if the $(1,1)$ element of $T(s)$ divides all of the elements of the matrix that is generated if we eliminate the 1st row and the 1st column of $T(s)$.

- In this loop we execute the steps 2 – 7. The first two $While$ loops execute steps 2 and 3.
- Then comes a For loop that is used to perform step 4.
- Similarly , we use another two $While$ loops to execute steps 5 and 6.
- And a For loop that refers to step 7.
- The variable **zerocount** is used to count the number of the elements of the $(p-1)\times(m-1)$ submatrix whose remainder is equal to 0. If $zerocount = (p-1)(m-1)$, then the loop including steps 2 – 8 is terminated. Otherwise , we act according to step 8 and the loop continues.
- At the start of the program we entered the variable **count** . This variable counts the number of times that we moved on to the submatrix as described in step 9.
- We pick out the submatrix using the command **Take** and we consider this submatrix as our new matrix $T(s)$. We can now explain why the **count** variable is useful. That is because the new matrix $T(s)$ doesn't have the same dimensions with the initial one. For instance , if we need to switch rows $1,2$ of the submatrix , this actually means that we need to switch rows $2,3$ of the initial matrix that was given. This error is countered with the use of this variable.
- Finally , we need a condition that will terminate the initial loop and end steps 1 – 9.We will present an example of how the condition works. Suppose that we

have a 3×5 matrix. After consecutive eliminations we will finally be left with a row matrix 1×3. We repeat steps 2,3,4 to obtain the desired form and then we terminate the loop. The same applies to a 5×3 matrix. In this case , we will be left with a column matrix 3×1. We repeat steps 5,6,7 and we terminate the loop. If the initial matrix is a square one , then we simply terminate the loop without any further actions.

- We determine the matrices $T_L(s), T_R(s)$ by calculating a product of the matrices that were previously saved in the arrays **aleft , aright ,** and we take the Smith – Form of the matrix $T(s)$.
- The final loop is used to execute step 10 of the algorithm.
- We print the Smith – Form and we display an appropriate message to the user.

Commands and Syntax in Mathematica

- Array[f,n] $\rightarrow$ Generates a list of length n with elements $f[i]$
- While[*test , body*] $\rightarrow$ Evaluates *test* , then *body,* repetitively , until *test* first fails to become a true expression.
- Coefficient[*expr , form , n*] $\rightarrow$ Gives the coefficient of *form^n* in *expr*.

Examples of running the program

If the matrix $T(s)=\begin{pmatrix} s+1 & s-1 & s^2-1 \\ s^2-2s & s^2 & s^2+2s \\ s & s & s+1 \end{pmatrix}$ is inserted then the program yields :

```
SmithDecomposition[{{s+1, s-1, s^2-1}, {s^2-2s, s^2, s^2+2s}, {s, s, s+1}}]
```

The Smith form of the polynomial matrix $\begin{pmatrix} 1+s & -1+s & -1+s^2 \\ -2s+s^2 & s^2 & 2s+s^2 \\ s & s & 1+s \end{pmatrix}$ is the matrix $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s-s^3+s^4 \end{pmatrix} = \begin{pmatrix} -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 \\ -s & -1 & s\left(2-s+s^2\right) \end{pmatrix} \begin{pmatrix} 1+s & -1+s & -1+s^2 \\ -2s+s^2 & s^2 & 2s+s^2 \\ s & s & 1+s \end{pmatrix} \begin{pmatrix} -1 & -\frac{1}{2}(-1+s)s & \frac{1}{2}(-1+s)^2(1+s) \\ 1 & \frac{1}{2}\left(-2-s+s^2\right) & \frac{1}{2}\left(1+3s+s^2-s^3\right) \\ 0 & 1 & -s \end{pmatrix}$

```
Det[rightunimodular]
```

```
1
```

```
Det[leftunimodular]
```

$$-\frac{1}{2}$$

(Figure 3.1)

If we insert the matrix $T(s)=\begin{pmatrix} s^3-1 & s & s-1 \\ s^2-1 & s^2-2s & s+2 \\ s & s & s-4 \end{pmatrix}$ then the outcomes will be :

```
SmithDecomposition[{{s^3 - 1, s, s - 1}, {s^2 - 1, s^2 - 2 s, s + 2}, {s, s, s - 4}}]
```

The Smith form of the polynomial matrix $\begin{pmatrix} -1+s^3 & s & -1+s \\ -1+s^2 & -2s+s^2 & 2+s \\ s & s & -4+s \end{pmatrix}$ is the matrix $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s\left(-9+7s+6s^2+5s^3-7s^4+s^5\right) \end{pmatrix}$ =

$$\begin{pmatrix} -1 & 0 & 0 \\ \frac{1}{4}s\left(-1+s^2\right) & 0 & -\frac{1}{4} \\ \frac{1}{3}\left(-12+9s+21s^2+5s^3-26s^4+17s^5+4s^6-7s^7+s^8\right) & 4 & \frac{1}{3}\left(9+9s-10s^2-5s^3+7s^4-s^5\right) \end{pmatrix} \begin{pmatrix} -1+s^3 & s & -1+s \\ -1+s^2 & -2s+s^2 & 2+s \\ s & s & -4+s \end{pmatrix} \begin{pmatrix} 1 & -1 & \frac{3s}{4} \\ -s^2 & \frac{1}{3}\left(-4+s+4s^2-s^3\right) & \frac{1}{4}\left(4-s^2-4s^3+s^4\right) \\ 0 & \frac{1}{3}\left(3-s+s^3\right) & \frac{1}{4}\left(s+s^2-s^4\right) \end{pmatrix}$$

```
Det[rightunimodular]
```

```
-1
```

```
Det[leftunimodular]
```

```
-1
```

(Figure 3.2)

Finally , if we insert the matrix

$T(s)=\begin{pmatrix} (s+2)^2(s+3) & 0 & (s+2)(s+3)^2 \\ (s+1)(s+2)(s+3) & (s+1)(s+2)^2 & (s+1)(s+3)^2 \end{pmatrix}$ then :

```
SmithDecomposition[{{((s + 2) ^2) * (s + 3), 0, (s + 2) * (s + 3) ^2}, {(s + 1) (s + 2) (s + 3), (s + 1) (s + 2) ^2, (s + 1) (s + 3) ^2}}]
```

The Smith form of the polynomial matrix $\begin{pmatrix} (2+s)^2(3+s) & 0 & (2+s)(3+s)^2 \\ (1+s)(2+s)(3+s) & (1+s)(2+s)^2 & (1+s)(3+s)^2 \end{pmatrix}$ is the matrix $\begin{pmatrix} 1 & 0 & 0 \\ 0 & (2+s)^3\left(3+4s+s^2\right) & 0 \end{pmatrix}$

$$= \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ -10-22s-18s^2-7s^3-s^4 & 12+22s+18s^2+7s^3+s^4 \end{pmatrix} \begin{pmatrix} (2+s)^2(3+s) & 0 & (2+s)(3+s)^2 \\ (1+s)(2+s)(3+s) & (1+s)(2+s)^2 & (1+s)(3+s)^2 \end{pmatrix} \begin{pmatrix} -2-2s-s^2 & -\frac{1}{2}(1+s)(2+s)^2 & 3+s \\ 1 & \frac{3+s}{2} & 0 \\ 2+2s+s^2 & \frac{1}{2}(1+s)(2+s)^2 & -2-s \end{pmatrix}$$

```
Det[rightunimodular]
```

```
-1
```

```
Det[leftunimodular]
```

```
1
```

(Figure 3.3)

CHAPTER 4

GREATEST COMMON LEFT DIVISOR OF TWO POLYNOMIAL MATRICES

First we give some introductory definitions of the terms *divisor, multiple* and *greatest common divisor,* [1] & [3].

Suppose that we have three polynomial matrices $A(s) \in \mathbb{R}[s]^{p\times m}, B(s) \in \mathbb{R}[s]^{p\times q}, C(s) \in \mathbb{R}[s]^{q\times m}$ that satisfy : $A(s) = B(s)C(s)$.

We say that :

- $B(s)$ is a left divisor of $A(s)$
- $C(s)$ is a right divisor of $A(s)$
- $A(s)$ is a left multiple of $C(s)$
- $A(s)$ is a right multiple of $B(s)$

Suppose now that $T_1[s] \in \mathbb{R}[s]^{p\times l}$, $T_2[s] \in \mathbb{R}[s]^{p\times t}$ are two polynomial matrices with the same number of rows and let $T_L(s) \in \mathbb{R}[s]^{p\times p}$ be a left divisor of $T_1(s), T_2(s)$, that is :

$T_1(s) = T_L(s)\bar{T}_1(s)$ and $T_2(s) = T_L(s)\bar{T}_2(s)$, where $\bar{T}_1(s) \in \mathbb{R}[s]^{p\times l}$ and $\bar{T}_2(s) \in \mathbb{R}[s]^{p\times t}$.

Definition 4.1 , [1] & [3]

The matrix $T_L(s)$ is called common left divisor of the polynomial matrices $T_1(s), T_2(s)$.

If $T_L(s)$ is a right multiple of every common left divisor $\bar{T}_L(s)$, that is if $T_L(s)=\bar{T}_L(s)T_3(s)$ for some $T_3(s)\in\mathbb{R}[s]^{p\times p}$, then the matrix $T_L(s)$ is called greatest common left divisor of the polynomial matrices $T_1(s),T_2(s)$.

Given two polynomial matrices with the same number of rows $T_1[s]\in\mathbb{R}[s]^{p\times l}$, $T_2[s]\in\mathbb{R}[s]^{p\times t}$, we will present the algorithm that finds their greatest common divisor.

We form the matrix $T(s)=\begin{bmatrix}T_1(s) & T_2(s)\end{bmatrix}\in\mathbb{R}[s]^{p\times m}$ where $l+t=m\geq p=rank_{\mathbb{R}(s)}T(s)$.

By applying the algorithm described in the previous section, we determine a unimodular square matrix $T_R(s)\in\mathbb{R}[s]^{m\times m}$ such that :

$T(s)T_R(s)=\begin{bmatrix}T_{GL}(s) & 0_{p,m-p}\end{bmatrix}$, where $T_{GL}(s)\in\mathbb{R}[s]^{p\times p}$, $rank_{\mathbb{R}(s)}T_{GL}(s)=p$

We will now prove that the matrix $T_{GL}(s)$ is the greatest common divisor of the given matrices.

Proof

Let $T_R(s)^{-1}=\hat{T}(s)=\begin{bmatrix}\hat{T}_1(s) & \hat{T}_2(s)\\ \hat{T}_3(s) & \hat{T}_4(s)\end{bmatrix}$.

Then $T(s)T_R(s)=\begin{bmatrix}T_{GL}(s) & 0_{p,m-p}\end{bmatrix}$ gives :

$$T(s)=\begin{bmatrix}T_{GL}(s) & 0_{p,m-p}\end{bmatrix}T_R(s)^{-1}=T_{GL}(s)\begin{bmatrix}I_p & 0_{p,m-p}\end{bmatrix}\begin{bmatrix}\hat{T}_1(s) & \hat{T}_2(s)\\ \hat{T}_3(s) & \hat{T}_4(s)\end{bmatrix}=T_{GL}(s)\begin{bmatrix}\hat{T}_1(s) & \hat{T}_2(s)\end{bmatrix}$$

This in turn gives :

$T_1(s)=T_{GL}(s)\hat{T}_1(s)$ and $T_2(s)=T_{GL}(s)\hat{T}_2(s)$, which means that $T_{GL}(s)$ is a common left divisor of the matrices $T_1(s),T_2(s)$.

Now let $T_R(s)=\begin{bmatrix} T_{R1}(s) & T_{R2}(s) \\ T_{R3}(s) & T_{R4}(s) \end{bmatrix}$.

Then $T(s)T_R(s)=\begin{bmatrix} T_{GL}(s) & 0_{p,m-p} \end{bmatrix}$ gives :

$$T(s)\begin{bmatrix} T_{R1}(s) & T_{R2}(s) \\ T_{R3}(s) & T_{R4}(s) \end{bmatrix}=\begin{bmatrix} T_{GL}(s) & 0_{p,m-p} \end{bmatrix}\Rightarrow$$

$$\begin{bmatrix} T_1(s) & T_2(s) \end{bmatrix}\begin{bmatrix} T_{R1}(s) & T_{R2}(s) \\ T_{R3}(s) & T_{R4}(s) \end{bmatrix}=\begin{bmatrix} T_{GL}(s) & 0_{p,m-p} \end{bmatrix}\Rightarrow$$

$$T_1(s)T_{R1}(s)+T_2(s)T_{R3}(s)=T_{GL}(s) \quad (4.1)$$

Let $\bar{T}_L(s)\in\mathbb{R}[s]^{p\times p}$ be another common left divisor of the matrices $T_1(s),T_2(s)$, that is

$$T_1(s)=\bar{T}_L(s)R(s)\ ,\ T_2(s)=\bar{T}_L(s)G(s) \quad (4.2)$$

where $R(s)\in\mathbb{R}[s]^{p\times l}$ and $G(s)\in\mathbb{R}[s]^{p\times t}$.

(4.1) and (4.2) combined result in :

$$T_1(s)T_{R1}(s)+T_2(s)T_{R3}(s)=\bar{T}_L(s)\left[R(s)T_{R1}(s)+G(s)T_{R3}(s)\right]=T_{GL}(s)$$

which means that $T_{GL}(s)$ is a right multiple of every common left divisor $\bar{T}_L(s)$.

By definition , this means that the matrix $T_{GL}(s)$ is the greatest common left divisor of the polynomial matrices $T_1(s),T_2(s)$.

▲

Brief Explanation of the algorithm used in Mathematica

- We ask the user to insert two polynomial matrices and we save their dimensions.
- After that we use two loops to form the matrix $T(s)=\begin{bmatrix} T_1(s) & T_2(s) \end{bmatrix}$.

- We then apply the algorithm for the Smith - Form that we used in the previous section in order to determine the matrix $T_R(s)$.
- Having done that , we simply pick the matrix $T_{GL}(s)$ from the product $T(s)T_R(s)$.
- We print the result and an appropriate message to the user.

Examples of running the program

We insert the following matrices :

- $T_1(s)=\begin{pmatrix}(s+1)(s+2) & 0\\ 0 & (s+1)(s+2)\end{pmatrix}$
- $T_2(s)=\begin{pmatrix}s+1 & s+2\\ s+1 & 0\end{pmatrix}$

The greatest common left divisor of them is the matrix :

```
LeftGcd[{{(s+1) (s+2), 0}, {0, (s+1) (s+2)}}, {{s+1, s+2}, {s+1, 0}}]
The greatest common left divisor of the matrices {{(1+s) (2+s), 0}, {0, (1+s) (2+s)}}, {{1+s, 2+s}, {1+s, 0}} is the matrix {{1, 0}, {-1-s, (1+s) (2+s)}}
```

(Figure 4.1)

If we insert the matrices :

- $T_1(s)=\begin{pmatrix}s+1 & s & s-1\\ s^2-2s & s^2-1 & s\end{pmatrix}$
- $T_2(s)=\begin{pmatrix}s & 1\\ s+1 & s+2\end{pmatrix}$

Then the greatest common left divisor will be :

```
LeftGcd[{{s+1, s, s-1}, {s^2-2s, s^2-1, s}}, {{s, 1}, {s+1, s+2}}]
The greatest common left divisor of the matrices {{1+s, s, -1+s}, {-2s+s^2, -1+s^2, s}}, {{s, 1}, {1+s, 2+s}} is the matrix {{1, 0}, {2+s, 1}}
```

(Figure 4.2)

CHAPTER 5

GREATEST COMMON RIGHT DIVISOR OF TWO POLYNOMIAL MATRICES

Given two polynomial matrices $T_1(s) \in \mathbb{R}[s]^{l\times m}$, $T_2(s) \in \mathbb{R}[s]^{t\times m}$ with the same number of columns $m \le l+t = p$, we want to determine the greatest common right divisor $T_{GR}(s) \in \mathbb{R}[s]^{m\times m}$.

The algorithm , as well as the proof , is similar to the one presented in the previous section.

So , we form the matrix :

$$T(s) = \begin{bmatrix} T_1(s) \\ T_2(s) \end{bmatrix} \in \mathbb{R}[s]^{p\times m}.$$

Next , we find a unimodular square matrix $T_L(s) \in \mathbb{R}[s]^{p\times p}$ using the algorithm of the chapter 3 so that :

$$T_L(s)T(s) = \begin{bmatrix} T_{GR}(s) \\ 0_{p-m,m} \end{bmatrix}.$$

The greatest common right divisor of the matrices $T_1(s), T_2(s)$ is the matrix $T_{GR}(s)$.

Examples of running the program

If we insert the matrices :

- $T_1(s) = \begin{pmatrix} s+1 & s \\ s^2-1 & s-1 \end{pmatrix}$
- $T_2(s) = \begin{pmatrix} s+1 & s-1 \end{pmatrix}$

Then the program yields the following matrix as a greatest common right divisor :

```
RightGcd[{{s + 1, s}, {s^2 - 1, s - 1}}, {{s + 1, s - 1}}]
```

The greatest common right divisor of the matrices $\begin{pmatrix} 1+s & s \\ -1+s^2 & -1+s \end{pmatrix}$, $(1+s \quad -1+s)$ is the matrix $\begin{pmatrix} -1-s & -s \\ 1+s & 1+s \end{pmatrix}$

(Figure 5.1)

CHAPTER 6

ROW PROPER REDUCTION OF A POLYNOMIAL MATRIX

We call *degree* of a polynomial matrix $T(s) \in \mathbb{R}[s]^{p\times m}$, the highest degree of all the maximum order non - zero minors of $T(s)$.

For instance , the degree of the polynomial matrix $T(s) = \begin{pmatrix} (s+1)^2 & 0 \\ 1 & 1 \\ 0 & s+1 \end{pmatrix}$ is equal to 3 , because the minor that is generated by the 1st and 3rd row has degree equal to 3.

If $p = m \Rightarrow \deg T(s) = \deg\left[\det T(s)\right]$.

Corollary 6.1 , [1]

$T(s) \in \mathbb{R}[s]^{p\times p}$ is unimodular $\Leftrightarrow \deg T(s) = 0$.

Definition 6.1 , [1]

The *row complexity* of $T(s)$ is the sum of the highest degrees of each row of $T(s)$ and is denoted by $c_r(T)$. Similarly , the *column complexity* of $T(s)$ is the sum of the highest degrees of each column of $T(s)$ and is denoted by $c_c(T)$.

For any polynomial matrix $T(s)$ it holds that :

$c_r(T) \geq \deg T(s)$ and $c_c(T) \geq \deg T(s)$.

The polynomial matrix $T(s)$ is :

row proper $\Leftrightarrow$ $c_r(T) = \deg T(s)$

column proper $\Leftrightarrow$ $c_c(T) = \deg T(s)$

Equivalently , the matrix $T(s)$ is :

row proper $\Leftrightarrow rank_{\mathbb{R}}\left[T(s)\right]_r^h = \min\{p,m\}$

column proper $\Leftrightarrow rank_{\mathbb{R}}\left[T(s)\right]_c^h = \min\{p,m\}$

where $\left[T(s)\right]_r^h$ is the highest row coefficient matrix , i.e. a $p\times m$ matrix which is formed by placing in the (i,j) position the coefficient of the highest degree of the i^{th} row , and $\left[T(s)\right]_c^h$ is the highest column coefficient matrix , i.e. a $p\times m$ matrix which is formed by placing in the (i,j) position the coefficient of the highest degree of the j^{th} column.

Corollary 6.2 , [1]

If r_i is the highest degree of the i^{th} row, $i=1,2,\ldots p$ and q_j is the highest degree of the j^{th} column , $j=1,2,\ldots m$, then :

- $T(s)=diag\left[s^{r_1}\quad s^{r_2}\quad \ldots\quad s^{r_p}\right]\left[T(s)\right]_r^h+T_r(s)$
- $T(s)=\left[T(s)\right]_c^h diag\left[s^{q_1}\quad s^{q_2}\quad \ldots\quad s^{q_m}\right]+T_c(s)$

For example , if $T(s)=\begin{pmatrix}2s^2+3s+1 & 1\\ 5s+2 & s^2\\ 5s^2+2s & -s^3\end{pmatrix}$, then :

- $T(s)=\begin{pmatrix}2 & 0\\ 0 & 0\\ 5 & -1\end{pmatrix}\begin{pmatrix}s^2 & 0\\ 0 & s^3\end{pmatrix}+\begin{pmatrix}2s+1 & 1\\ s+2 & s^2\\ 2s & 0\end{pmatrix}$
- $T(s)=\begin{pmatrix}s^2 & 0 & 0\\ 0 & s^2 & 0\\ 0 & 0 & s^3\end{pmatrix}\begin{pmatrix}2 & 0\\ 0 & 1\\ 0 & -1\end{pmatrix}+\begin{pmatrix}3s+1 & 1\\ 5s+2 & 0\\ 5s^2+2s & 0\end{pmatrix}$

Algorithm for row proper reduction of a polynomial matrix , [1]

Suppose that we have a polynomial matrix $T(s)\in\mathbb{R}[s]^{p\times m}$. Then there exists a square unimodular matrix $T_L(s)\in\mathbb{R}[s]^{p\times p}$ such that the matrix :

$\bar{T}(s)=T_L(s)T(s)\in\mathbb{R}[s]^{p\times m}$ is row – proper.

Proof of the row proper reduction

If $T(s)$ is not row proper then $c_r(T)>\deg T(s)$.

We pick the matrix $T_L(s)$ so that the row complexity of the matrix $\bar{T}(s)$ can be reduced :

$c_r\left(\bar{T}(s)\right)=c_r\left(T_L(s)T(s)\right)<c_r(T)$ till it becomes equal to the degree :

$\deg\bar{T}(s)=\deg T_L(s)T(s)=\deg T_L(s)+\deg T(s)=\deg T(s)$.

When $c_r\left(\bar{T}\right)=\deg\bar{T}(s)$, then the algorithm can be terminated.

The matrix $T_L(s)$ can be picked according to the following procedure :

If $T(s)$ is not row proper then :

$rank_{\mathbb{R}}\left[T(s)\right]_r^h<p$.

Let $a^T=\begin{bmatrix}a_1 & a_2 & \dots & a_p\end{bmatrix}\in\mathbb{R}^{1\times p}$ such that : $a^T\left[T(s)\right]_r^h=0$.

Now let $r_0=\max\{r_i\}$ and $a(s)^T=\begin{bmatrix}a_1s^{r_0-r_1} & a_2s^{r_0-r_2} & \dots & a_ps^{r_0-r_p}\end{bmatrix}\in\mathbb{R}[s]^{1\times p}$, where $\deg a(s)^T<r_0$.

Then we notice that :

$a(s)^T T(s)=a(s)^T\left[diag\begin{bmatrix}s^{r_1} & \dots & s^{r_p}\end{bmatrix}\left[T(s)\right]_r^h+T_r(s)\right]=$

$$\begin{bmatrix} a_1 s^{r_0-r_1} & a_2 s^{r_0-r_2} & \dots & a_p s^{r_0-r_p} \end{bmatrix} \begin{bmatrix} s^{r_1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & s^{r_p} \end{bmatrix} \left[T(s)\right]_r^h + \begin{bmatrix} a_1 s^{r_0-r_1} & a_2 s^{r_0-r_2} & \dots & a_p s^{r_0-r_p} \end{bmatrix} T_r(s) =$$

$$s^{r_0} a^T \left[T(s)\right]_r^h + a(s)^T T_r(s) = a(s)^T T_r(s).$$

We consider the identity matrix I_p , and we replace the i_0 row (the row that gives r_0) with $a(s)^T$ and we denote this matrix by $T_{L_{i_0}}(s)$.

The matrix $T_{i_0}(s) = T_{L_{i_0}}(s) T(s)$ has the same rows as $T(s)$ except for the row i_0.

So this fact means that :

$c_r(T_{i_0}) < c_r(T)$, and since $T_{L_{i_0}}(s)$ is unimodular $\Rightarrow \deg T_{i_0}(s) = \deg T(s)$.

▲

Example of the row – properness algorithm

$$T(s) = \begin{pmatrix} 1 & s^2 & 0 \\ 0 & s & 1 \end{pmatrix}, \quad \left[T(s)\right]_r^h = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

$c_r(T) = 3 > 2 = \deg T(s) \Rightarrow T(s)$ is not row proper.

We see that : $\begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$, so $a^T = \begin{bmatrix} 1 & -1 \end{bmatrix}$.

$r_1 = \deg\begin{bmatrix} 1 & s^2 & 0 \end{bmatrix} = 2$, $r_2 = \deg\begin{bmatrix} 0 & s & 1 \end{bmatrix} = 1$.

$r_0 = \max\{r_1, r_2\} = r_1 = 2 \Rightarrow i_0 = 1$.

$a(s)^T = \begin{bmatrix} 1 & -s \end{bmatrix}$

$$T_{L1}(s) = \begin{bmatrix} 1 & -s \\ 0 & 1 \end{bmatrix}$$

$$\bar{T}(s)=T_{L1}(s)T(s)=\begin{bmatrix}1 & -s\\ 0 & 1\end{bmatrix}\begin{bmatrix}1 & s^2 & 0\\ 0 & s & 1\end{bmatrix}=\begin{bmatrix}1 & 0 & -s\\ 0 & s & 1\end{bmatrix}$$

And the polynomial matrix $\bar{T}(s)$ is row – proper.

Brief Explanation of the algorithm used in Mathematica

- First we ask the user to enter the matrix and we save the dimensions . Then we proceed to the row – properness algorithm.
- The variable **highestrowdegree** is used to save the values of the r_i , while the variable **use** is an array that is used to save the matrix $T_{L_{i_0}}$ each time the algorithm is executed , that is until the matrix T_{i_0} becomes row – proper.
- We insert a $While$ loop to start the algorithm.
- We then use a slightly modified For loop to determine the values of the r_i and find the array **highestrowdegree.**
- The next step is to find the degree of the given matrix , by using the Minors command and by applying a loop to find the maximum degree of the elements of the matrix that is generated using the Minors command.
- We calculate the row complexity of the given matrix by applying a loop to find the sum of the elements of the array **highestrowdegree.**
- If the row complexity is equal to the degree of the matrix , that is if the matrix is row – proper , then the algorithm is terminated. Otherwise , we execute the steps that we described above.
- We determine the matrix $\left[T(s)\right]_r^h$ which is saved using the variable **highestrowcoef.**
- We find a numerical solution of the system of equations $a^T\left[T(s)\right]_r^h=0$ (unknown variables $w[1],w[2],\ldots,w[p]$). This action is performed by using the command $NSolve\left[\{eq_1,eq_2,\ldots\},Integers\right]$ which solves the system of equations over the domain of integer numbers and produces results in terms

of the constants $C[1], C[2], \ldots$. We use a loop to replace each of the previous constants with the number 1. In this way , we come up with a pure numerical solution of the system that we want to solve.

- When solving a system of equations that contain the unknown variables $w[1], w[2], \ldots w[p]$ with the NSolve command , Mathematica will yield results in the form : $\{\{w[1] \to c_1, w[2] \to c_2, \ldots, w[p] \to c_p\}\}$, $c_i \in \mathbb{N}$. We save this solution with the use of the variable **sol .** By typing $sol[[1,1,1]]$ we refer to the 1st part of the 1st solution , that is $w[1]$. By typing $sol[[1,1,2]]$ we refer to the 2nd part of the 1st solution , that is c_1. This pattern is used to manipulate the parts of the solution **sol**
- We determine r_0 and i_0 (variables $r0$, x) , and we form the matrix $a(s)^T$ with the use of the matrix **row.**
- We form the matrix $T_{L_{i_0}}$ (variable $matrix$) , and we calculate the product $T_{L_{i_0}}(s)T(s) = T_{i_0}(s)$. The matrix $T_{L_{i_0}}$ is saved using the array **use.** We can see that the variable **count** that was inserted at the start of the algorithm is used to count the number of the matrices $T_{L_{i_0}}$, in other words , to count how many times the algorithm is executed until we get a row –proper matrix .
- We use the command $Clear[w]$ to clear the values and the definitions of the variables $w[1], w[2], \ldots, w[p]$ and we repeat the previous steps.
- We calculate the product of the matrices of the array **use** (matrix $\to mtr$). This is the matrix that transforms the given matrix into a row – proper one.
- We print the result and we display an appropriate message to the user.

Remark 6.1

- If the matrix $mtr = I_p$, then this fact means that the given matrix was row – proper.

Examples of running the program

If the matrix $T(s)=\begin{pmatrix}(s+1)^2 & 0\\ 1 & 1\\ 0 & s+1\end{pmatrix}$ is inserted :

```
RowProper[{{(s+1)^2, 0}, {1, 1}, {0, s+1}}]
```

The left multiplication of the given matrix $\begin{pmatrix}(1+s)^2 & 0\\ 1 & 1\\ 0 & 1+s\end{pmatrix}$ with the matrix $\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}$ yields the matrix $\begin{pmatrix}(1+s)^2 & 0\\ 1 & 1\\ 0 & 1+s\end{pmatrix}$ which is row-proper

(Figure 6.1)

So the initial matrix is row – proper.

If we insert the matrix $T(s)=\begin{pmatrix}2s^2+3s+1 & s^2+2 & s\\ 4s^3 & s^3+4s & 3s^2+3\end{pmatrix}$:

```
RowProper[{{2 s^2+3 s+1, s^2+2, s}, {4 s^3, s^3+4 s, 3 s^2+3}}]
```

The left multiplication of the given matrix $\begin{pmatrix}1+3s+2s^2 & 2+s^2 & s\\ 4s^3 & 4s+s^3 & 3+3s^2\end{pmatrix}$ with the matrix $\begin{pmatrix}1&0\\0&1\end{pmatrix}$ yields the matrix $\begin{pmatrix}1+3s+2s^2 & 2+s^2 & s\\ 4s^3 & 4s+s^3 & 3+3s^2\end{pmatrix}$ which is row-proper

(Figure 6.2)

So , again , the initial matrix is row – proper.

If we insert the matrix $T(s)=\begin{pmatrix}s & 1\\ 1 & s^2\\ s+1 & 0\end{pmatrix}$:

```
RowProper[{{s, 1}, {1, s^2}, {s+1, 0}}]
```

The left multiplication of the given matrix $\begin{pmatrix}s & 1\\ 1 & s^2\\ 1+s & 0\end{pmatrix}$ with the matrix $\begin{pmatrix}1&0&-1\\s&0&-s\\0&0&1\end{pmatrix}$ yields the matrix $\begin{pmatrix}-1 & 1\\ -s & s\\ 1+s & 0\end{pmatrix}$ which is row-proper

```
count
2
```

(Figure 6.3)

If we examine the structure of the matrix $T_L(s)$ carefully , we will notice that the row – properness algorithm has been executed more than once.

If we insert the matrix $T(s)=\begin{pmatrix} s^3 & s^5 \\ 1 & s^7+s^6 \end{pmatrix}$:

```
RowProper[{{s^3, s^5}, {1, s^6 + s^7}}]
```

The left multiplication of the given matrix $\begin{pmatrix} s^3 & s^5 \\ 1 & s^6+s^7 \end{pmatrix}$ with the matrix $\begin{pmatrix} 1 & 0 \\ s\,(1+s) & -1 \end{pmatrix}$ yields the matrix $\begin{pmatrix} s^3 & s^5 \\ -1+s^4+s^5 & 0 \end{pmatrix}$ which is row-proper

```
count
2
```

(Figure 6.4)

CHAPTER 7

COLUMN PROPER REDUCTION OF A POLYNOMIAL MATRIX

The column – properness algorithm is exactly the same as the row – properness algorithm except for the fact that all the actions are performed on columns.

For instance , let $q_0 = \max\{q_j\}$ and $a(s) = \begin{bmatrix} a_1 s^{q_0 - q_1} \\ a_2 s^{q_0 - q_2} \\ \vdots \\ a_m s^{q_0 - q_m} \end{bmatrix} \in \mathbb{R}[s]^{m \times 1}$, where $\deg a(s) < q_0$.

We consider the identity matrix I_m , and we replace the j_0 column (the column that gives q_0) with $a(s)$ and we denote this matrix by $T_{R_{j_0}}(s)$.

The matrix $T_{j_0}(s) = T(s) T_{R_{j_0}}(s)$ has the same columns as $T(s)$ except for the column j_0.

The matrix $T_{R_{j_0}}$ is used to transform the initial matrix into a column – proper one.

We present an example to indicate how the column – properness algorithm works.

Example of the column – properness algorithm

$$T(s) = \begin{pmatrix} s+1 & 3s^2+2 \\ 1 & 0 \\ 2 & -s \end{pmatrix}, \quad [T(s)]_c^h = \begin{pmatrix} 1 & 3 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

$c_c(T) = 3 > 2 = \deg T(s) \Rightarrow T(s)$ is not column proper.

We see that : $\begin{pmatrix} 1 & 3 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 3 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$, so $a = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$.

$$q_1 = \deg \begin{bmatrix} s+1 \\ 1 \\ 2 \end{bmatrix} = 1 \ , \ q_2 = \deg \begin{bmatrix} 3s^2+2 \\ 0 \\ -s \end{bmatrix} = 2 .$$

$q_0 = \max\{q_1, q_2\} = q_2 = 2 \Rightarrow j_0 = 2$.

$$a(s) = \begin{bmatrix} 3s \\ -1 \end{bmatrix}$$

$$T_{R2}(s) = \begin{bmatrix} 1 & 3s \\ 0 & -1 \end{bmatrix}$$

$$\bar{T}(s) = T(s)T_{R2}(s) = \begin{pmatrix} s+1 & 3s^2+2 \\ 1 & 0 \\ 2 & -s \end{pmatrix}\begin{pmatrix} 1 & 3s \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} s+1 & 3s-2 \\ 1 & 3s \\ 2 & 7s \end{pmatrix}$$

And the polynomial matrix $\bar{T}(s)$ is column – proper.

<u>Examples of running the program</u>

If we insert the matrix $T(s) = \begin{pmatrix} s^2 & 1 & 2 \\ 0 & s & 1 \end{pmatrix}$:

```
ColumnProper[{{s^2, 1, 2}, {0, s, 1}}]
```

The right multiplication of the given matrix $\begin{pmatrix} s^2 & 1 & 2 \\ 0 & s & 1 \end{pmatrix}$ with the matrix $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ yields the matrix $\begin{pmatrix} s^2 & 1 & 2 \\ 0 & s & 1 \end{pmatrix}$ which is column-proper

(Figure 7.1)

So the initial matrix is column – proper.

If the matrix $T(s) = \begin{pmatrix} s^2 & s^2 \\ 1 & s \\ s+1 & 0 \end{pmatrix}$ is inserted :

```
ColumnProper[{{s^2, s^2}, {1, s}, {s + 1, 0}}]
```

The right multiplication of the given matrix $\begin{pmatrix} s^2 & s^2 \\ 1 & s \\ 1+s & 0 \end{pmatrix}$ with the matrix $\begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$ yields the matrix $\begin{pmatrix} 0 & s^2 \\ 1-s & s \\ 1+s & 0 \end{pmatrix}$ which is column-proper

```
count
1
```

(Figure 7.2)

CHAPTER 8

LEFT MATRIX FRACTION DESCRIPTION (LMFD)

Suppose that we have a rational matrix $T(s)=\left[t_{ij}(s)\right]\in\mathbb{R}(s)^{p\times m}$, $t_{ij}(s)=\dfrac{n_{ij}(s)}{d_{ij}(s)}$.

If $d(s)$ is the least common multiple of the denominators of the elements of the matrix $T(s)$ then :

$$T(s)=\frac{1}{d(s)}\cdot N(s)=d(s)^{-1}N(s)=d(s)^{-1}\cdot I_p\cdot N(s)=\begin{pmatrix} d(s)^{-1} & 0 & \dots & 0\\ 0 & d(s)^{-1} & 0 & \vdots\\ \vdots & 0 & \ddots & 0\\ 0 & \dots & 0 & d(s)^{-1}\end{pmatrix}_{p\times p}\cdot N(s)=$$

$$\begin{pmatrix} d(s) & 0 & \dots & 0\\ 0 & d(s) & 0 & \vdots\\ \vdots & 0 & \ddots & 0\\ 0 & \dots & 0 & d(s)\end{pmatrix}_{p\times p}^{-1}\cdot N(s)$$

Let $\tilde{D}_L=\begin{pmatrix} d(s) & 0 & \dots & 0\\ 0 & d(s) & 0 & \vdots\\ \vdots & 0 & \ddots & 0\\ 0 & \dots & 0 & d(s)\end{pmatrix}_{p\times p}$, $\tilde{N}_L=N(s)$.

The product $\tilde{D}_L^{-1}\cdot\tilde{N}_L$ is a left fraction description of the matrix $T(s)$.

Definition 8.1 ,[1] & [3]

The matrices $\tilde{D}_L,\tilde{N}_L$ are left coprime if :

- $rank_{\mathbb{R}}\left[\tilde{D}_L(s_i)\quad \tilde{N}_L(s_i)\right]=\min\{p,m\},\forall s_i\in\mathbb{C}$
- The greatest common left divisor of $\tilde{D}_L(s),\tilde{N}_L(s)$ is unimodular

And the above conditions are equivalent.

If the matrices $\tilde{D}_L(s), \tilde{N}_L(s)$ are not left coprime , that is if the greatest common left divisor is not a unimodular matrix , then we perform the following actions :

Let G_L be the greatest common left divisor of the matrices $\tilde{D}_L(s), \tilde{N}_L(s)$.

This implies that there exist matrices D_L, N_L such that :

$\tilde{D}_L = G_L D_L \Rightarrow D_L = G_L^{-1}\tilde{D}_L$, $\tilde{N}_L = G_L N_L \Rightarrow N_L = G_L^{-1}\tilde{N}_L$.

We now consider the left fraction description $D_L^{-1}N_L$.

The matrices D_L, N_L are left coprime and in addition :

$$D_L^{-1}N_L = \left(G_L^{-1}\tilde{D}_L\right)^{-1} G_L^{-1}\tilde{N}_L = \tilde{D}_L^{-1}G_L G_L^{-1}\tilde{N}_L = \tilde{D}_L^{-1}\tilde{N}_L = T(s)$$

Hence , the new product $D_L^{-1}N_L$ is a left fraction description of the matrix $T(s)$ where the matrices D_L, N_L are left coprime.

Our goal is to find a description $\bar{D}_L^{-1}\bar{N}_L$, where the matrices $\bar{D}_L, \bar{N}_L$ are left coprime and the matrix $\bar{D}_L$ is row – proper.

To achieve that we act as follows :

If the matrix D_L is not row – proper , then we apply the row – properness algorithm that was described in the previous section , and with the help of a matrix T_L , we gain a row – proper matrix $\bar{D}_L = T_L D_L$.

We also consider a new matrix $\bar{N}_L = T_L N_L$.

Since $\bar{D}_L^{-1}\bar{N}_L = \left(T_L D_L\right)^{-1} T_L N_L = D_L^{-1}T_L^{-1}T_L N_L = D_L^{-1}N_L = T(s)$, we conclude that this left fraction description is the one that we are looking for.

In light of the above analysis , we provide the following algorithm :

Algorithm for finding the LMFD , [1] :

1. Given a rational matrix $T(s)\in\mathbb{R}(s)^{p\times m}$, we first find a description of the form $\tilde{D}_L^{-1}\cdot\tilde{N}_L$, where $\tilde{N}_L\in\mathbb{R}[s]^{p\times m}$ and $\tilde{D}_L\in\mathbb{R}[s]^{p\times p}$.
2. If the matrices $\tilde{N}_L,\tilde{D}_L$ are not left coprime , then we consider the matrices $D_L=G_L^{-1}\cdot\tilde{D}_L$, $N_L=G_L^{-1}\cdot\tilde{N}_L$, where G_L is the greatest common left divisor of the matrices $\tilde{N}_L,\tilde{D}_L$.
3. If the matrix D_L is not row – proper , then we determine a matrix T_L such that the matrix $\bar{D}_L=T_L\cdot D_L$ is row - proper. We also consider $\bar{N}_L=T_L\cdot N_L$.
4. The left matrix fraction description is the product $T(s)=\bar{D}_L^{-1}\cdot\bar{N}_L$.

Example of the left fraction description algorithm

Suppose that $T(s)=\begin{pmatrix}\frac{1}{s+2} & \frac{1}{s+1}\\ \frac{1}{s+2} & 0\end{pmatrix}$. $d(s)=(s+1)(s+2)$

So $\tilde{D}_L(s)=\begin{bmatrix}(s+1)(s+2) & 0\\ 0 & (s+1)(s+2)\end{bmatrix}$, $\tilde{N}_L(s)=\begin{bmatrix}s+1 & s+2\\ s+1 & 0\end{bmatrix}$.

We run the program for the greatest common left divisor to see if the above matrices are left coprime. So if we insert $\tilde{D}_L,\tilde{N}_L$ we get :

```
LeftGcd[{{(s+1) (s+2), 0}, {0, (s+1) (s+2)}}, {{s+1, s+2}, {s+1, 0}}]
```

The greatest common left divisor of the matrices $\begin{pmatrix}(1+s)(2+s) & 0\\ 0 & (1+s)(2+s)\end{pmatrix}$, $\begin{pmatrix}1+s & 2+s\\ 1+s & 0\end{pmatrix}$ is the matrix $\begin{pmatrix}1 & 0\\ -1-s & (1+s)(2+s)\end{pmatrix}$

```
Det[leftgcd]
```

$2+3s+s^2$

(Figure 8.1)

The matrices are not left coprime.

Hence , we consider :

- $D_L(s)=\begin{pmatrix}(s+1)(s+2) & 0\\ s+1 & 1\end{pmatrix}=G_L^{-1}\tilde{D}_L$
- $N_L(s)=\begin{pmatrix}s+1 & s+2\\ 1 & 1\end{pmatrix}=G_L^{-1}\tilde{N}_L$

Now ,

```
LeftGcd[{{(s + 1) (s + 2), 0}, {s + 1, 1}}, {{s + 1, s + 2}, {1, 1}}]
```

The greatest common left divisor of the matrices $\begin{pmatrix}(1+s)(2+s) & 0\\ 1+s & 1\end{pmatrix}, \begin{pmatrix}1+s & 2+s\\ 1 & 1\end{pmatrix}$ is the matrix $\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$

```
Det[leftgcd]
-1
```

(Figure 8.2)

We now run the row – properness algorithm for the matrix $D_L(s)$. The outcome is :

```
RowProper[{{(s + 1) (s + 2), 0}, {s + 1, 1}}]
```

The left multiplication of the given matrix $\begin{pmatrix}(1+s)(2+s) & 0\\ 1+s & 1\end{pmatrix}$ with the matrix $\begin{pmatrix}1 & -s\\ 0 & 1\end{pmatrix}$ yields the matrix $\begin{pmatrix}2(1+s) & -s\\ 1+s & 1\end{pmatrix}$ which is row-proper

(Figure 8.3)

So , :

- $T_L(s)=\begin{pmatrix}1 & -s\\ 0 & 1\end{pmatrix}$
- $\bar{D}_L(s)=T_L(s)D_L(s)=\begin{pmatrix}2(1+s) & -s\\ s+1 & 1\end{pmatrix}\rightarrow row-proper$
- $\bar{N}_L(s)=T_L(s)N_L(s)=\begin{pmatrix}1 & 2\\ 1 & 1\end{pmatrix}$

Finally , the left fraction description is $\begin{pmatrix}-2(1+s) & s\\ s+1 & 1\end{pmatrix}^{-1}\cdot\begin{pmatrix}1 & 2\\ 1 & 1\end{pmatrix}$

Examples of running the program

If we insert the matrix $T(s)=\begin{pmatrix} \frac{1}{s} & \frac{1}{s+1} & \frac{1}{s-1} \\ 1 & 1 & s \end{pmatrix}$, then the results are :

```
LMFD[{{1/s, 1/(s+1), 1/(s-1)}, {1, 1, s}}]
The left fraction description of the matrix
{{1/s, 1/(1+s), 1/(-1+s)}, {1, 1, s}} is {{s(-1+s^2), 0}, {0, 2}}^-1 {{-1+s^2, (-1+s) s, s (1+s)}, {2, 2, 2 s}}
```

(Figure 8.4)

If the matrix $T(s)=\begin{pmatrix} \frac{1}{s} & 0 \\ \frac{1}{s+1} & \frac{1}{s(s+1)} \\ 1 & \frac{1}{s} \end{pmatrix}$ is inserted , then :

```
LMFD[{{1/s, 0}, {1/(s+1), 1/(s (s+1))}, {1, 1/s}}]
The left fraction description of the matrix {{1/s, 0}, {1/(1+s), 1/(s (1+s))}, {1, 1/s}} is {{0, 0, s}, {s, 0, 0}, {0, -1-s, 1}}^-1 {{s, 1}, {1, 0}, {0, 0}}
```

(Figure 8.5)

CHAPTER 9

RIGHT MATRIX FRACTION DESCRIPTION (RMFD)

The Right Matrix Fraction Description algorithm is similar to the one for the Left Matrix Fraction Description , with a few slight changes.

We briefly mention the basic steps that are required :

1. Given a rational matrix $T(s) \in \mathbb{R}(s)^{p\times m}$, we first find a description of the form $\tilde{N}_R \cdot \tilde{D}_R^{-1}$, where $\tilde{N}_R \in \mathbb{R}[s]^{p\times m}$ and $\tilde{D}_R \in \mathbb{R}[s]^{m\times m}$.
2. If the matrices $\tilde{N}_R, \tilde{D}_R$ are not right coprime , then we consider the matrices $D_R = \tilde{D}_R \cdot G_R^{-1}$, $N_R = \tilde{N}_R \cdot G_R^{-1}$, where G_R is the greatest common right divisor of the matrices $\tilde{N}_R, \tilde{D}_R$.
3. If the matrix D_R is not column – proper , then we determine a matrix T_R such that the matrix $\bar{D}_R = D_R \cdot T_R$ is column proper. We also consider $\bar{N}_R = N_R \cdot T_R$.
4. The right matrix fraction description is the product $T(s) = \bar{N}_R \cdot \bar{D}_R^{-1}$.

Examples of running the program

If we insert the matrix $T(s) = \begin{pmatrix} \frac{1}{s+1} & \frac{1}{s+2} \\ \frac{1}{s+1} & 0 \end{pmatrix}$, then the program displays :

```
RMFD[{{1 / (s + 1), 1 / (s + 2)}, {1 / (s + 1), 0}}]
The right fraction description of the matrix
```

$$\begin{pmatrix} \frac{1}{1+s} & \frac{1}{2+s} \\ \frac{1}{1+s} & 0 \end{pmatrix} \text{ is } \begin{pmatrix} -1 & 0 \\ -2 & -1 \end{pmatrix} \begin{pmatrix} -2(1+s) & -1-s \\ 2+s & 2+s \end{pmatrix}^{-1}$$

(Figure 9.1)

CHAPTER 10

SOLUTION OF POLYNOMIAL MATRIX DIOPHANTINE EQUATIONS

Given a strictly proper transfer function matrix $P(s) \in \mathbb{R}_{sp}(s)^{p \times m}$ (plant) , with a right matrix fraction description $P(s) = N_R(s) D_R(s)^{-1}$, we will present an algorithm that calculates the class $\Phi(P, D_C)$ of proper compensators $C(s) := X_L(s)^{-1} Y_L(s)$ that when employed in a unity feedback loop , result in closed – loop systems $S(P, D_C)$ with a desired denominator $D_C(s)$.

Equivalently , a numerical algorithm , [2], is derived for the computation of all the polynomial solutions $X_L(s), Y_L(s)$ of the polynomial matrix Diophantine equation :

$$X_L(s) D_R(s) + Y_L(s) N_R(s) = D_C(s) \qquad (10.1)$$

- Step 1

We obtain a right matrix fraction description $N_R(s) D_R(s)^{-1}$ of $P(s)$, where $D_R(s)$ is column – proper and let $k_i, i \in \{1, 2, \ldots, m\}$ denote the column degrees of $D_R(s)$.

- Step 2

We form the matrix $F(s) = \begin{bmatrix} D_R(s) \\ N_R(s) \end{bmatrix} \in \mathbb{R}[s]^{(m+p) \times m}$ and we determine the minimum k for which M_{ek} has full column rank. Then $\mu = k$ and we choose $\xi_i \geq \mu - 1, i \in \{1, 2, \ldots, m\}$.

First of all we need to give the definition of the generalized Sylvester resultant $R_k^{F(s)}$ of $F(s)$ of order k. Let $F(s) = F_0 + sF_1 + s^2 F_2 + \ldots + s^q F_q$, $F_i \in \mathbb{R}^{(m+p) \times m}, i = 0, 1, 2, \ldots, q$

Then $R_k^{F(s)} = \begin{bmatrix} F_0 & F_1 & \dots & F_q & 0 & \dots & 0 \\ 0 & F_0 & F_1 & \dots & F_q & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & & \ddots & 0 \\ 0 & 0 & \dots & F_0 & F_1 & \dots & F_q \end{bmatrix} \in \mathbb{R}^{k(m+p)\times m(q+k)}$ (10.2)

We now consider the matrix $F(s)$ in terms of its columns , so that :

$$F(s) = \begin{bmatrix} f_1(s) & f_2(s) & \dots & f_m(s) \end{bmatrix}$$

with $f_i(s) = \begin{bmatrix} f_{i1}(s) \\ f_{i2}(s) \\ \vdots \\ f_{i(m+p)}(s) \end{bmatrix} = f_{i0} + s f_{i1} + \dots + s^{q_i} f_{iq_i} \in \mathbb{R}[s]^{(m+p)\times 1}$, $i = 1, 2, \dots, m$

If $R_k^{f_i(s)} = \begin{bmatrix} f_{i0} & f_{i1} & \dots & f_{iq_i} & 0 & \dots & 0 \\ 0 & f_{i0} & f_{i1} & \dots & f_{iq_i} & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & & \ddots & 0 \\ 0 & 0 & \dots & f_{i0} & f_{i1} & \dots & f_{iq_i} \end{bmatrix} \in \mathbb{R}^{k(m+p)\times(k+q_i)}$ is the generalized Sylvester resultant of order k of the column $f_i(s)$, then the generalized Wolovich resultant of $F(s)$ of order k , M_{ek} , is given by :

$$M_{ek} = \begin{bmatrix} R_k^{f_1(s)} & R_k^{f_2(s)} & \dots & R_k^{f_m(s)} \end{bmatrix} \in \mathbb{R}^{k(m+p)\times\left(mk+\sum_{i=1}^{m} q_i\right)}$$ (10.3)

Example 10.1

If $P(s) = \begin{bmatrix} \frac{s+1}{s(s-2)} & 0 \\ \frac{1}{s(s-1)} & \frac{1}{s-1} \end{bmatrix} = \begin{bmatrix} s+1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} s^2-2s & 0 \\ 1 & s-1 \end{bmatrix}^{-1} = N_R(s) D_R(s)^{-1}$

then $F(s)=\begin{bmatrix} s^2-2s & 0 \\ 1 & s-1 \\ s+1 & 0 \\ 1 & 1 \end{bmatrix}=\begin{bmatrix} f_1(s) & f_2(s) \end{bmatrix}$, $\deg f_1(s)=q_1=2$,

$\deg f_2(s)=q_2=1$

$$f_1(s)=\begin{bmatrix} s^2-2s \\ 1 \\ s+1 \\ 1 \end{bmatrix}=s^2 f_{12}+sf_{11}+f_{10}=s^2\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}+s\begin{bmatrix} -2 \\ 0 \\ 1 \\ 0 \end{bmatrix}+\begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$f_2(s)=\begin{bmatrix} 0 \\ s-1 \\ 0 \\ 1 \end{bmatrix}=sf_{21}+f_{20}=s\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}+\begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \end{bmatrix}.$$ For $k=1$:

$$M_{e1}=\begin{bmatrix} R_1^{f_1(s)} & R_1^{f_2(s)} \end{bmatrix}\in\mathbb{R}^{(m+p)k\times\left(mk+\sum_{i=1}^{m}q_i\right)}=\mathbb{R}^{(2+2)1\times(2\cdot1+1+2)}=\mathbb{R}^{4\times5}$$

$$R_1^{f_1(s)}=\begin{bmatrix} f_{10} & f_{11} & f_{12} \end{bmatrix}=\begin{bmatrix} 0 & -2 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$R_1^{f_2(s)}=\begin{bmatrix} f_{20} & f_{21} \end{bmatrix}=\begin{bmatrix} 0 & 0 \\ -1 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}$$

$$M_{e1}=\begin{bmatrix} R_1^{f_1(s)} & R_1^{f_2(s)} \end{bmatrix}=\begin{bmatrix} 0 & -2 & 1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}\in\mathbb{R}^{4\times5}$$

For $k=2$: $M_{e2}=\begin{bmatrix} R_2^{f_1(s)} & R_2^{f_2(s)} \end{bmatrix}\in\mathbb{R}^{(m+p)k\times\left(mk+\sum_{i=1}^{m}q_i\right)}=\mathbb{R}^{(2+2)2\times(2\cdot2+1+2)}=\mathbb{R}^{8\times7}$

$$R_2^{f_1(s)}=\begin{bmatrix} f_{10} & f_{11} & f_{12} & 0 \\ 0 & f_{10} & f_{11} & f_{12} \end{bmatrix}=\begin{bmatrix} 0 & -2 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -2 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}\in\mathbb{R}^{8\times 4}$$

$$R_2^{f_2(s)}=\begin{bmatrix} f_{20} & f_{21} & 0 \\ 0 & f_{20} & f_{21} \end{bmatrix}=\begin{bmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\in\mathbb{R}^{8\times 3}$$

$$M_{e2}=\begin{bmatrix} R_2^{f_1(s)} & R_2^{f_2(s)} \end{bmatrix}=\begin{bmatrix} 0 & -2 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}\in\mathbb{R}^{8\times 7}$$

Similarly , we can compute the generalized Wolovich resultant M_{ek} for $k\geq 3$.

- Step 3

We construct the generalized Wolovich resultant $M_{e(\xi+1)}$, where $\xi=\max_{i\in\{1,2,\ldots,m\}}\{\xi_i\}$

- Step 4

We choose a diagonal matrix $D_C(s)\in\mathbb{R}[s]^{m\times m}$ which is row and column proper with row and column degrees equal to ξ_i+k_i , and we construct the

matrix $\bar{D}_{\xi+1}$ **by decomposing** $D_C(s)=\bar{D}_{\xi+1}S_{e(\xi+1)}(s)$, **with**

$$S_{ek}(s)=\underset{i\in\{1,2,\ldots,m\}}{blockdiagonal}\left\{\begin{bmatrix}1\\ s\\ \vdots\\ s^{k_i+k-1}\end{bmatrix}\right\}\in\mathbb{R}^{\left(mk+\sum_{i=1}^{m}k_i\right)\times m}$$

To indicate how this decomposition works , suppose that we consider the plant of the previous example. After a few calculations , it turns out that the matrix M_{e2} has full column rank. Hence , $\mu=2$ and we choose $\xi_1=1\geq 2-1=1$, $\xi_2=2\geq 2-1=1$. Moreover it holds that $k_1=2$, $k_2=1$ and $\xi=\max\{\xi_i\}=2$.

So let us consider $D_C(s)=\begin{bmatrix}s^3+8s^2+24s+32 & 0\\ 0 & s^3+15s^2+62s+48\end{bmatrix}$.

Then $D_C(s)=\bar{D}_3S_{e3}(s)=\begin{bmatrix}32 & 24 & 8 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 48 & 62 & 15 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ s & 0\\ s^2 & 0\\ s^3 & 0\\ s^4 & 0\\ 0 & 1\\ 0 & s\\ 0 & s^2\\ 0 & s^3\end{bmatrix}$

- Step 5

We construct the compound matrix $\bar{M}_{e(\xi+1)}=\begin{bmatrix}M_{e(\xi+1)}\\ \bar{D}_{\xi+1}\end{bmatrix}$

- Step 6

We reduce $\bar{M}_{e(\xi+1)}$ **into column echelon form to obtain** $\bar{R}_{e(\xi+1)}=\begin{bmatrix}R_{e(\xi+1)}\\ \Delta_{\xi+1}\end{bmatrix}$

- Step 7

We compute the general solution for each row $\bar{\omega}_i^T$, $i=1,2,\ldots,m$, using the first $(\xi_i+1)(p+m)$ rows of $\bar{R}_{e(\xi+1)}$ (matrix denoted X) and the i^{th} row of $\Delta_{\xi+1}$ (matrix denoted Y) , discarding the last $(\xi-\xi_i)m$ columns on both matrices , by solving the system $\bar{\omega}_i^T \cdot X = Y$

- Step 8

Using the formula $\omega_i^T(s)=\sum_{j=0}^{\xi_i}\omega_{ij}^T s^j$, $\omega_{ij}^T \in \mathbb{R}^{1\times(m+p)}, i=1,2,\ldots,m$, where

$\bar{\omega}_i^T=\left[\omega_{i0}^T,\omega_{i1}^T,\ldots,\omega_{i\xi_i}^T\right]\in\mathbb{R}^{1\times(p+m)(\xi_i+1)}, i=1,2,\ldots,m$, we calculate the row $\omega_i^T(s)$ of the matrix $\Omega(s)=\left[X_L(s)\quad Y_L(s)\right]\in\mathbb{R}[s]^{m\times(m+p)}$, thus computing the polynomial matrices $X_L(s), Y_L(s)$ and the compensator $C(s)$.

For further analysis and explanations on the above mentioned algorithm the reader is referred to [2].

We have designed a program that performs all the above steps , and in addition , enables the user to manipulate and examine the plot of the step responses of the closed – loop system that emerges.

<u>Example of running the Diophantine Equations code</u>

Suppose that we insert the matrix $P(s)=\begin{bmatrix}\frac{s+1}{s^2} & 0\\ \frac{1}{s(s-1)} & \frac{1}{1-s}\end{bmatrix}$. Then :

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

- $\mu = 2, \xi_1 = 1, \xi_2 = 2$
- Desired Zeros : $-1, -1+i, -1-i, -2, -3+2i, -3-2i$ (inserted by the user)
- $D_C(s) = \begin{bmatrix} s^3+3s^2+4s+2 & 0 \\ 0 & (s+2)\left[(s+3-2i)(s+3+2i)\right] \end{bmatrix}$

The solution of the polynomial matrix Diophantine equation : $X_L(s)\begin{pmatrix} s^2 & 0 \\ 1 & -1+s \end{pmatrix} + Y_L(s)\begin{pmatrix} 1+s & 0 \\ 1 & -1 \end{pmatrix} = \begin{pmatrix} 2+4s+3s^2+s^3 & 0 \\ 0 & (2+s)\,((3-2i)+s)\,((3+2i)+s) \end{pmatrix}$

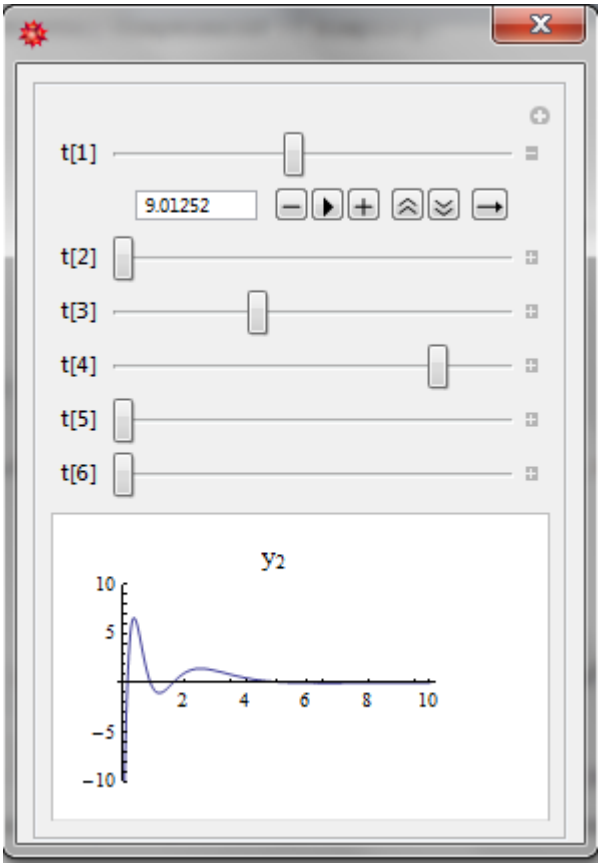
is : $X_L(s) = \begin{pmatrix} 3+s-t[1] & 2-t[1] \\ t[2]+s\,(1+t[2]+t[3]+t[4]) & -1+s^2-t[3]+s\,(9+t[4]) \end{pmatrix}$, $Y_L(s) = \begin{pmatrix} 2+s\,t[1] & -(-1+s)\,(-2+t[1]) \\ 26+s\,t[3]-s^2\,(1+t[2]+t[3]+t[4]) & -25+t[3]+s^2\,t[4]-s\,(35+t[3]+t[4]) \end{pmatrix}$

The class of compensators $C(s) := X_L(s)^{-1} Y_L(s)$ that when employed in a unity feedback loop, result in closed-loop systems S(P,C) with a desired denominator $\begin{pmatrix} 2+4s+3s^2+s^3 & 0 \\ 0 & (2+s)\,((3-2i)+s)\,((3+2i)+s) \end{pmatrix}$ is equal to :

$$\begin{pmatrix} \frac{26t[1]+s^3t[1]-2\,(27+t[3])-s\,(t[1]+2\,(-3+t[3]-t[4]))+s^2\,(-t[1]\,(-3+t[2]+t[3])+2\,(2+t[2]+t[3]+t[4]))}{-3+s^3-2t[2]-3t[3]+t[1]\,(1+t[2]+t[3])+s^2\,(12-t[1]+t[4])+s\,(24-2t[2]-3t[3]+t[1]\,(-3+t[2]+t[3])+t[4])} & -\frac{\left(26+25s+8s^2+s^3\right)(-2+t[1])}{-3+s^3-2t[2]-3t[3]+t[1]\,(1+t[2]+t[3])+s^2\,(12-t[1]+t[4])+s\,(24-2t[2]-3t[3]+t[1]\,(-3+t[2]+t[3])+t[4])} \\ -\frac{2\,(-29+13t[1]+s[2])+s\,((2+t[1])\,t[2]+(-1+t[1])\,t[3]+2\,(-12+t[4]))+s^3\,(1+t[2]+t[3]+t[4])+s^2\,(3+3t[2]+2t[3]+3t[4])}{-3+s^3-2t[2]-3t[3]+t[1]\,(1+t[2]+t[3])+s^2\,(12-t[1]+t[4])+s\,(24-2t[2]-3t[3]+t[1]\,(-3+t[2]+t[3])+t[4])} & \frac{-75+2t[2]+3t[3]-t[1]\,(-25+t[2]+t[3])+s^2\,(-37+2t[2]+3t[3]+t[1]\,(1+t[2]+t[3]))+s\,(-128+26t[1]+t[4])+s^3\,t[4]}{-3+s^3-2t[2]-3t[3]+t[1]\,(1+t[2]+t[3])+s^2\,(12-t[1]+t[4])+s\,(24-2t[2]-3t[3]+t[1]\,(-3+t[2]+t[3])+t[4])} \end{pmatrix}$$

(Figure 10.1)

The plot of the y_2 step response that is displayed is the following :



(Figure 10.2)

CHAPTER 11

A REMARK ON THE COLUMN – PROPER ALGORITHM BY VARDULAKIS – THE COLUMN PROPER ALGORITHM BY W.H.L. NEVEN AND C.PRAAGMAN

The column – proper algorithm that we have mentioned and described was the one informally developed by **Vardulakis**. In the current section , we will point out an example in which the Vardulakis algorithm fails to respond accordingly and does not yield the desired results.

Let $T(s)=\begin{bmatrix} s^3-1 & 1 & s-1 \\ s & 2s^2-2 & s^2+1 \\ s+1 & -2s^2 & -s^2 \end{bmatrix}$. Obviously , $\deg T(s)=5, c_c(T)=7$.

So the matrix is not column – proper.

$\left[T(s)\right]_c^h=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & -2 & -1 \end{bmatrix}$, so we seek a numerical solution of the system :

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & -2 & -1 \end{bmatrix}\cdot\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\Leftrightarrow\begin{bmatrix} a_1 \\ 2a_2+a_3 \\ -2a_2-a_3 \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\Leftrightarrow a_1=0 \text{ and } 2a_2+a_3=0.$$

We pick $a_2=1, a_3=-2$ and we form the column -vector :

$\begin{bmatrix} a_1 s^{3-3} \\ a_2 s^{3-2} \\ a_3 s^{3-2} \end{bmatrix}=\begin{bmatrix} 0 \\ s \\ -2s \end{bmatrix}$, and we consider the unimodular matrix :

$T_R(s)=\begin{bmatrix} 0 & 0 & 0 \\ s & 1 & 0 \\ -2s & 0 & 1 \end{bmatrix}$. However , it holds that $Det\left[T_R(s)\right]=0$, so the matrix that we considered is not unimodular. The determinant of the matrix $T(s)T_R(s)$ will be equal to zero , and hence the degree of the new matrix will also be equal to zero.

This error is countered by **W.H.L Neven** and **C.Praagman.**

Column – Proper Reduction algorithm by **Neven** and **Praagman , [4] & [5]**

After the calculation of the vector y that solves the previous system , we act as follows :

- We find the smallest integer $l \in \mathbb{N}$ such that $w := s^l \left(\Delta^T\right)^{-1} y \in \mathbb{R}^n\left[s\right]$, where $\Delta^T\left(s\right) = diag\left(s^{d_1(T)},\ldots,s^{d_n(T)}\right)$ and $d_j\left(T\right)$ is the degree of the j^{th} column of $T\left(s\right)$. The exponent n denotes the number of columns of the matrix $T\left(s\right)$. In other words , the algorithm can be applied for arbitrary rectangular matrices.
- We pick a j such that $y_j \neq 0$ and $d_j\left(T\right) = l$ and we define the unimodular matrix $U = I_n + \left(w - e_j\right)\cdot e_j^T$, where e_j is an element of the standard orthonormal basis of $\mathbb{R}^n$.
- We calculate the product $T \cdot U$ and we repeat the previous steps until we get a column – proper matrix.

We will now apply this algorithm to the previous example .

$$y=\begin{bmatrix}0\\1\\-2\end{bmatrix}\text{ , so }\quad s^l\left(\Delta^T\right)^{-1}y=s^l\begin{bmatrix}\frac{1}{s^3}&0&0\\0&\frac{1}{s^2}&0\\0&0&\frac{1}{s^2}\end{bmatrix}\cdot\begin{bmatrix}0\\1\\-2\end{bmatrix}=s^l\begin{bmatrix}0\\\frac{1}{s^2}\\\frac{-2}{s^2}\end{bmatrix}=\begin{bmatrix}0\\\frac{s^l}{s^2}\\\frac{-2s^l}{s^2}\end{bmatrix}$$

Clearly , $l=2$ and $w=\begin{bmatrix}0\\1\\-2\end{bmatrix}$. In addition , $y_2\neq 0$. Hence :

$$U=I_3+\left(\begin{bmatrix}0\\1\\-2\end{bmatrix}-\begin{bmatrix}0\\1\\0\end{bmatrix}\right)\cdot\begin{bmatrix}0&1&0\end{bmatrix}=I_3+\begin{bmatrix}0\\0\\-2\end{bmatrix}\cdot\begin{bmatrix}0&1&0\end{bmatrix}=I_3+\begin{bmatrix}0&0&0\\0&0&0\\0&-2&0\end{bmatrix}=\begin{bmatrix}1&0&0\\0&1&0\\0&-2&1\end{bmatrix}$$

The product $T'(s)=T(s)U(s)=\begin{bmatrix}s^3-1&3-2s&s-1\\s&-4&s^2+1\\s+1&0&-s^2\end{bmatrix}$. Notice that the new matrix is not column – proper , but the column – complexity has been reduced to 6 while the degree is equal to 5.

If we apply the algorithm for $T'(s)$ once more , then the new unimodular matrix that arises is $U(s)=\begin{bmatrix}2&0&0\\s^2&1&0\\0&0&1\end{bmatrix}$, while the product $T'(s)U(s)$ is equal to :

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

The final matrix is column – proper and the algorithm can be terminated.

Let us now discuss the previous topic from the Diophantine equations perspective. So , we limit the scope of the column – proper algorithm to square matrices only.

It seems that the Vardulakis algorithm fails when we come up with a zero solution.

This idea , however , is refuted by the following example :

The matrix $T(s)=\begin{bmatrix} s^3 & s+2 & 1 \\ s^2 & s^2 & -2s \\ 2s-2 & -s^2 & 2s-1 \end{bmatrix}$ has a highest column coefficient matrix equal to $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -2 \\ 0 & -1 & 2 \end{bmatrix}$. The Vardulakis algorithm will fail to apply for this example .

On the other hand , the matrix $T(s)=\begin{bmatrix} s^2 & s+2 & 1 \\ s & s^2 & -2s^3 \\ 2s-2 & -s^2 & 2s^3-1 \end{bmatrix}$ has the same highest column coefficient matrix , but the Vardulakis algorithm works properly. All these lead us to :

Conjecture

Let $T(s)\in\mathbb{R}[s]^{p\times p}$ and $x=(x_1,x_2,\ldots,x_p)$ vector such that $[T(s)]_c^h\cdot x=0$. Let also A be a subset of $\{1,2,\ldots,p\}$ such that , $k\in A\Leftrightarrow x_k=0$ and j_0 be the highest column degree of $T(s)$. If $j_0\in A$ then the Vardulakis algorithm cannot be applied for T .

We will now compare the results of the two algorithms.

- $T(s)=\begin{bmatrix} 1 & -s+2 \\ s & s^2-1 \end{bmatrix}(\deg=2)$

Vardulakis algorithm $\rightarrow\begin{bmatrix} 1 & 2(s-1) \\ s & 1 \end{bmatrix}$

Neven & Praagman algorithm $\rightarrow\begin{bmatrix} 1 & 2(s-1) \\ s & 1 \end{bmatrix}$

- $T(s)=\begin{bmatrix} s^2-1 & s^3+3s^2 \\ 1 & s \end{bmatrix}(\deg=2)$

Vardulakis algorithm $\rightarrow\begin{bmatrix} -3-s & -8s \\ 3 & 9s \end{bmatrix}$

Neven & Praagman algorithm $\rightarrow \begin{bmatrix} -1-3s & -3-s \\ 1+3s & 3 \end{bmatrix}$

- $T(s)=\begin{bmatrix} s^7 & s^6+s^5 \\ 1 & s^3 \end{bmatrix}(\deg=10)$

Vardulakis algorithm $\rightarrow \begin{bmatrix} s^5 & -s^5 \\ 1+s^3-s^4 & s-s^3+s^4-s^5 \end{bmatrix}$

Neven & Praagman algorithm $\rightarrow \begin{bmatrix} 0 & s^5 \\ 1+s-s^5 & 1+s^3-s^4 \end{bmatrix}$

We can notice that the algorithm by Neven and Praagman has almost the same behavior as the Vardulakis version.

The program that has been created to solve Diophantine equations allows the value of the variable μ to run from 1 to 20. As demonstrated by the following example , this range can be easily violated , due to the fact that the complexity increases dramatically while the dimensions of the inserted plant get bigger.

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

For the previous example it holds that $\mu > 20$.

Concluding this section , the Vardulakis algorithm does not work in some complicated cases , but when it does, it yields very convenient and handy results.

Of course , the Neven & Praagman algorithm has universal validity.

CHAPTER 12

APPENDIX – MATHEMATICA PROGRAM CODES

Division code

```
n=Input["Insert numerator matrix in the form
{{p11(s),p12(s),...},{p21(s),p22(s),...}....}"];
d=Input["Insert denominator matrix in the form
{{p11(s),p12(s),...},{p21(s),p22(s),...}....}"];
dim=Dimensions[n];
p=dim[[1]];
m=dim[[2]];
dimen=Dimensions[d];
w=dimen[[1]];
z=dimen[[2]];

If[w==z∧z==m,



  If[Exponent[Det[d],s]≠-∞,
   tright=Dot[n,Inverse[d]]//Simplify;
   For[i=1,i≤p,++i,
    For[j=1,j≤m,++j,
      tright[[i,j]]=Together[tright[[i,j]]];
      ];
    ];
   qright=ConstantArray[0,{p,m}];
   xright=ConstantArray[0,{p,m}];
   For[i=1,i≤p,++i,
    For[j=1,j≤m,++j,

If[Exponent[Numerator[tright[[i,j]]],s]≥Exponent[Denomina
tor[tright[[i,j]]],s],

qright[[i,j]]=PolynomialQuotient[Numerator[tright[[i,j]]]
,Denominator[tright[[i,j]]],s];

xright[[i,j]]=PolynomialRemainder[Numerator[tright[[i,j]]
],Denominator[tright[[i,j]]],s]/Denominator[tright[[i,j]]
];
       ];

If[Exponent[Numerator[tright[[i,j]]],s]<Exponent[Denomina
tor[tright[[i,j]]],s],
       qright[[i,j]]=0;
       xright[[i,j]]=tright[[i,j]];
       ];
```

```
      If[Exponent[Denominator[tright[[i,j]]],s]==0,
       qright[[i,j]]=tright[[i,j]];
       xright[[i,j]]=0;
       ];
      ];
    ];
   rightremainder=xright.d//Simplify//MatrixForm;
   rightquotient=qright//Simplify//MatrixForm;
   Print["The right quotient of the division is the
matrix ",rightquotient]
    Print["The right remainder of the division is the
matrix  ",rightremainder]
    Print["The right division formula for the given
matrices                                          is
:",n//MatrixForm,"=",rightquotient,"",d//MatrixForm,"+",r
ightremainder]
   ];
  If[Exponent[Det[d],s]==-∞,Print["The determinant of the
denominator matrix must not be equal to zero.Please
insert another matrix."]]
  ];

If[w==z∧z==p,



  If[Exponent[Det[d],s]≠-∞,
   tleft=Dot[Inverse[d],n]//Simplify;
   For[i=1,i≤p,++i,
    For[j=1,j≤m,++j,
      tleft[[i,j]]=Together[tleft[[i,j]]];
      ];
    ];
   qleft=ConstantArray[0,{p,m}];
   xleft=ConstantArray[0,{p,m}];
   For[i=1,i≤p,++i,
    For[j=1,j≤m,++j,

If[Exponent[Numerator[tleft[[i,j]]],s]≥Exponent[Denominat
or[tleft[[i,j]]],s],

qleft[[i,j]]=PolynomialQuotient[Numerator[tleft[[i,j]]],D
enominator[tleft[[i,j]]],s];

xleft[[i,j]]=PolynomialRemainder[Numerator[tleft[[i,j]]],
Denominator[tleft[[i,j]]],s]/Denominator[tleft[[i,j]]];
       ];

If[Exponent[Numerator[tleft[[i,j]]],s]<Exponent[Denominat
or[tleft[[i,j]]],s],
```

```
       qleft[[i,j]]=0;
       xleft[[i,j]]=tleft[[i,j]];
       ];
      If[Exponent[Denominator[tleft[[i,j]]],s]==0,
       qleft[[i,j]]=tleft[[i,j]];
       xleft[[i,j]]=0;
       ];
      ];
    ];
   leftremainder=d.xleft//Simplify//MatrixForm;
   leftquotient=qleft//Simplify//MatrixForm;
   Print["The left quotient of the division is the matrix
",leftquotient]
    Print["The left remainder of the division is the
matrix  ",leftremainder]
    Print["The left division formula for the given
matrices                                              is
:",n//MatrixForm,"=",d//MatrixForm,"",leftquotient,"+",le
ftremainder]
   ];
  If[Exponent[Det[d],s]==-∞,Print["The determinant of the
denominator matrix must not be equal to zero.Please
insert another matrix."]]
  ];

If[w≠zV(z≠m∧z≠p),Print["The denominator matrix needs to


be a square matrix with its dimension equal to the number
of rows or the number of columns
  of the numerator matrix."]]
```

Smith – McMillan Form code

```
a=Input["Insert       matrix       in       the       form
{{n11(s)/d11(s),n12(s)/d12(s),...},{n21(s)/d21(s),n22(s)/
d22(s),...},...}"];
dim=Dimensions[a];
p=dim[[1]];
m=dim[[2]];
r=MatrixRank[a];
d=ConstantArray[0,{1,r}];
i=ConstantArray[0,{1,r}];
smith=ConstantArray[0,{p,m}];
x=1;
For[k=1,k≤p,++k,
```

```
  For[j=1,j≤m,++j,
    lcm=PolynomialLCM[Denominator[a[[k,j]]],x];
    x=lcm;
    ];
  ];
leastcommonmultiple=lcm;
b=leastcommonmultiple*a//Simplify;
For[k=1,k≤r,++k,
  mat=Minors[b,k];
  dim=Dimensions[mat];
  v=dim[[1]];
  l=dim[[2]];
  x=0;
  For[j=1,j≤v,++j,
   For[t=1,t≤l,++t,
     gcd=PolynomialGCD[mat[[j,t]],x];
     x=gcd;
     ];
   ];
  greatestcommondivisor=gcd;
  d[[1,k]]=greatestcommondivisor;
  ];
For[k=2,k≤r,++k,
  i[[1,k]]=d[[1,k]]/d[[1,k-1]];
  ];
i[[1,1]]=d[[1,1]];
For[k=1,k≤r,++k,
  smith[[k,k]]=i[[1,k]];
  ];
finalform=1/leastcommonmultiple*smith//Simplify;
smithmcmillan=ConstantArray[0,{p,m}];
For[i=1,i≤r,++i,
  smithmcmillan[[i,i]]=Factor[finalform[[i,i]]];
  ];
Print["The Smith-McMillan form of the
matrix",a//MatrixForm," is the matrix
",smithmcmillan//MatrixForm]
```

Smith – Decomposition code

```
d=Input["Insert matrix in the form
{{p11(s),p12(s),...},{p21(s),p22(s),...},...}}"];
dim=Dimensions[d];
p=dim[[1]];
m=dim[[2]];
pinitial=p;
minitial=m;
```

```
a=d;
aleft=Array[f,50];
aright=Array[g,50];
For[i=1,i≤50,++i,
  aleft[[i]]=IdentityMatrix[p];
  aright[[i]]=IdentityMatrix[m];
  ];
column=0;
row=0;
condition=0;
count=0;
While[condition<1,
  firstcondition=0;
  deg=1000;
  For[i=1,i≤p,++i,
   For[j=1,j≤m,++j,

     If[Exponent[a[[i,j]],s]≠-∞∧Exponent[a[[i,j]],s]<deg,



       x=i;
       y=j;
       deg=Exponent[a[[i,j]],s];
       ];
     ];
   ];
  help=a;
  a[[All,1]]=help[[All,y]];
  a[[All,y]]=help[[All,1]];
  help=a;
  a[[1]]=help[[x]];
  a[[x]]=help[[1]];
  help=row;
  row=help+1;
  help=aleft[[row]];
  aleft[[row]][[1+count]]=help[[x+count]];
  aleft[[row]][[x+count]]=help[[1+count]];
  help=column;
  column=help+1;
  help=aright[[column]];
  aright[[column]][[All,1+count]]=help[[All,y+count]];
  aright[[column]][[All,y+count]]=help[[All,1+count]];
  While[firstcondition<1,
   k=2;
   While[k≤m,
    q=PolynomialQuotient[a[[1,k]],a[[1,1]],s];
    r=PolynomialRemainder[a[[1,k]],a[[1,1]],s];
    While[Exponent[r,s]≠-∞,
     help=a;
```

```
      a[[All,k]]=help[[All,k]]-q*help[[All,1]];
      help=column;
      column=help+1;
      help=aright[[column]];
      aright[[column]][[All,k+count]]=help[[All,k+count]]-
q*help[[All,1+count]];
      help=a;
      a[[All,1]]=help[[All,k]];
      a[[All,k]]=help[[All,1]];
      help=column;
      column=help+1;
      help=aright[[column]];
      aright[[column]][[All,1+count]]=help[[All,k+count]];
      aright[[column]][[All,k+count]]=help[[All,1+count]];
      q=PolynomialQuotient[a[[1,k]],a[[1,1]],s];
      r=PolynomialRemainder[a[[1,k]],a[[1,1]],s];
      ];
     help=k;
     k=help+1;
     ];
    For[k=2,k≤m,++k,
     q=PolynomialQuotient[a[[1,k]],a[[1,1]],s];
     r=PolynomialRemainder[a[[1,k]],a[[1,1]],s];
     help=a;
     a[[All,k]]=help[[All,k]]-q*help[[All,1]];
     help=column;
     column=help+1;
     help=aright[[column]];
     aright[[column]][[All,k+count]]=help[[All,k+count]]-
q*help[[All,1+count]];
     ];
    k=2;
    While[k≤p,
     q=PolynomialQuotient[a[[k,1]],a[[1,1]],s];
     r=PolynomialRemainder[a[[k,1]],a[[1,1]],s];
     While[Exponent[r,s]≠-∞,
      help=a;
      a[[k]]=help[[k]]-q*help[[1]];
      help=row;
      row=help+1;
      help=aleft[[row]];
      aleft[[row]][[k+count]]=help[[k+count]]-
q*help[[1+count]];
      help=a;
      a[[1]]=help[[k]];
      a[[k]]=help[[1]];
      help=row;
      row=help+1;
      help=aleft[[row]];
      aleft[[row]][[1+count]]=help[[k+count]];
      aleft[[row]][[k+count]]=help[[1+count]];
```

```
     q=PolynomialQuotient[a[[k,1]],a[[1,1]],s];
     r=PolynomialRemainder[a[[k,1]],a[[1,1]],s];
     ];
    help=k;
    k=help+1;
    ];
   For[k=2,k≤p,++k,
    q=PolynomialQuotient[a[[k,1]],a[[1,1]],s];
    r=PolynomialRemainder[a[[k,1]],a[[1,1]],s];
    help=a;
    a[[k]]=help[[k]]-q*help[[1]];
    help=row;
    row=help+1;
    help=aleft[[row]];
    aleft[[row]][[k+count]]=help[[k+count]]-
q*help[[1+count]];
    ];
   zerocount=0;
   For[i=2,i≤p,++i,
    For[j=2,j≤m,++j,
      r=PolynomialRemainder[a[[i,j]],a[[1,1]],s];
      If[Exponent[r,s]==-∞,
       help=zerocount;
       zerocount=help+1;
       ];
      If[Exponent[r,s]≠-∞,
       v=j;
       ];
      ];
    ];
   If[zerocount==(p-1)*(m-1),
    firstcondition=1;
    ];
   If[zerocount<(p-1)*(m-1),
    help=a;
    a[[All,1]]=help[[All,1]]+help[[All,v]];
    help=column;
    column=help+1;
    help=aright[[column]];

aright[[column]][[All,1+count]]=help[[All,1+count]]+help[
[All,v+count]];
    ];
   ];
  help=count;
  count=help+1;
  sub=Take[a,{2,p},{2,m}];
  dim=Dimensions[sub];
  p=dim[[1]];
  m=dim[[2]];
  a=sub;
```

```
  If[p==1∧m==1,



   condition=1;
   ];

  If[p==1∧m≠1,



   k=2;
   While[k≤m,
    q=PolynomialQuotient[a[[1,k]],a[[1,1]],s];
    r=PolynomialRemainder[a[[1,k]],a[[1,1]],s];
    While[Exponent[r,s]≠-∞,
     help=a;
     a[[All,k]]=help[[All,k]]-q*help[[All,1]];
     help=column;
     column=help+1;
     help=aright[[column]];
     aright[[column]][[All,k+count]]=help[[All,k+count]]-
q*help[[All,1+count]];
     help=a;
     a[[All,1]]=help[[All,k]];
     a[[All,k]]=help[[All,1]];
     help=column;
     column=help+1;
     help=aright[[column]];
     aright[[column]][[All,1+count]]=help[[All,k+count]];
     aright[[column]][[All,k+count]]=help[[All,1+count]];
     q=PolynomialQuotient[a[[1,k]],a[[1,1]],s];
     r=PolynomialRemainder[a[[1,k]],a[[1,1]],s];
     ];
    help=k;
    k=help+1;
    ];
   For[k=2,k≤m,++k,
    q=PolynomialQuotient[a[[1,k]],a[[1,1]],s];
    r=PolynomialRemainder[a[[1,k]],a[[1,1]],s];
    help=a;
    a[[All,k]]=help[[All,k]]-q*help[[All,1]];
    help=column;
    column=help+1;
    help=aright[[column]];
    aright[[column]][[All,k+count]]=help[[All,k+count]]-
q*help[[All,1+count]];
    ];
   condition=1;
```

```
   ];

  If[p≠1∧m==1,



   k=2;
   While[k≤p,
    q=PolynomialQuotient[a[[k,1]],a[[1,1]],s];
    r=PolynomialRemainder[a[[k,1]],a[[1,1]],s];
    While[Exponent[r,s]≠-∞,
     help=a;
     a[[k]]=help[[k]]-q*help[[1]];
     help=row;
     row=help+1;
     help=aleft[[row]];
     aleft[[row]][[k+count]]=help[[k+count]]-
q*help[[1+count]];
     help=a;
     a[[1]]=help[[k]];
     a[[k]]=help[[1]];
     help=row;
     row=help+1;
     help=aleft[[row]];
     aleft[[row]][[1+count]]=help[[k+count]];
     aleft[[row]][[k+count]]=help[[1+count]];
     q=PolynomialQuotient[a[[k,1]],a[[1,1]],s];
     r=PolynomialRemainder[a[[k,1]],a[[1,1]],s];
     ];
    help=k;
    k=help+1;
    ];
   For[k=2,k≤p,++k,
    q=PolynomialQuotient[a[[k,1]],a[[1,1]],s];
    r=PolynomialRemainder[a[[k,1]],a[[1,1]],s];
    help=a;
    a[[k]]=help[[k]]-q*help[[1]];
    help=row;
    row=help+1;
    help=aleft[[row]];
    aleft[[row]][[k+count]]=help[[k+count]]-
q*help[[1+count]];
    ];
   condition=1;
   ];
  ];
help=IdentityMatrix[pinitial];
For[i=0,i≤49,++i,
  leftunimodular=help.aleft[[50-i]];
  help=leftunimodular;
```

```
  ];
help=IdentityMatrix[minitial];
For[i=1,i≤50,++i,
  rightunimodular=help.aright[[i]];
  help=rightunimodular;
  ];
smith=leftunimodular.d.rightunimodular;
For[i=1,i≤pinitial,++i,
  For[j=1,j≤minitial,++j,
    If[i==j,
      If[Exponent[smith[[i,i]],s]≠-∞,
        help=row;
        row=help+1;
        help=aleft[[row]];

aleft[[row]][[i]]=1/Coefficient[smith[[i,i]],s,Exponent[s
mith[[i,i]],s]]*help[[i]];
        ];
      ];
    ];
  ];
help=IdentityMatrix[pinitial];
For[i=0,i≤49,++i,
  leftunimodular=help.aleft[[50-i]];
  help=leftunimodular;
  ];
smith=leftunimodular.d.rightunimodular//Simplify//MatrixF
orm;
Print["The Smith form of the polynomial matrix
",d//MatrixForm,"is the matrix
",smith,"=",leftunimodular//Simplify//MatrixForm,"",d//MatrixF
orm,"",rightunimodular//Simplify//MatrixForm]
```

Greatest Common Left Divisor code

```
t1=Input["Insert first matrix in the form
{{p11(s),p12(s),...},{p21(s),p22(s),...},...}}"];
t2=Input["Insert second matrix in the form
{{p11(s),p12(s),...},{p21(s),p22(s),...},...}}"];
dim1=Dimensions[t1];
dim2=Dimensions[t2];
p=dim1[[1]];
l=dim1[[2]];
v=dim2[[2]];
m=l+v;
```

```
t=ConstantArray[0,{p,m}];
For[i=1,i≤p,++i,
  For[j=1,j≤l,++j,
    t[[i,j]]=t1[[i,j]];
    ];
  ];
For[i=1,i≤p,++i,
  For[j=1,j≤v,++j,
    t[[i,l+j]]=t2[[i,j]];
    ];
  ];
```

We then use the Smith – Decomposition code , adding the part :

```
product=d.rightunimodular//Simplify;
leftgcd=ConstantArray[0,{pinitial,pinitial}];
For[i=1,i≤pinitial,++i,
  For[j=1,j≤pinitial,++j,
    leftgcd[[i,j]]=product[[i,j]];
    ];
  ];
Print["The greatest common left divisor of the matrices
",t1//MatrixForm,",",t2//MatrixForm,"is the matrix
",leftgcd//MatrixForm]
```

Greatest Common Right Divisor code

```
t1=Input["Insert first matrix in the form
{{p11(s),p12(s),...},{p21(s),p22(s),...},...}}"];
t2=Input["Insert second matrix in the form
{{p11(s),p12(s),...},{p21(s),p22(s),...},...}}"];
dim1=Dimensions[t1];
dim2=Dimensions[t2];
m=dim1[[2]];
l=dim1[[1]];
v=dim2[[1]];
p=l+v;
t=ConstantArray[0,{p,m}];
For[i=1,i≤l,++i,
  For[j=1,j≤m,++j,
    t[[i,j]]=t1[[i,j]];
    ];
  ];
For[i=1,i≤v,++i,
  For[j=1,j≤m,++j,
```

```
    t[[l+i,j]]=t2[[i,j]];
    ];
  ];
```

Again , we apply the Smith – Decomposition algorithm , followed by :

```
product=leftunimodular.d//Simplify;
rightgcd=ConstantArray[0,{minitial,minitial}];
For[i=1,i≤minitial,++i,
  For[j=1,j≤minitial,++j,
    rightgcd[[i,j]]=product[[i,j]];
    ];
  ];
Print["The greatest common right divisor of the matrices
",t1//MatrixForm,",",t2//MatrixForm,"is the matrix
",rightgcd//MatrixForm]
```

Row – Proper Reduction code

```
giv=Input["Insert matrix in the form
{{p11(s),p12(s),...},{p21(s),p22(s),...},...}}"];
t=giv;
dim=Dimensions[t];
p=dim[[1]];
m=dim[[2]];
highestrowdegree=Array[f,p];
use=Array[z,20];
For[i=1,i≤20,++i,
  use[[i]]=IdentityMatrix[p];
  ];
count=0;
condition=0;
While[condition<1,
  For[i=1,i≤p,++i,
   max=0;
   For[j=1,j≤m,++j,
    highestrowdegree[[i]]=Max[max,Exponent[t[[i,j]],s]];
    max=highestrowdegree[[i]];
    ];
   ];
  min=Min[p,m];
  mat=Minors[t,min];
  dim=Dimensions[mat];
  v=dim[[1]];
  k=dim[[2]];
```

```
  max=0;
  For[j=1,j≤v,++j,
   For[i=1,i≤k,++i,
     degree=Max[max,Exponent[mat[[j,i]],s]];
     max=degree;
     ];
   ];
  sum=0;
  For[i=1,i≤p,++i,
   rowcomplexity=sum+highestrowdegree[[i]];
   sum=rowcomplexity;
   ];
  If[rowcomplexity==degree,
   condition=1;
   proper=t;
   ];
  If[rowcomplexity>degree,
   highestrowcoef=ConstantArray[0,{p,m}];
   For[i=1,i≤p,++i,
    For[j=1,j≤m,++j,

highestrowcoef[[i,j]]=Coefficient[t[[i,j]],s,highestrowde
gree[[i]]];
      ];
    ];
   a=ConstantArray[0,{1,p}];
   For[i=1,i≤p,++i,
    a[[1,i]]=w[i];
    ];
   product=a.highestrowcoef//Simplify;
   zero=ConstantArray[0,{1,m}];
   sol=NSolve[{product==zero},Integers];
   dim=Dimensions[sol];
   For[i=1,i≤dim[[2]],++i,
    For[j=1,j≤p,++j,
      If[sol[[1,i,1]]==w[j],
        a[[1,j]]=sol[[1,i,2]];
        ];
      ];
    ];
   For[i=1,i≤p,++i,
    help=a/.C[i]→1;
    a=help;
    ];
   max=0;
   For[i=1,i≤p,++i,
    If[highestrowdegree[[i]]>max,
      r0=highestrowdegree[[i]];
      x=i;
      max=highestrowdegree[[i]];
      ];
```

```
    ];
   row=ConstantArray[0,{1,p}];
   For[i=1,i≤p,++i,
    row[[1,i]]=a[[1,i]]*s^(r0-highestrowdegree[[i]]);
    ];
   help=IdentityMatrix[p];
   help[[x]]=row[[1]];
   matrix=help;
   proper=matrix.t//Simplify;
   count=count+1;
   use[[count]]=matrix;
   t=proper;
   Clear[w];
   ];
  ];
help=IdentityMatrix[p];
For[i=0,i≤19,++i,
  mtr=help.use[[20-i]]//Simplify;
  help=mtr;
  ];
Print["The left multiplication of the given matrix
",giv//MatrixForm,"with the matrix
",mtr//MatrixForm,"yields the matrix
",proper//MatrixForm,"which is row-proper"]
```

Column – Proper Reduction code

```
giv=Input["Insert matrix in the form
{{p11(s),p12(s),...},{p21(s),p22(s),...},...}}"];
t=giv;
dim=Dimensions[t];
p=dim[[1]];
m=dim[[2]];
highestcolumndegree=Array[f,m];
use=Array[z,20];
For[i=1,i≤20,++i,
  use[[i]]=IdentityMatrix[m];
  ];
count=0;
condition=0;
While[condition<1,
  For[i=1,i≤m,++i,
   max=0;
   For[j=1,j≤p,++j,
```

```
highestcolumndegree[[i]]=Max[max,Exponent[t[[j,i]],s]];
    max=highestcolumndegree[[i]];
    ];
   ];
  min=Min[p,m];
  mat=Minors[t,min];
  dim=Dimensions[mat];
  v=dim[[1]];
  k=dim[[2]];
  max=0;
  For[j=1,j≤v,++j,
   For[i=1,i≤k,++i,
     degree=Max[max,Exponent[mat[[j,i]],s]];
     max=degree;
     ];
   ];
  sum=0;
  For[i=1,i≤m,++i,
   columncomplexity=sum+highestcolumndegree[[i]];
   sum=columncomplexity;
   ];
  If[columncomplexity==degree,
   condition=1;
   proper=t;
   ];
  If[columncomplexity>degree,
   highestcolumncoef=ConstantArray[0,{p,m}];
   For[i=1,i≤p,++i,
    For[j=1,j≤m,++j,

highestcolumncoef[[i,j]]=Coefficient[t[[i,j]],s,highestco
lumndegree[[j]]];
      ];
    ];
   a=ConstantArray[0,{m,1}];
   For[i=1,i≤m,++i,
    a[[i,1]]=w[i];
    ];
   product=highestcolumncoef.a//Simplify;
   zero=ConstantArray[0,{p,1}];
   sol=NSolve[{product==zero},Integers];
   dim=Dimensions[sol];
   For[i=1,i≤dim[[2]],++i,
    For[j=1,j≤m,++j,
      If[sol[[1,i,1]]==w[j],
        a[[j,1]]=sol[[1,i,2]];
        ];
      ];
    ];
   For[i=1,i≤m,++i,
```

```
    help=a/.C[i]→1;
    a=help;
    ];
   max=0;
   For[i=1,i≤m,++i,
    If[highestcolumndegree[[i]]>max,
      r0=highestcolumndegree[[i]];
      x=i;
      max=highestcolumndegree[[i]];
      ];
    ];
   column=ConstantArray[0,{m,1}];
   For[i=1,i≤m,++i,
    column[[i,1]]=a[[i,1]]*s^(r0-
highestcolumndegree[[i]]);
    ];
   help=IdentityMatrix[m];
   help[[All,x]]=column[[All,1]];
   matrix=help;
   proper=t.matrix//Simplify;
   count=count+1;
   use[[count]]=matrix;
   t=proper;
   Clear[w];
   ];
  ];
help=IdentityMatrix[m];
For[i=1,i≤20,++i,
  mtr=help.use[[i]]//Simplify;
  help=mtr;
  ];
Print["The right multiplication of the given matrix
",giv//MatrixForm,"with the matrix
",mtr//MatrixForm,"yields the matrix
",proper//MatrixForm,"which is column-proper"]
```

Left Matrix Fraction Description code

```
given=Input["Insert matrix in the form
{{n11(s)/d11(s),n12(s)/d12(s),...},{n21(s)/d21(s),n22(s)/
d22(s),...},...}}"];
a=given;
dim=Dimensions[a];
p=dim[[1]];
m=dim[[2]];
```

```
x=1;
For[k=1,k≤p,++k,
  For[j=1,j≤m,++j,
    lcm=PolynomialLCM[Denominator[a[[k,j]]],x];
    x=lcm;
    ];
  ];
leastcommonmultiple=lcm;
nleft=leastcommonmultiple*a//Simplify;
dleft=ConstantArray[0,{p,p}];
For[i=1,i≤p,++i,
  dleft[[i,i]]=leastcommonmultiple;
  ];
```

We then use the greatest common left divisor code for the matrices dleft , nleft followed by:

```
If[Exponent[Det[leftgcd], s] != 0,

 nleft = Inverse[leftgcd].nleft // Simplify;

 dleft = Inverse[leftgcd].dleft // Simplify;

 ];
```

We apply the row – proper algorithm for the matrix dleft , and finally :

```
dleft=mtr.dleft//Simplify;
nleft=mtr.nleft//Simplify;
Print["The left fraction description of the matrix
",given//MatrixForm," is
",dleft//MatrixForm,"",nleft//MatrixForm]
```

Right Matrix Fraction Description code

```
given=Input["Insert matrix in the form
{{n11(s)/d11(s),n12(s)/d12(s),...},{n21(s)/d21(s),n22(s)/
d22(s),...},...}}"];
a=given;
dim=Dimensions[a];
p=dim[[1]];
m=dim[[2]];
x=1;
For[k=1,k≤p,++k,
  For[j=1,j≤m,++j,
    lcm=PolynomialLCM[Denominator[a[[k,j]]],x];
    x=lcm;
    ];
  ];
leastcommonmultiple=lcm;
nright=leastcommonmultiple*a//Simplify;
dright=ConstantArray[0,{m,m}];
For[i=1,i≤m,++i,
  dright[[i,i]]=leastcommonmultiple;
  ];
```

We then use the greatest common right divisor code for the matrices dright , nright.

If[Exponent[Det[rightgcd],s]!=0,

dright=dright.Inverse[rightgcd]//Simplify;

nright=nright.Inverse[rightgcd]//Simplify;

];

And finally , after the matrix dright has been reduced to a column – proper one :

```
nright=nright.mtr//Simplify;
dright=dright.mtr//Simplify;
Print["The right fraction description of the matrix
",given//MatrixForm," is
",nright//MatrixForm,"",dright//MatrixForm,""]
```

Solution of polynomial matrix Diophantine equations code

The following part of the code enables the user to insert the matrix through a simple and elementary , dynamic user interface. This code is used in all the user interfaces that we have designed.

```
row1=Input["Insert number of rows"];
column1=Input["Insert number of columns"];
return=0;
condition=0;
count=0;
While[condition<1,
  If[count<1,

DynamicModule[{x=ConstantArray[0,{row1,column1}],a},a={Co
lumn[(Row[#]&/@Table[With[{i=i,j=j},InputField[Dynamic[x[
[i,j]]],FieldSize→Medium]],{i,1,row1},{j,1,column1}])]};

AppendTo[a,{Text[Plant],CancelButton[],DefaultButton[Dial
ogReturn[return1=x]]}];

CreateDialog[Column[Flatten@{a},Spacings→{1,Automatic},Al
ignment→Left],Modal→True]];
   count=count+1;
   ];
  If[MatrixQ[return1],condition=1];
  ];
```

We can now use the matrix through the use of the variable *return1.*

We find a right fraction description of the matrix *return1* and we apply the algorithm that we described in chapter 10.

```
dim=Dimensions[return1];
p=dim[[1]];
m=dim[[2]];
ef=ConstantArray[0,{m+p,m}];
For[i=1,i≤m,++i,
  For[j=1,j≤m,++j,
    ef[[i,j]]=dright[[i,j]];
    ];
  ];
For[i=1,i≤p,++i,
  For[j=1,j≤m,++j,
    ef[[m+i,j]]=nright[[i,j]];
    ];
  ];
```

```
efcolumns=Array[h,m];
For[i=1,i≤m,++i,
  efcolumns[[i]]=ef[[All,i]];
  ];
Clear[g];
kapa=Array[g,m];
For[i=1,i≤m,++i,
  max=0;
  For[j=1,j≤m+p,++j,
   If[Exponent[efcolumns[[i]][[j]],s]≥max,
     kapa[[i]]=Exponent[efcolumns[[i]][[j]],s];
     max=Exponent[efcolumns[[i]][[j]],s];
     ];
   ];
  ];
Clear[w];
sos=Array[w,m];
For[i=1,i≤m,++i,
  sos[[i]]=ConstantArray[0,{m+p,kapa[[i]]+1}];
  ];
Clear[k];
k=1;
While[k≤m,
  j=0;
  While[j≤kapa[[k]],
   i=1;
   While[i≤m+p,
    If[Exponent[efcolumns[[k]][[i]],s]≠-∞,

sos[[k]][[i,j+1]]=Coefficient[efcolumns[[k]][[i]],s,j];
     ];
    If[Exponent[efcolumns[[k]][[i]],s]==-∞,
     sos[[k]][[i,j+1]]=0;
     ];
    i=i+1;
    ];
   j=j+1;
   ];
  k=k+1;
  ];
sylvester=Array[o,m];
For[i=1,i≤m,++i,
  sylvester[[i]]=Array[c,20];
  ];
For[i=1,i≤m,++i,
  For[k=1,k≤20,++k,

sylvester[[i,k]]=ConstantArray[0,{k(m+p),kapa[[i]]+k}];
    ];
  ];
For[i=1,i≤m,++i,
```

```
  For[k=1,k≤20,++k,
    var=0;
    While[var≤k-1,
     q=1;
     While[q≤m+p,
      r=1;
      While[r≤kapa[[i]]+1,

sylvester[[i,k]][[var(m+p)+q,var+r]]=sos[[i]][[q,r]];
       r=r+1;
       ];
      q=q+1;
      ];
     var=var+1;
     ];
    ];
  ];
help=0;
For[i=1,i≤m,++i,
  sum=help+kapa[[i]];
  help=sum;
  ];
Clear[v]
wolovich=Array[v,20];
For[k=1,k≤20,++k,
  wolovich[[k]]=ConstantArray[0,{k(m+p),m*k+sum}];
  ];
For[k=1,k≤20,++k,
  var=0;
  For[i=1,i≤m,++i,
   q=1;
   While[q≤k(m+p),
    r=1;
    While[r≤kapa[[i]]+k,
     wolovich[[k]][[q,var+r]]=sylvester[[i,k]][[q,r]];
     r=r+1;
     ];
    q=q+1;
    ];
   help=var+kapa[[i]]+k;
   var=help;
   ];
  ];
k=1;
condition=0;
While[condition<1,
  temp=Dimensions[wolovich[[k]]];
  rank=MatrixRank[wolovich[[k]]];
  If[rank==temp[[2]],
   mi=k;
   condition=1;
```

```
    ];
  k=k+1;
  ];
ksi=ConstantArray[0,m];
For[i=1,i≤m,++i,
  If[OddQ[i],
   ksi[[i]]=mi-1;
   ];
  If[EvenQ[i],
   ksi[[i]]=mi;
   ];
  ];
resultant=wolovich[[mi+1]];
dc=ConstantArray[0,{m,m}];
For[i=1,i≤m,++i,
  what=highestcolumndegree[[i]]+ksi[[i]];
  return=0;
  condition=0;
  count=0;
  While[condition<1,
   If[count<1,

DynamicModule[{x=ConstantArray[0,{1,what}],a},a={Column[(
Row[#]&/@Table[With[{i=i,j=j},InputField[Dynamic[x[[i,j]]
],FieldSize→Medium]],{i,1,1},{j,1,what}])]};
     AppendTo[a,{Text[Desired
Zeros],CancelButton[],DefaultButton[DialogReturn[return=x
]]}];

CreateDialog[Column[Flatten@{a},Spacings→{1,Automatic},Al
ignment→Left],Modal→True]];
    count=count+1;
    ];
   If[MatrixQ[return],condition=1];
   ];
  help=1;
  For[g=1,g≤highestcolumndegree[[i]]+ksi[[i]],++g,
   prod=(s-return[[1,g]])*help;
   help=prod;
   ];
  dc[[i,i]]=prod//Simplify;
  ];
dbar=ConstantArray[0,{m,columncomplexity+m*(mi+1)}];
var=0;
For[i=1,i≤m,++i,
  j=0;
  While[j≤Exponent[dc[[i,i]],s],
   dbar[[i,var+j+i]]=Coefficient[dc[[i,i]],s,j];
   j=j+1;
   ];
  help=var+1+Exponent[dc[[i,i]],s];
```

```
  var=help;
  ];
dim1=Dimensions[resultant];
dim2=Dimensions[dbar];
compound=ConstantArray[0,{dim1[[1]]+dim2[[1]],dim1[[2]]}]
;
For[i=1,i≤dim1[[1]],++i,
  For[j=1,j≤dim1[[2]],++j,
    compound[[i,j]]=resultant[[i,j]];
    ];
  ];
For[i=1,i≤dim2[[1]],++i,
  For[j=1,j≤dim2[[2]],++j,
    compound[[dim1[[1]]+i,j]]=dbar[[i,j]];
    ];
  ];
help=RowReduce[Transpose[compound]];
compound=Transpose[help];
Clear[o];
omegabar=Array[o,m];
For[i=1,i≤m,++i,
  sub=Take[compound,{1,(p+m)(ksi[[i]]+1)},{1,dim1[[2]]-
m*(mi-ksi[[i]])}];

res=Take[compound,{dim1[[1]]+i,dim1[[1]]+i},{1,dim1[[2]]-
m*(mi-ksi[[i]])}];
  dim=Dimensions[sub];
  helpus=dim[[1]];
  Clear[q];
  omegabar[[i]]=ConstantArray[0,{1,helpus}];
  For[z=1,z≤helpus,++z,
   omegabar[[i]][[1,z]]=q[z];
   ];
  prod=omegabar[[i]].sub;
  sol=Solve[{prod==res}];
  dimen=Dimensions[sol];
  neartheend=dimen[[2]];
  For[e=1,e≤neartheend,++e,
   For[j=1,j≤helpus,++j,
     If[sol[[1,e,1]]==q[j],
       omegabar[[i]][[1,j]]=sol[[1,e,2]];
       ];
     ];
   ];
  ];
Clear[t];
count=0;
For[i=1,i≤m,++i,
  For[j=1,j≤50,++j,
    If[!FreeQ[omegabar[[i]],q[j]],
      help=omegabar[[i]]/.{q[j]→t[count+1]};
```

```
      omegabar[[i]]=help;
      count=count+1;
      ];
    ];
  ];
Clear[o];
omega=Array[o,m];
For[i=1,i≤m,++i,
  dim=Dimensions[omegabar[[i]]];
  helpsum=0;
  For[j=1,j≤dim[[2]],++j,
   qr=QuotientRemainder[j,p+m];
   If[qr[[2]]==0,
    ending=Take[omegabar[[i]],{1,1},{(qr[[1]]-
1)(p+m)+1,(qr[[1]]-1)(p+m)+p+m}];
    help=ending*s^(qr[[1]]-1);
    sum=helpsum+help;
    helpsum=sum;
    ];
   ];
  omega[[i]]=sum;
  ];
iamomega=ConstantArray[0,{m,p+m}];
For[i=1,i≤m,++i,
  iamomega[[i]]=omega[[i]][[1]];
  ];
xleft=ConstantArray[0,{m,m}];
yleft=ConstantArray[0,{m,p}];
For[i=1,i≤m,++i,
  For[j=1,j≤m,++j,
    xleft[[i,j]]=iamomega[[i,j]];
    ];
  ];
For[i=1,i≤m,++i,
  For[j=1,j≤p,++j,
    yleft[[i,j]]=iamomega[[i,m+j]];
    ];
  ];
compensator=Inverse[xleft].yleft//Simplify;
Print["The solution of the polynomial matrix Diophantine
equation                                                :
","","X_L(s)",dright//MatrixForm,"+Y_L(s)",nright//MatrixFor
m,"=",dc//MatrixForm,"is                                :
X_L(s)=",xleft//Simplify//MatrixForm,",
Y_L(s)=",yleft//Simplify//MatrixForm];
Print["The class of compensators C(s):=X_L(s)^-1 Y_L(s) that
when employed in a unity feedback loop, result in closed-
loop      systems      S(P,C)      with      a      desired
denominator",dc//MatrixForm,"is      equal      to      :
",compensator//MatrixForm];
```

```
transferfunction=return1.Inverse[IdentityMatrix[m]+compen
sator.return1].compensator//Simplify;
condition=0;
ind=Array[p,10];
count=0;
While[condition<1,
  in=Input["Insert number i , where i refers to the
respective y_i step response that you want to
check.Insert number 0 when you are finished."];
  If[in≠0,
   count=count+1;
   ind[[count]]=in;

help:=InverseLaplaceTransform[transferfunction[[in,in]],s
,t];
   CreateDialog[Manipulate[Plot[#,{t,0,10},PlotRange→{-
10,10},PlotLabel→Row@{"y"ind[[count]]}],{t[1],0,20},{t[2],0,2
0},{t[3],0,20},{t[4],0,20},{t[5],0,20},{t[6],0,20}]&@help
]
   ];
  If[in==0,condition=1;]
  ];
```

We have created some new Mathematica functions using all the above codes and the command *Put* .

ACKNOWLEDGEMENTS

Special Thanks goes to Stephen Wolfram and Wolfram Research for developing a powerful computational software program which made this study a pleasant task.

REFERENCES

[1] VARDULAKIS , A.I.G. (1991) *Linear Multivariable Control – Algebraic Analysis and Synthesis Methods.* New York : Wiley

[2] E.N. ANTONIOU and A.I.G. VARDULAKIS (2005) *On the computation and parametrization of proper denominator assigning compensators for strictly proper plants* . IMA Journal of Mathematical Control and Information

[3] WOLOVICH , W.A. (1974) *Linear Multivariable Systems.* New York : Springer

[4] TH.G.J. BEELEN , G.J.H.H. VAN DEN HURK and C. PRAAGMAN (1988) *A new method for computing a column reduced polynomial matrix.* Systems and Control Letters.

[5] W.H.L. NEVEN and C.PRAAGMAN (1993) *Column reduction of polynomial matrices. Linear Algebra and its applications.* Special issue on Numerical Linear Algebra Methods for Control , Signals and Systems.