Ricardo Zavala Yoe

Modelling and Control of Dynamical Systems: Numerical Implementation in a Behavioral Framework

Studies in Computational Intelligence, Volume 124

Editor-in-chief
Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Further volumes of this series can be found on our homepage: springer.com

Vol. 104. Maria Virvou and Lakhmi C. Jain (Eds.)
Intelligent Interactive Systems in Knowledge-Based Environment, 2008
ISBN 978-3-540-77470-9

Vol. 105. Wolfgang Guenthner
Enhancing Cognitive Assistance Systems with Inertial Measurement Units, 2008
ISBN 978-3-540-76996-5

Vol. 106. Jacqueline Jarvis, Dennis Jarvis, Ralph Rönnquist and Lakhmi C. Jain (Eds.)
Holonic Execution: A BDI Approach, 2008
ISBN 978-3-540-77478-5

Vol. 107. Margarita Sordo, Sachin Vaidya and Lakhmi C. Jain (Eds.)
Advanced Computational Intelligence Paradigms in Healthcare - 3, 2008
ISBN 978-3-540-77661-1

Vol. 108. Vito Trianni
Evolutionary Swarm Robotics, 2008
ISBN 978-3-540-77611-6

Vol. 109. Panagiotis Chountas, Ilias Petrounias and Janusz Kacprzyk (Eds.)
Intelligent Techniques and Tools for Novel System Architectures, 2008
ISBN 978-3-540-77621-5

Vol. 110. Makoto Yokoo, Takayuki Ito, Minjie Zhang, Juhnyoung Lee and Tokuro Matsuo (Eds.)
Electronic Commerce, 2008
ISBN 978-3-540-77808-0

Vol. 111. David Elmakias (Ed.)
New Computational Methods in Power System Reliability, 2008
ISBN 978-3-540-77810-3

Vol. 112. Edgar N. Sanchez, Alma Y. Alanís and Alexander G. Loukianov
Discrete-Time High Order Neural Control: Trained with Kalman Filtering, 2008
ISBN 978-3-540-78288-9

Vol. 113. Gemma Bel-Enguix, M. Dolores Jiménez-López and Carlos Martín-Vide (Eds.)
New Developments in Formal Languages and Applications, 2008
ISBN 978-3-540-78290-2

Vol. 114. Christian Blum, Maria José Blesa Aguilera, Andrea Roli and Michael Sampels (Eds.)
Hybrid Metaheuristics, 2008
ISBN 978-3-540-78294-0

Vol. 115. John Fulcher and Lakhmi C. Jain (Eds.)
Computational Intelligence: A Compendium, 2008
ISBN 978-3-540-78292-6

Vol. 116. Ying Liu, Aixin Sun, Han Tong Loh, Wen Feng Lu and Ee-Peng Lim (Eds.)
Advances of Computational Intelligence in Industrial Systems, 2008
ISBN 978-3-540-78296-4

Vol. 117. Da Ruan, Frank Hardeman and Klaas van der Meer (Eds.)
Intelligent Decision and Policy Making Support Systems, 2008
ISBN 978-3-540-78306-0

Vol. 118. Tsau Young Lin, Ying Xie, Anita Wasilewska and Churn-Jung Liau (Eds.)
Data Mining: Foundations and Practice, 2008
ISBN 978-3-540-78487-6

Vol. 119. Slawomir Wiak, Andrzej Krawczyk and Ivo Dolezel (Eds.)
Intelligent Computer Techniques in Applied Electromagnetics, 2008
ISBN 978-3-540-78489-0

Vol. 120. George A. Tsihrintzis and Lakhmi C. Jain (Eds.)
Multimedia Interactive Services in Intelligent Environments, 2008
ISBN 978-3-540-78491-3

Vol. 121. Nadia Nedjah, Leandro dos Santos Coelho and Luiza de Macedo Mourelle (Eds.)
Quantum Inspired Intelligent Systems, 2008
ISBN 978-3-540-78531-6

Vol. 122. Tomasz G. Smolinski, Mariofanna G. Milanova and Aboul-Ella Hassanien (Eds.)
Applications of Computational Intelligence in Biology, 2008
ISBN 978-3-540-78533-0

Vol. 123. Shuichi Iwata, Yukio Ohsawa, Shusaku Tsumoto, Ning Zhong, Yong Shi and Lorenzo Magnani (Eds.)
Communications and Discoveries from Multidisciplinary Data, 2008
ISBN 978-3-540-78732-7

Vol. 124. Ricardo Zavala Yoe
Modelling and Control of Dynamical Systems: Numerical Implementation in a Behavioral Framework, 2008
ISBN 978-3-540-78734-1

Ricardo Zavala Yoe

Modelling and Control of Dynamical Systems: Numerical Implementation in a Behavioral Framework

With 35 Figures and 2 Tables

Springer

Ricardo Zavala Yoe
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Ciudad de México
Calzada del Puente 222, México 14380
México DF (Mexico City)

ISBN 978-3-540-78734-1 e-ISBN 978-3-540-78735-8
DOI: 10.1007/978-3-540-78735-8

Studies in Computational Intelligence ISSN 1860-949X

Library of Congress Control Number: 2008923732

Cover design: Deblik, Berlin, Germany

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Preface

El que sabe que sabe es un sabio. Síguelo.
El que no sabe que sabe esta dormido. Despiértalo.
El que sabe que no sabe es sencillo. Instrúyelo.
El que no sabe que no sabe es un necio. Apártate de él

(He who knows and knows he knows, he is wise. Follow him.
He who knows and knows not he knows, he is asleep. Wake him.
He who knows not and knows he knows not, he is simple. Teach him.
He who knows not and knows not he knows not, he is a fool. Shun him.)
Proverbio Arabe.
Corrige al sabio y lo harás más sabio, corrige al necio y lo harás tu enemigo.

Sedulo curavi, humanas actiones non ridere, non lugere, neque detestari, sed intelligere.
(I have made a ceaseless effort not to ridicule, not to bewail, not to scorn human actions, but to understand them.)

Benedictus de Spinoza (1632-1677).

At its introduction the Behavioral Approach for systems and control offered a novel point of view: it is a representation free framework. By now is a well developed theory with applications throughout the field. This viewpoint was born out of the necessity to provide more effective and general tools for the modelling and control design process of physical systems appearing in engineering. The main goal of the Behavioral Approach is to develop a suitable mathematical framework for discussing dynamical systems aimed at modelling, analysis, and synthesis. Hence, computer assisted modelling and control considers systems that are:

a. Dynamical.

b. Open.

c. Interconnected.

d. Modular.

With this in mind, we shall consider a *polynomial matrix approach* as a particular case of our general behavioral point of view. In this thesis we develop theory to enrich the behavioral approach. We analyze moreover numerical problems that arise during actual computer implementations of control algorithms. In this sense, we warn about (until now) unknown numerical complications that appear in this kind of implementations but we also propose solutions for this. Proceeding this way, we *reduce* the gap between theory and practice.

Contents

1 Motivating the Behavioral Approach **1**
- 1.1 Suitable Modelling and Control of Systems . . . 1
- 1.2 Paradigms in Modelling . . . 2
  - 1.2.1 Closed dynamical systems . . . 3
  - 1.2.2 Open dynamical systems and the input/output approach . . . 4
  - 1.2.3 More about the input/output approach . . . 9
  - 1.2.4 The behavior of the system is the key . . . 10
  - 1.2.5 Some other frameworks for systems and control . . . 10

2 Behavioral framework **13**
- 2.1 Modelling by Tearing and Zooming . . . 13
  - 2.1.1 Constitutive models . . . 13
- 2.2 Dynamical Systems . . . 19
  - 2.2.1 Linear Differential Systems . . . 19
- 2.3 Latent variables and elimination . . . 21
- 2.4 Equivalent representations of behaviors . . . 23
- 2.5 Observability and detectability . . . 23
- 2.6 Controllability and stabilizability . . . 24
- 2.7 Autonomous behaviors . . . 26
- 2.8 Defining inputs and outputs . . . 27
- 2.9 Controllable part of a behavior . . . 29
- 2.10 Interconnection of dynamical systems . . . 30
  - 2.10.1 Control as interconnection . . . 30

3 Full Interconnection Issues **35**
- 3.1 Implementability . . . 36
  - 3.1.1 Minimal Annihilators of a Polynomial Matrix . . . 37
- 3.2 Stabilization and pole placement by regular full interconnection . . . 45
- 3.3 All regularly implementing controllers . . . 49
- 3.4 All stabilizing controllers . . . 53
- 3.5 Summary . . . 57

4 Partial Interconnection Issues **59**
4.1 Regular implementability by partial interconnection 59
4.2 Pole placement and stabilization by regular partial interconnection 60
4.2.1 Pole placement by regular partial interconnection 60
4.2.2 Stabilization by regular partial interconnection 67
4.3 All regularly implementing controllers: the observable case 71
4.4 All regularly implementing controllers: the nonobservable case 77
4.4.1 Reduction to the case that R_2 has full column rank 78
4.4.2 Reduction to the observable case 79
4.5 All stabilizing controllers 82
4.6 Examples for the nonobservable case 88
4.7 Summary 93

5 Embedding Algorithms **95**
5.1 Problem formulation 95
5.2 Preliminaries 97
5.2.1 Historical overview 97
5.2.2 Notation 100
5.3 Pencils and Matrix Pencils 101
5.3.1 Canonical forms of pencils 101
5.3.2 A little bit deeper into matrix pencils 101
5.4 The state space representation 103
5.5 Embedding for a pencil 105
5.6 Transforming the pencil 106
5.7 The algorithm 107
5.7.1 QR Decompositions 107
5.7.2 Staircase form of $\xi E - A$ 109
5.7.3 Algorithm: Embedding $P(\xi)$ 111
5.8 Summary 113

6 Numerical Implementation **115**
6.1 Introduction 115
6.2 Analysis of an example 115
6.3 The geometry of the orbit of a pencil 117
6.4 Matrix pencils as mathematical relations 120
6.5 Conditioning of the pencil 121
6.6 Modelling polynomially and assessing numerically 128
6.7 Computing the determinant of a polynomial matrix 131
6.8 Concluding remarks 133

7 A new algorithm for embedding problems **135**
7.1 The algorithm 135
7.2 Inside the algorithm 137
7.3 Numerical computation 138
7.4 Examples 138
7.5 Summary 140

Conclusions and further research **141**

Bibliography **143**

Summary **151**

Index **153**

List of Figures

1.1 *Energy transformation in a linear system.* . . . 4
1.2 *Input/Output systems.* . . . 5
1.3 *Thermal system.* . . . 6
1.4 *Is an Input/Output structure always suitable?.* . . . 7
1.5 *Does there exist actual signal flow in dynamical systems?* . . . 8
1.6 *General scheme for interconnection.* . . . 8
1.7 *RC circuit diagram.* . . . 9
1.8 *RC circuit block diagram.* . . . 9
1.9 *The permitted trajectories defines the behavior.* . . . 10

2.1 *A black box model interacts via terminals.* . . . 14
2.2 *Mechanical structure.* . . . 15
2.3 *Translational damper.* . . . 15
2.4 *Translational spring.* . . . 16
2.5 *Mass.* . . . 16
2.6 *Tearing : Labeling modules.* . . . 17
2.7 *Observability.* . . . 24
2.8 *Controllability.* . . . 26
2.9 *Interconnection of systems.* . . . 31
2.10 *Controller interconnection.* . . . 32
2.11 *The hidden behavior.* . . . 33

5.1 *3D storing of E_c and A_c* . . . 112

6.1 *Corruption of controllability.*$Ex_1, d = 7$ . . . 119
6.2 *QR(null([A -E])),*$Ex_1, d = 3$. . . . 120
6.3 $\kappa(P(\lambda)), Ex_1, d = 3$. . . . 121
6.4 $\kappa(\Pi(\lambda)), Ex_1, d = 3$. . . . 122
6.5 $\mathtt{p}(\lambda)), Ex_1, d = 3$. . . . 123
6.6 $\kappa(P(\lambda)), Ex_1, d = 4$. . . . 125
6.7 $\kappa(\Pi(\lambda)), Ex_1, d = 4$. . . . 126
6.8 $\mathtt{p}(\lambda)), Ex_1, d = 4$. . . . 127
6.9 *Mechanical structure* $P(\xi)$. . . . 128
6.10 $\kappa(P(\lambda))$ *(scaled) of the mechanical structure* . . . 129

6.11 *Zooming from below at the pole* $\widehat{\lambda_1}$ *in* $\kappa(P(\lambda))$. 130
6.12 $\kappa(\lambda E - A)$ *associated to the mechanical model* 131
6.13 *Conditioning gain* $p(\lambda)$. 132
6.14 *Number of digits lost due to round off errors in* $\kappa(\Pi(\lambda))$ 133

List of Algorithms

Algorithm 1:
Tests whether ker(K) is implementable by regular full interconnection.
Syntax: `C=behimpl(R,K)`. (Section 3.1.1).

Algorithm 2:
Computation of the controllable part of a given behavior.
Syntax: `[D,R1]=behctrb(R)`. (Section 3.1.1).

Algorithm 3:
Tests whether ker(K) is implementable by regular full interconnection using minimal left annihilators.
Syntax: `C=behimplmla(R,K)`. (Section 3.1.1).

Algorithm 4:
Places poles by means of a suitable controller (full interconnection case).
Syntax: `C=behplace(R,r)`. (Section 3.2).

Algorithm 5:
Finds a stabilizing controller (full interconnection case).
Syntax: `C=behstab(R,r)`. (Section 3.2).

Algorithm 6:
Computes the set of all controllers that regularly implement a given behavior.
Syntax: `W=behallimpl(R,K)`. (Section 3.3).

Algorithm 7:
Computes the set of all controllers that regularly stabilize a given behavior.
Syntax: `[R1,D,C0]=behallstab(R)`. (Section 3.4).

Algorithm 8: Computes a pole placing controller (the partial interconnection case).
Syntax: `C=behplace(R1,R2,r)`. (Section 4.2.1).

Algorithm 9: Finds a stabilizing controller (partial interconnection case).
Syntax: `C=behstab(R1,R2,r)`. (Section 4.2.2).

Algorithm 10: Gives the family of all implementing controllers (partial interconnection, observable case).
Syntax: `[W,F1,V1,V2]=behallimpl(R1,R2,r)`. (Section 4.3).

Algorithm 11: Provides the set of all stabilizing controllers by partial interconnection.
Syntax: `C=behallstab(R1,R2)`. (Section 4.5).

Algorithm 12: Computes a QR decomposition of a matrix.
Syntax: `[R,Q,P]=algorithmQR(A)`. (Section 5.7.1).

Algorithm 13: Computes an RQ decomposition of a matrix.
Syntax: `[R,Q,P]=algorithmRQ(A)`. (Section 5.7.1).

Algorithm 14: Finds a block quasi-triangular form $\xi\widehat{E} - \widehat{A}$ of a pencil $\xi E - A$. (Section 5.7.2).
Syntax: `[Pencilhat,M,N]=staircase(Pencil)`.

Algorithm 15: Embedding a row echelon matrix (via pencil approach).
Syntax: `K=Regular_embedding`. (Section 5.7.3).

Algorithm 16: Embedding a polynomial matrix into a unimodular one (via pencils).
Syntax: `Q=embedding(P)`. (Section 5.7.3).

Algorithm 17: Embedding a polynomial matrix into a unimodular one (via row/column compressions applied directly on the polynomial matrix).
Syntax: `Q=embedding(P)`. (Section 7.1).

Chapter 1

Motivating the Behavioral Approach

Modelling and Control of physical systems are well developed branches of Systems and Control Theory and Control Engineering. Traditionally, these branches of human knowledge have been developed within a paradigmatic perspective of input/output thinking: the way an abstraction is obtained from the physical entity involves the notions of input and output variables, and the ideas of cause and effect[1]. This process is informally referred to as modelling. In this chapter our aim is to reconsider the input/output paradigm in modelling by studying some fundamental questions concerning the modelling of physical systems. After doing this, we hope to understand better why the traditional way of modelling and control of systems may not be the best one. We propose to use ideas from the Behavioral Approach to Systems and Control. This approach does not wish to neglect or forget the paradigmatic input/output approach; rather it offers new possibilities.

1.1 Suitable Modelling and Control of Systems

What is a system? This is the first question we have to answer in order to discuss the topic of this chapter, modelling and control from a different point of view: the Behavioral Approach. As a matter of fact, etymology teaches us that the word *system* comes from late Latin *systemat (systema)*, from Greek *systemat-, systema*, from *synistanai:* to combine, from *syn (together) + histanai (to cause, to stand): A system is a regularly interacting or interdependent group of items forming a unified whole.* For us, if those items are physical elements, then we talk about a *physical system.*
Intuitively, when we face a phenomenon we observe a series of events, called the outcomes.

[1]However we have to reflect about the following. Although modelling methods do not impose a cause - effect make up in the resulting equations, when the to be modeled system is considered linear, we can compute the associated impedance matrix which induces an input/output structure of the model [83]. Naturally this is not the case if we consider our system as a nonlinear one (modeled for instance by Euler - Lagrange equations, [83]). Imposing an input/output form in this case has to be done *a posteriori* for instance to test the model via computer simulations and further to design a controller. In this paragraph we mean we have to follow the modelling - testing (via simulations) - controller design (also helped by numerical simulations) process in order to implement computationally a complete design.

R.Z. Yoe: *Modelling and Control of Dynamical Systems: Numerical Implementation in a Behavioral Framework*, Studies in Computational Intelligence (SCI) **124**, 1–11 (2008)
www.springerlink.com

Looking for a model of the outcomes that can appear, we have to consider that (before any modelling) the outcomes are in a set, which we will call the *universum*. In obtaining a model of the phenomenon, the outcomes are declared to belong to the *behavior* of the model, a subset of the universum. This subset is called the *mathematical model*. Thus we arrive at the following definition:

Definition 1 *A mathematical model is a subset* $\mathfrak{B}$ *of a set* $\mathbb{U}$*, called the universum, or the set of outcomes:*

$$\mathfrak{B} \subseteq \mathbb{U}. \tag{1.1}$$

The subset $\mathfrak{B}$ *is called the behavior of the model.*

What does modelling a system mean? For us, a system[2] will be any physical structure that we want to obtain a model of. Traditionally, such model is a mathematical abstraction in terms of differential (or difference) equations. The way a model is obtained depends on the point of view of the modeler. We will argue that the Behavioral Approach offers a generalized perspective to modelling.
When is a representation of a system considered suitable? Again the answer depends on the modelers considerations and goals. We will argue that providing a suitable abstraction is one of the goals of the Behavioral Approach to modelling. If we have a good model available, we shall be able to go further: designing a controller for our system, also from a Behavioral perspective.

1.2 Paradigms in Modelling

Before the conception and development of the behavioral approach, researchers were looking for an adequate method to model systems from a very general point of view (or from the *most* general point of view as possible). We pose the questions given above because modelling implies to isolate (mentally) a portion of the universe (fixing then a *boundary*) we are interested in (the *system*) in order to study it. Anything outside the boundary becomes the *environment* of the system. However, how general can a modelling approach be? How to constrain it or how to choose a good enough framework to work with? In order to give an answer to that, we have to create, to develop such a modelling tool from the most general point of view we can provide for studying *macroscopic*[3] systems: thermodynamics. Thermodynamics says that an *isolated* system is one which does not allow exchange of

[2]The Behavioral Approach deals with a deterministic system concept.

[3]The state of a system in mechanics is completely specified at a given instant of time if the position and velocity of each mass point of the system are given. Naturally, for a system constituted of n mass points, we would need to know $6n$ variables in every instant of time. In thermodynamics a different concept of the state is considered. In fact, the usual dynamical definition of state would be inopportune because all the systems which are dealt with in thermodynamics contain a very large number of mass points (atoms, molecules) so it would be rather impossible to determine those $6n$ variables for all of them. Moreover, it would be unnecessary to do so, because the main variables thermodynamics considers are average properties of the system, and hence a detailed knowledge of the motion of each mass point would be superfluous. In addition, in almost every system studied by thermodynamics, one assumes that the

matter or energy through its boundary. A *closed* system is defined as the one which does allow interchange of energy but not of matter with the environment. Finally, an *open* system considers both energy and matter circulating through its boundary as possible [22]. Linking these universal laws with a dynamical systems framework[4], yields the definitions provided in the following section which are key for the behavioral approach: *closed* and *open* dynamical systems[5].

1.2.1 Closed dynamical systems

Closed dynamical systems have been studied extensively in mathematics. This kind of systems can be represented[6] by the class $\frac{dx}{dt} = f(x)$ (see [89]). Note that the evolution in time of this kind of flows is described by the dynamical laws (in terms of the vector field f) and the initial state $x_0 = x(0)$ associated to the dynamical system at hand. In this case, from definition 1, $\mathfrak{B}$ is given by the set of all state trajectories. Improving the latter flow model by taking into account an output the modeler can observe, would yield to the following observed flow given by

$$\frac{dx}{dt} = f(x),\ y = h(x) \tag{1.2}$$

Although also for this case $\mathfrak{B}$ is the set of all output trajectories y, we have to note that it may be impossible to express the behavior $\mathfrak{B}$ as the set of solutions of a differential equation involving only y [89]. The auxiliary nature of the variable x (the state) then arises. In fact, the state permits to define an energy flux through the entire system involved. To clarify the latter, let us consider that equation 1.2 has as particular form $\frac{dx}{dt} = Ax,\ y = Cx$. An energetic interpretation scheme for such expression is given in figure 1.1. We can appreciate that energy transformation and dissipation are linked to the matrix A, and an information extraction map is associated to the matrix C. However, this scheme is based strongly on one single point belonging to the state space of this system (denoted in the figure as x_0). Thus we realize that, in order to construct the state of the system we are forced to start with an initial state, which appears to imply a vicious circle.

different parts of the system either are at rest or are moving so slowly that kinetic energies can be neglected [22].

[4]Let us consider for a moment the thermodynamics state equation $f(P, V, T) = 0$ (P=pressure, V=volume,T=temperature). Its form depends on the substance under study. Any variable there can be expressed as a function of the other two. Therefore, the state of the system is completely determined by any two of the three quantities P, V, T. For instance, if we consider T constant, a point in the (V, P) plane defines a state of the system. Such a thermodynamical state alone is not sufficient for the determination of the dynamical state. With increasing time, the system exists successively in all these dynamical states that correspond to the given thermodynamical state. From this point of view, we may say that a thermodynamical state is the *ensemble of all dynamical states* through which, as a result of molecular motion, the system is rapidly passing [22].

[5]Nevertheless these terms in the Behavioral Approach are stronger than in Thermodynamics.

[6]This form is also known as the zero input form of a system, [52]

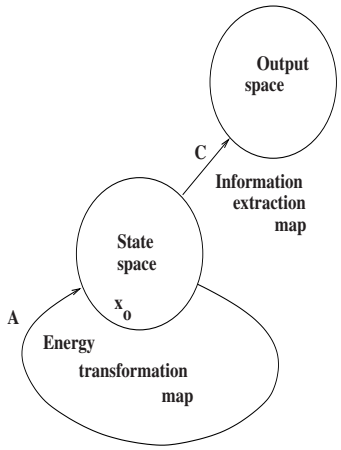

Figure 1.1: *Energy transformation in a linear system.*

1.2.2 Open dynamical systems and the input/output approach

In contrast, a dynamical system that interacts with its environment is referred to as *open* [89]. Such interaction is done exchanging a physical quantity as mass or energy, or it may consist simply of exchange of information. In this case, the evolution in time is characterized by the dynamical laws, the initial state and, in addition, by the influence of the *environment.* The latter may take the form of an input function that drives the system. For instance, let us consider the following input/output form of a dynamical system:

$$f_1\left(w_1, \frac{dw_1}{dt}, \frac{d^2w_1}{dt^2}, \ldots, t\right) = f_2\left(w_2, \frac{dw_2}{dt}, \frac{d^2w_2}{dt^2} \ldots, t\right) \tag{1.3}$$

with the following associated spaces: $\mathbb{T} = \mathbb{R}$ (time), $\mathbb{W} = \mathbb{I} \times \mathbb{Y}$ (input $\times$ output signal spaces) and $\mathfrak{B}$ = all input/output pairs of trajectories. Depending of the nature of the system we deal with, it may be difficult to define what variables deserve the honor of being called inputs or outputs just by inspection of equation 1.3. We would need to know more about the nature of the system to decide whether obtaining a model within an input/output framework is adequate or not. It might not be natural.

For the time being, let us assume that we have been able to find an input/output representation for our plant, which is assumed linear, causal, time invariant and relaxed (initial conditions equal to zero) at $t = 0$. Then for a given input u the corresponding output y can be expressed by means of

$$y(t) = \int_0^t H(t-\tau)u(\tau)d\tau \tag{1.4}$$

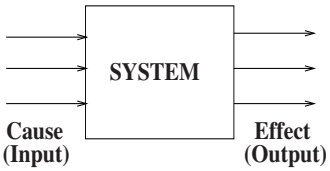

Figure 1.2: *Input/Output systems.*

where H is the impulse response matrix. If we do not assume the plant to be relaxed at $t = 0$, then after applying the input u part of the response y will be excited by the initial conditions and for different initial conditions, different responses will be excited by the *same* input u. Hence, the input/output framework fails dealing with initial conditions.
On the other hand, let us consider the representation of a plant as it is given by equation 1.2 (closed dynamical system). If we think of it as an open system, we obtain

$$\frac{dx}{dt} = f(x, u), \; y = h(x, u) \tag{1.5}$$

Observing the latter expression we can wonder which of the variables u, y and x deserves to be called input and which of the variables output of the plant?
In order to show how ambiguous the choice of input and output may be, let us consider the example provided below.

Example 1 *[86] Let us consider a mechanical system of k points with masses m_i, $i = 1, 2, \ldots, k$ placed at positions q_i, $i = 1, 2, \ldots, k$ with F_i, $i = 1, 2, \ldots, k$ forces acting on them. We want to determine what set of variables deserves to be called input and what set of variables will be considered as output of this system.*
Newton's second law establishes that the particles are related by

$$m_i \ddot{q}_i = F_i$$

As we see, our natural experience indicates to consider the forces F_i as inputs and the resulting positions q_i as outputs. Nevertheless, considering that the forces act in a potential field V with the paths of the q_i's determined by some external action, the relation becomes

$$F_i = -\frac{\partial V(q_1, q_2, \ldots, q_k)}{\partial q_i}$$

Now, the roles have been interchanged. The q_i's are inputs and the F_i's are outputs.
Now, let us combine the equations give above in order to get an expression to determine the motion of these particles. Thus we obtain

$$m_i \ddot{q}_i + \frac{\partial V(q_1, q_2, \ldots, q_k)}{\partial q_i} = 0$$

The latter model says that our mechanical system has no external influence from its environment (we would be obtaining its zero input response). The states are q_i and $p_i = m_i\dot{q}$ for $i = 1, 2, \ldots, k$ and the outputs are q_i and F_i. The motion would be caused by initial conditions, i.e., the initial state.
Then, as a result of this example, we realize that there are situations where assigning the role of input/output to some set of variables may be unclear.

Not always a set of differential equations represent suitably the behavior of a physical system. Thermal systems are classical examples of this as we show in the example given next.

Example 2 *A thermodynamic engine is illustrated in figure 1.3. There is a heating terminal which concerns two variables: the heat supplied by the environment Q_h and an associated temperature T_h. Also, at the cooling side, we have Q_c and T_c as terminal variables. The dual nature of the work (either applied to or developed by the machine) implies a bidirectional terminal for this variable.*

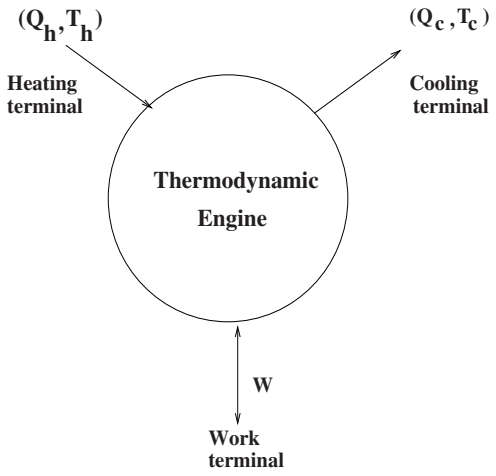

Figure 1.3: *Thermal system.*

The system shown in figure 1.3 involves the following: a time axis $\mathbb{T} = \mathbb{R}$, variable of interest (Q_h, T_h, Q_c, T_c, W) that takes its values in what we call the signal space $\mathbb{W}$, with $\mathbb{W} = \mathbb{R}_+ \times \mathbb{R}_+ \times \mathbb{R}_+ \times \mathbb{R}_+ \times \mathbb{R}$, and a behavior $\mathfrak{B}$, a suitable family of trajectories. $\mathfrak{B}$ will

be restricted by the first and second law of thermodynamics:

$$\oint (Q_h - Q_c - W)\, dt = 0 \tag{1.6}$$

$$\oint \left(\frac{Q_c}{T_c}\right) dt \geq \oint \left(\frac{Q_h}{T_h}\right) dt \tag{1.7}$$

These laws deal with open systems but not with input/output systems. In order to impose an input/output structure to the diagram displayed in figure 1.3 we have to act against the nature of the physical system involved in order to fit in such action/reaction paradigm. The resulting diagram would be figure 1.4. Equations (1.6) and (1.7) do not correspond to such a cause/effect diagram.

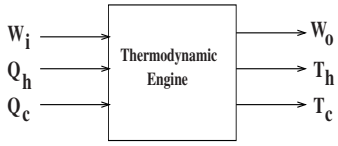

Figure 1.4: *Is an Input/Output structure always suitable?.*

Hence, after explaining the ideas given above, we can conclude that a framework defined as $\mathfrak{B} \subseteq \mathbb{U}$ has better perspectives because of its flexibility and universality. In fact, such definition will be more precise in section 2.2.

From the ideas given above, we realize that a dynamical system does not always work as a *signal processor*, i.e., just mapping inputs into outputs. Rather a physical system will be considered as an open dynamical system which does not impose restrictions on the signal flow of the signals associated to it. As a consequence, although a dynamical system may be represented as a block diagram, or as as a black box, the traditional unidirectional input/output flow (as it is shown in figure 1.2) will seldomly be a given structure. In fact, more frequently we shall meet systems interacting with the environment by means of bidirectional flows of signals. But always, as we showed in example 1.3, the interaction of the system with the universe will be via *terminal variables*. The latter consideration does not impose directional restrictions on the interaction of the system with its environment, (see figure 1.5).

Moreover, from the latter diagrams we can infer now how to describe interaction between systems, namely, how to interconnect systems. What we said before about the flow of signals for individual blocks remains valid for the interconnection of two blocks. An input/output flow or a bidirectional flow in the blocks will not be the rule, interconnection of blocks through terminals is the key again (see figure 1.6).

Interconnection is a word that rings a bell to engineers because it is associated with simulation of interconnected systems. Obviously, before being able to simulate, we have to

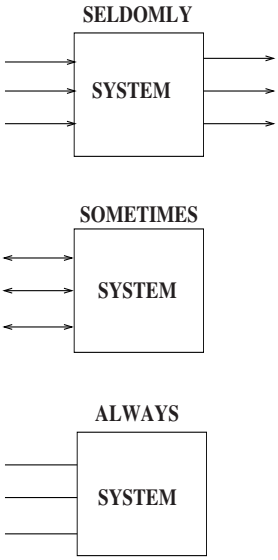

Figure 1.5: *Does there exist actual signal flow in dynamical systems?*

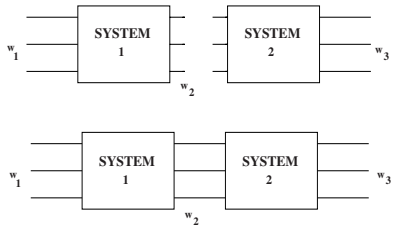

Figure 1.6: *General scheme for interconnection.*

obtain some representation of the plant or interacting systems. Traditionally, within the input/output approach we have to adjust physical interconnections to signal flow assignment. For instance, although drawing a block diagram for a simple RC-circuit is simple, it is in general not immediately clear how to represent it as the interconnection of input/output systems. Let us take a look at the following example.

Example 3 *Consider the simple network given by the following block diagram, where R is a resistor (in ohms) and C is a capacitor (in farads).*

Applying Kirchhoff's voltage and current laws (KVL and KCL respectively) we get the following equations for the model

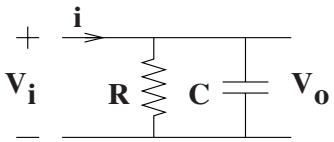

Figure 1.7: *RC circuit diagram.*

$$I = \frac{V_i - V_o}{R},\ V_o = \frac{1}{C}\int I dt$$

The corresponding diagram is as shown below.

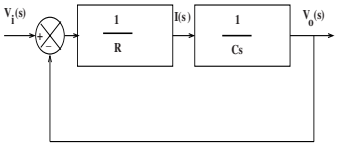

Figure 1.8: *RC circuit block diagram.*

Thus, in order to simulate such a simple network (using the popular SIMULINK [49] we would need to draw a block diagram for the resistor and the capacitor and draw the final interconnected input/output representation. Moreover, suppose we want to model a more complicated circuit consisting of many components (resistors, capacitors, inductors, transformers, etc.) We would need to proceed analogously as we did for the simple circuit. We learn from this example that physical interconnections do not necessarily coincide with the signal flow of the diagram we need to simulate the system from. Thus, although useful, sometimes a transfer function or transfer matrix approach may not be the best one. Nevertheless such input/output - based software is widely used in industry and in the academic world. An example of software which respects the system's physical interconnections is MODELICA [50]. On the other hand, we have not said anything about modelling of distributed parameters systems, like electric cables, antennas, etc. They are difficult to study within this traditional viewpoint because partial differential equations come into the picture.

1.2.3 More about the input/output approach

The latter section may seem to indicate that the input/output approach deals only with linear systems. This is not the case. The input/output framework also applies to nonlinear systems. In this case, operators between suitable spaces are defined in order to represent

the systems. The way they interact is by means of energy exchange; *passivity* is the keyword (different kinds of *passive operators* are used). Analysis and synthesis of interconnected systems are investigated and developed in this spirit. Nevertheless, even this formal framework ([85] and [80]), although well applicable to real physical systems, ([53], [42], [94],) is not adequate enough to model some systems where interchange of matter and energy actually occurs, as we already explained. The latter again requires to introduce input/output maps, regardless of the physical nature of the system.

1.2.4 The behavior of the system is the key

Motivated by the considerations in the latter sections, a more general viewpoint is required. The central issue seems to be: *how does the system under consideration behave? What does it do?* The answer is that a system behaves displaying by different trajectories with respect to the *time*. Therefore, we should define the *behavior* of a system as *the set of all trajectories of the system variables that, according to the mathematical model, are possible.*

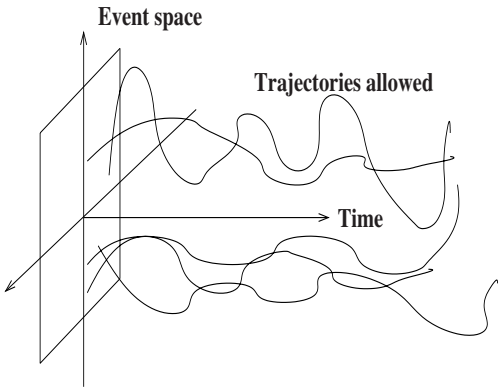

Figure 1.9: *The permitted trajectories defines the behavior.*

1.2.5 Some other frameworks for systems and control

In the past sections we have tried to motivate the behavioral perspective. Nevertheless, there are other viewpoints to systems and control. Without going into details we mention the following just to give an idea of how different are some other approaches with respect to our Behavioral Approach:

a. Input/Output blocks: Wiener [84], Heaviside (see for instance [57] and references therein).

b. Input/State/Output systems, Kalman [36]. The state is an important addition because it avoids to think of the I/O approach as a map and allows us to take care of initial conditions.

c. Power Ports - Bond Graphs: Ports; Tearing and zooming of subsystems; energy storage, transmission and dissipation: Paynter [55], Kron (see for instance [39], a very interesting paper), Karnopp (see [37] and references therein).

d. Geometric Control: Poincaré (see [26] and references therein), Hermann (see for instance [28]), Brockett [13].

e. Behavioral Control. A dynamical system is considered as a collection of trajectories. Control is viewed as interconnection of components in open systems, considering feedback as particular case of interconnecting: Willems [58].

f. Port Hamiltonian Systems: Geometric modelling, control by interconnection, energy shaping, damping injection: van der Schaft [67], Ortega [53].

Chapter 2

Behavioral framework

2.1 Modelling by Tearing and Zooming

In general, a physical system is constituted of smaller structures called subsystems. As a consequence, a system is said to be *modular*. Such structures are interconnected by means of *terminals*. A system interacts with its environment via such terminals. We shall associate to each terminal a certain number of physical variables. Hence, each subsystem will display a terminal *behavior*, understood informally now as the evolution in time of all the physical signals or variables associated with each terminal.

2.1.1 Constitutive models

In order to illustrate the ideas exposed in the latter paragraph, in this section we introduce an example of modelling a physical system. The modelling takes place by the process of tearing and zooming, i.e. by considering the system as the interconnection of simpler subsystems. A system may be conceived as a black box with terminals. For the classical input/output approach such terminals would be labeled as inputs and outputs, but the Behavioral Approach is not constrained in that sense. Rather, a set of terminals is considered as the media utilized by the system (the black box) to interact with its environment. The terminals are then the interface used to exchange information with the external world. Assigning a vector valued variable $s_i = (s_1, s_2, ..., s_n) \in \mathbb{S}_i$ (for some set $\mathbb{S}_i$) to the $i-th$ terminal, we define the terminal signal space of the system as $\mathbb{S} = \mathbb{S}_1 \times \mathbb{S}_2 \ldots \mathbb{S}_n$. If we consider our variables s_i as functions of another set variables x, called the independent variables (such as space, time, etc.) defined as belonging to an indexing set $\mathbb{I}$, the set of all possible outcomes our system can -in principle- take is the set of all possible maps from $\mathbb{I}$ to $\mathbb{S}$, i.e., our universum in this case is $\mathbb{U} = \mathbb{S}^{\mathbb{I}}$. Nevertheless, the system is governed as well by restricting laws (internal constraints) in such a way that the actual set of allowed signals become just a subset $\mathfrak{B} \subseteq \mathbb{S}^{\mathbb{I}}$ which represents the permitted terminal behavior of the system involved. Such subset is called the terminal behavior. The latter idea will be illustrated by means of a modelling example below.

R.Z. Yoe: *Modelling and Control of Dynamical Systems: Numerical Implementation in a Behavioral Framework*, Studies in Computational Intelligence (SCI) **124**, 13–34 (2008)
www.springerlink.com

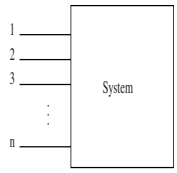

Figure 2.1: *A black box model interacts via terminals.*

The way a relatively complex system is modelled is by tearing the system into subsystems and then zooming in on the individual subsystems. Hence, a model of a physical system will be characterized by the following elements (see also [17] and [87]:

a. *Modules, which represent subsystems.*
A module is characterized by its type, parametrization, and parameter values. Once we have chosen the type of the module (which specifies a set of possible behaviors), we fix the physical variables corresponding to an ordered set of terminals. Parametrization is concerned with mapping the inherent parameters of the module to a suitable behavior. Then when we define a module in this way, we actually obtain the behavior of the terminal variables of this module.

b. *Terminals, the physical links between subsystems.*
A terminal is specified by its type. The type imposes an ordered set of terminal variables.

c. *The interconnection architecture, the layout of the modules and their interconnection.*
It consists of a set of terminal pairs (unordered, disjoint, and with distinct elements). If a terminal pair belong to such a set, then we say that the terminals are connected. Naturally, terminal pairs must be of the same kind (electrical, mechanical, etc. This case is called adapted). When the latter is not possible we need a connector (which can couple different kinds of terminal variables). Pairing of adapted terminals imposes an interconnection law.

d. *The manifest variable assignment, the variables we want to study.*
We realize that the model (at this stage) will contain, in addition to manifest variables, latent variables that arise either from interconnecting submodules or from using them as auxiliary modelling variables.

Example 4 Translational mechanical systems
Consider the mechanical system of figure 2.2. In general, the time evolution of a mechanical system in three dimensions involves translations, as well rotations and deformations. Nevertheless, we consider its components as ideal elements (non deformable). In this spirit, in translational mechanical systems we assume that there is no rotation (or it is neglected). In contrast, if we consider a mechanical system that rotates (with neglectable or zero translational movement) we call this a rotational mechanical system. The elements we use to

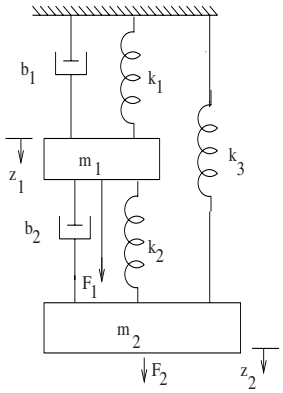

Figure 2.2: *Mechanical structure.*

model both kind of structures (translational and rotational) are the same (masses, springs and dampers), that is why we show only the translational case (in a rotational system torque plays the role of force). In addition, there exists (mechanical - electrical), (electrical - thermal), etc., analogies. These can be treated in analogous way. In order to model now the above translational mechanical system, we will first need to describe the following building blocks, which will be identified later on as the constitutive modules of the system.

Example 5 Translational damper.
This system is used to model energy dissipation. The latter phenomenon appears when kinetic energy is transformed in thermal energy by viscous friction. Such object requires a steady force to maintain a certain velocity proportional (in terms of the parameter b) to the force applied (this is the internal restricting law). It is assumed that kinetic and potential energy storage phenomena are absent. This mechanical system involves two terminal variables $s_i = (F_i, q_i), i = 1, 2$.

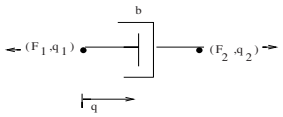

Figure 2.3: *Translational damper.*

$$\mathfrak{B} = \left\{ (F_1, q_1, F_2, q_2) : \mathbb{R} \to \mathbb{R}^4 \mid F_1 = F_2, F_1 = b\frac{d}{dt}(q_1 - q_2) \right\}$$

Example 6 Translational spring.
A spring is a mechanical object which subject to a force either compresses or elongates without loss of energy due to friction force. Also, it is assumed that the acceleration of its parts is neglectable. In figure 2.4 we see that it has two mechanical terminals, to each of which we associate a force F_i and a position q_i. Then in this case $s_i = (F_i, q_i), i = 1, 2$. The internal restricting law is known as Hooke's law. This physical law constrains the terminal variables to the set $\mathfrak{B}$ shown below:

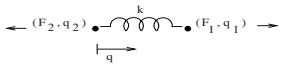

Figure 2.4: *Translational spring.*

$$\mathfrak{B}_n = \left\{(F_1, q_1, F_2, q_2) : \mathbb{R} \to \mathbb{R}^4 \mid F_1 = F_2, F_1 = k(q_1 - q_2))\right\}$$

Example 7 Translational mass.
A pure translational mass is a rigid mechanical object which is moving through a non-dissipative environment. It is characterized also by a forces F_i and positions q_i. Here also $s_i = (F_i, q_i), i = 1, 2$. The internal law which restricts the trajectories of a mass is Newton's second law, which links the momentum $p = mdq/dt$ of the mass linearly to the object's velocity dq/dt. p is defined as $F = dp/dt$.

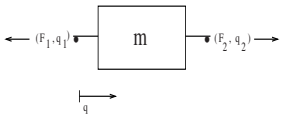

Figure 2.5: *Mass.*

$$\mathfrak{B} = \left\{(F_1, q_1, F_2, q_2) : \mathbb{R} \to \mathbb{R}^4 \mid F_1 = F_2, F_1 = m\frac{d^2}{dt^2}(q_1 - q_2))\right\}$$

Now, after the introduction of this set of mechanical modules, we proceed with the modelling process outlined above. That indicates that we have to assign a label to the terminals of each module, see figure 2.6. Next, by zooming in, in the table below we identify seven mechanical modules (two dampers, three springs and two masses) plus four connectors. The terminals are indicated as well as the units of each constitutive subsystem.

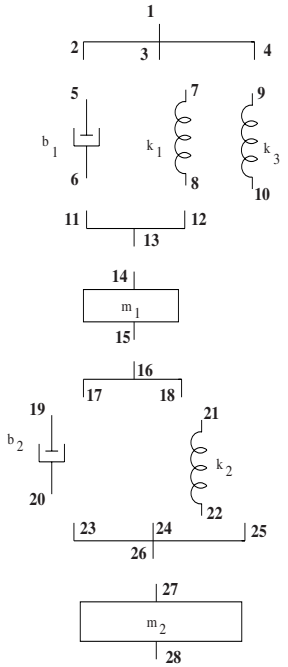

Figure 2.6: *Tearing : Labeling modules.*

Module	Type	Terminal	Parameter
Connector 1	4-terminal connector	(1,2,3,4)	
b_1	Damper	(5,6)	b (Ns/m)
k_1	Spring	(7,8)	k (N/m)
k_3	Spring	(9,10)	k (N/m)
Connector 2	3-terminal connector	(11,12,13)	
m_1	Mass	(14,15)	m (kg)
Connector 3	3-terminal connector	(16,17,18)	
b_2	Damper	(19,20)	b (Ns/m)
k_2	Spring	(21,22)	k (N/m)
Connector 4	4-terminal connector	(23,24,25,26)	
m_2	Mass	(27,28)	m (kg)

Next, we define the interconnection architecture to be given by

$$\text{Pairing}=\{(2,5),(3,7),(4,9),(6,11),(8,12),(13,14),(15,16),$$
$$(17,19),(18,21),(20,23),(22,24),(10,25),(26,27)\}$$

• Equations for the full behavior
By collecting the behavioral equations of the constitutive modules, we finally obtain the set of equations that represent a model of the original mechanical system. These equations describe what we will later call the *full behavior* of the system.

Module	Constitutive Equations	
Connector 1	$F_1 = F_2 = F_3 = F_4$	
b_1	$F_5 = F_6$	$F_5 = b_1(\dot{q}_5 - \dot{q}_6)$
k_1	$F_7 = F_8$	$F_7 = k_1(q_7 - q_8)$
k_3	$F_9 = F_{10}$	$F_9 = k_3(q_9 - q_{10})$
Connector 2	$F_{11} = F_{12} = F_{13}$	
m_1	$q_{14} = q_{15}$	$F_{14} - F_{15} = m_1\ddot{q}_{14}$
Connector 3	$F_{16} = F_{17} = F_{18}$	
b_2	$F_{19} = F_{20}$	$F_{19} = b_2(\dot{q}_{19} - \dot{q}_{20})$
Connector 4	$F_{23} = F_{24} = F_{25} = F_{26}$	
m_2	$q_{27} = q_{28}$	$F_{27} - F_{28} = m_2\ddot{q}_{27}$

Now we put the information about pairing and interconnection equations in order to show the relation among the pair terminals in our architecture:

Interconnection	Interconnection Equations	
$(2,5)$	$q_2 = q_5$	$F_2 + F_5 = 0$
$(3,7)$	$q_3 = q_7$	$F_3 + F_7 = 0$
$(4,9)$	$q_4 = q_9$	$F_4 + F_9 = 0$
$(6,11)$	$q_6 = q_{11}$	$F_6 + F_{11} = 0$
$(8,12)$	$q_8 = q_{12}$	$F_8 + F_{12} = 0$
$(13,14)$	$q_{13} = q_{14}$	$F_{13} + F_{14} = 0$
$(15,16)$	$q_{15} = q_{16}$	$F_{15} + F_{16} = 0$
$(17,19)$	$q_{17} = q_{19}$	$F_{17} + F_{19} = 0$
$(18,21)$	$q_{18} = q_{21}$	$F_{18} + F_{21} = 0$
$(20,23)$	$q_{20} = q_{23}$	$F_{20} + F_{23} = 0$
$(22,24)$	$q_{22} = q_{24}$	$F_{22} + F_{24} = 0$
$(10,25)$	$q_{10} = q_{25}$	$F_{10} + F_{25} = 0$
$(26,27)$	$q_{26} = q_{27}$	$F_{26} + F_{27} = 0$

All of these equations combined define a latent variable representation (a latent variable is an auxiliary variable, i.e. a variable we use to represent the variables we are actually interested in, called the manifest variables and that are denoted by w):

$$w = (F_1, q_1, F_{28}, q_{28})$$

with latent variables

$$\ell = (F_2, q_2, ..., F_{27}, q_{27})$$

In the next section we formalize the concept of dynamical system.

2.2 Dynamical Systems

In the behavioral approach to systems, a dynamical system is a special case of a mathematical model as defined in the previous chapter. Given is a set of independent variables denoted by $\mathbb{T}$, often called *the time axis* and a set of dependent variables $\mathbb{W}$, called *the signal space*. The universe $\mathbb{U}$ is defined as the set of all functions from $\mathbb{T}$ to $\mathbb{W}$ and the behavior $\mathfrak{B}$ of the model, the set of all possible outcomes, is a subset of the universe $\mathbb{U}$. In this way we arrive at the following definition of dynamical system:

Definition 2 *A dynamical system Σ is a triple $\Sigma = (\mathbb{T}, \mathbb{W}, \mathfrak{B})$ with $\mathbb{T}$ a set, called the time axis; $\mathbb{W}$ a set called the signal space, and $\mathfrak{B} \subseteq \mathbb{W}^{\mathbb{T}}$ is the behavior of the system.*

Definition 3 *A dynamical system $\Sigma = (\mathbb{T}, \mathbb{W}, \mathfrak{B})$ is called* linear *if*

a. *$\mathbb{W}$ is a vector space and*

b. *The behavior $\mathfrak{B}$ is a subspace of $\mathbb{W}^{\mathbb{T}}$*

The latter is nothing but the superposition principle:

$$w_1, w_2 \in \mathfrak{B}, \quad k_1, k_2 \text{ scalars, then } k_1 w_1 + k_2 w_2 \in \mathfrak{B}$$

If the time axis $\mathbb{T}$ is a semi-group, and $\sigma^t w$ is defined by $(\sigma^t w)(\tau) = w(t+\tau)$ for all $t, \tau \in \mathbb{T}$, then we can also define time invariance of a behavior.

Definition 4 *A dynamical system $\Sigma = (\mathbb{T}, \mathbb{W}, \mathfrak{B})$ is called* time-invariant *if for each trajectory $w \in \mathfrak{B}$ the shifted trajectory $\sigma^t w$ is again an element of $\mathfrak{B}$, for all $t \in \mathbb{T}$.*

In this book we deal with systems in which the time axis is equal to the real line $\mathbb{R}$, and where $\mathbb{W}$ is a finite dimensional vector space over the field of real numbers $\mathbb{R}$. We restrict ourselves moreover to linear and time-invariant systems.

2.2.1 Linear Differential Systems

In this book we will restrict ourselves even more to so-called lumped linear time-invariant dynamical systems. These are dynamical systems $\Sigma = (\mathbb{R}, \mathbb{R}^{\mathbb{w}}, \mathfrak{B})$, where $\mathbb{R}$ is the time axis, $\mathbb{R}^{\mathbb{w}}$ is the signal space, and where the behavior $\mathfrak{B}$ is a subset of $\mathfrak{C}^\infty(\mathbb{R}, \mathbb{R}^{\mathbb{w}})$ (the space of all infinitely often differentiable functions from $\mathbb{R}$ to $\mathbb{R}^{\mathbb{w}}$) consisting of all solutions of a set of linear, constant coefficient differential equations. It is easily seen that such dynamical system is linear and time-invariant in the sense defined above. The behavioral

approach makes a distinction between the behavior as the space of all solutions to a set of (differential) equations, and the set of equations itself. A set of equations in terms of which the behavior is defined, is called a *representation* of the behavior. Such equations are called *behavioral equations* as well[1]. The *behavior* consists of the solutions to these equations.

Polynomial Matrices and linear differential systems

In the scope of this work, physical systems are modelled by linear differential systems. That means that we shall frequently have to deal with systems of linear differential equations. The number of equations depends on the nature of the systems at hand. However, even for small sized systems, the mathematical representation may be cumbersome. For instance, let us consider the example shown in figure 2.2. For simplicity, assume that F_1 and F_2 are zero. Then a set of behavioral equations of this system turns out to be:

$$m_1\frac{d^2z_1}{dt^2} + b_1\frac{dz_1}{dt} + k_1z_1 + b_2\left(\frac{dz_2}{dt} - \frac{dz_1}{dt}\right) + k_2(z_2 - z_1) = 0 \tag{2.1}$$

$$m_2\frac{d^2z_2}{dt^2} + k_3z_2 + b_2\left(\frac{dz_2}{dt} - \frac{dz_1}{dt}\right) + k_2(z_2 - z_1) = 0 \tag{2.2}$$

We can imagine that for much bigger systems such a notation (and hence *representation*) becomes cumbersome. The latter set of equations is a special case of the general one shown below:

$$\begin{array}{ccccccc} r_{110}w_1 + \cdots + r_{11n_{11}}\dfrac{d^{n_{11}}}{dt^{n_{11}}}w_1 + \cdots + r_{1\mathtt{w}0}w_{\mathtt{w}} + \cdots + r_{1\mathtt{w}n_{1\mathtt{w}}}\dfrac{d^{n_{1\mathtt{w}}}}{dt^{n_{1\mathtt{w}}}}w_{\mathtt{w}} = 0 \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ r_{g10}w_1 + \cdots + r_{g1n_{11}}\dfrac{d^{n_{11}}}{dt^{n_{11}}}w_1 + \cdots + r_{g\mathtt{w}0}w_{\mathtt{w}} + \cdots + r_{g\mathtt{w}n_{1\mathtt{w}}}\dfrac{d^{n_{1\mathtt{w}}}}{dt^{n_{1\mathtt{w}}}}w_{\mathtt{w}} = 0 \end{array}$$

where we have g scalar differential equations. Each of them involves the scalar variables $w_1, w_2, ..., w_{\mathtt{w}}$. A more compact notation for the latter is obtained by defining polynomials $r_{ij}(\xi)$ by $r_{ij}(\xi) = r_{ij0} + r_{ij1}\xi + r_{ij2}\xi^2 + \ldots + r_{ijn_{ij}}\xi^{n_{ij}}$. In this way the above equations can be re-written as

$$\begin{bmatrix} r_{11}(\frac{\mathrm{d}}{\mathrm{d}t}) & \ldots & r_{1q}(\frac{\mathrm{d}}{\mathrm{d}t}) \\ \vdots & & \vdots \\ r_{g1}(\frac{\mathrm{d}}{\mathrm{d}t}) & \ldots & r_{gq}(\frac{\mathrm{d}}{\mathrm{d}t}) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{\mathtt{w}} \end{bmatrix} = 0$$

If we now introduce a matrix $R(\xi)$ as follows

$$R(\xi) = \begin{bmatrix} r_{11}(\xi) & \ldots & r_{1q}(\xi) \\ \vdots & & \vdots \\ r_{g1}(\xi) & \ldots & r_{gq}(\xi) \end{bmatrix}, \tag{2.3}$$

[1]Naturally, discrete systems given in terms of difference equations ($\mathbb{T} = \mathbb{Z}$) can also be modelled like this.

then these equations obtain the form

$$R\left(\frac{d}{dt}\right)w = 0. \tag{2.4}$$

$R(\xi)$ is called a polynomial matrix. A polynomial matrix is a matrix whose entries are polynomials (in one unknown ξ for lumped systems). Such polynomial matrix can also be expressed as a *matrix polynomial*, defined as a polynomial with matrix coefficients: $R(\xi) = R_0 + R_1\xi + \cdots + R_{d-1}\xi^{d-1} + R_d\xi^d$. Considering the space of all possible solutions of equation (2.4), this equation defines the system $\Sigma = (\mathbb{R}, \mathbb{R}^{\mathtt{w}}, \mathfrak{B})$ with

$$\mathfrak{B} = \left\{w \in \mathfrak{C}^\infty(\mathbb{R}, \mathbb{R}^{\mathtt{w}}) \mid R\left(\frac{d}{dt}\right)w = 0\right\} \tag{2.5}$$

Obviously, this system is linear and time-invariant. Such a system Σ will henceforth be called *a linear differential system*[2]. The set of all linear differential systems with $\mathtt{w}$ variables will be denoted by $\mathfrak{L}^{\mathtt{w}}$. Instead of $\Sigma \in \mathfrak{L}^{\mathtt{w}}$, we will often simply write $\mathfrak{B} \in \mathfrak{L}^{\mathtt{w}}$. The representation (2.5) of $\mathfrak{B}$ is called a *kernel representation* of $\mathfrak{B}$, we often write $\mathfrak{B} = \ker(R(\frac{\mathrm{d}}{\mathrm{d}t}))$.

2.3 Latent variables and elimination

As it was argued in [58], section 6.2, when we obtain a model of a system, in general we need to introduce a set of auxiliary variables that we later 'ignore' after obtaining the mathematical model of the variables that we are actually interested in (the manifest variables). Those auxiliary variables are referred to as *latent variables*. Thus a model often has the form of a dynamical system with latent variables, as defined below.

Definition 5 *A dynamical system with latent variables is a quadruple* $\Sigma_L = (\mathbb{T}, \mathbb{W}, \mathbb{L}, \mathfrak{B}_{full})$ *where* $\mathbb{T}$ *is the time axis,* $\mathbb{W}$ *is the manifest signal space,* $\mathbb{L}$ *the latent variable space and* $\mathfrak{B}_{full} \subseteq (\mathbb{W} \times \mathbb{L})^{\mathbb{T}}$. $\mathfrak{B}_{full}$ *is referred as to the* full behavior *of the system.*

Definition 6 *Let* $\Sigma_L = (\mathbb{T}, \mathbb{W}, \mathbb{L}, \mathfrak{B}_{\text{full}})$ *be dynamical system with latent variables. The dynamical system induced by* Σ_L *is defined as* $\Sigma = (\mathbb{T}, \mathbb{W}, \mathfrak{B})$ *with the* manifest behavior *defined as*

$$\mathfrak{B} = \left\{w \in \mathbb{W}^{\mathbb{T}} \mid \exists\ \ell \in \mathbb{L}^{\mathbb{T}}\ s.t.\ (w, \ell) \in \mathfrak{B}_{full}\right\}.$$

If we concentrate attention for a moment on definition 5, we notice that with respect to definition 6, the latter is obtained from the former after 'ignoring' $\ell \in \mathbb{L}^{\mathbb{T}}$. In fact, we have *projected* $\mathfrak{B}_{\text{full}}$ on the manifest variables w. More formally, defining the *projection operator* as $\Pi_w : (\mathbb{W} \times \mathbb{L})^{\mathbb{T}} \to \mathbb{W}^{\mathbb{T}}$ by $\Pi_w(w, \ell) := w$ we notice how the idea explained above can be performed. This allows us to write $\mathfrak{B} = \Pi_w(\mathfrak{B}_{\text{full}})$.

[2]Note that a linear differential system is *time invariant* by definition, in line with the convention used in [58], for brevity time invariance is not included in the name

Now, assume that Σ_L is a linear differential system. We want to know whether the associated system Σ induced by Σ_L is also a linear differential system. For this Π_w needs to preserve linearity and time-invariance. Due to the fact that in this work we consider system trajectories from $\mathfrak{C}^\infty(\mathbb{R}, \mathbb{R}^{\mathbb{w}})$, the following theorem, called *elimination theorem*, is valid (see [58], page 206).

Theorem 1 *Let $\mathfrak{B}_{\text{full}} \in \mathcal{L}^{\mathbb{w}+\ell}$. Consider the behavior defined by*

$$\mathfrak{B} := \left\{ w \in \mathfrak{C}^\infty(\mathbb{R}, \mathbb{R}^{\mathbb{w}}) \mid \exists\ \ell \in \mathfrak{C}^\infty(\mathbb{R}, \mathbb{R}^{\ell}) \text{ s.t. } (w, \ell) \in \mathfrak{B}_{\text{full}} \right\}$$

Then $\mathfrak{B} \in \mathcal{L}^{\mathbb{w}}$.

The latter theorem allows us to get $\mathfrak{B} \in \mathcal{L}^{\mathbb{w}}$ from $\mathfrak{B}_{\text{full}} \in \mathcal{L}^{\mathbb{w}+\ell}$ by *eliminating* the latent variables ℓ. That means that we can obtain a representation for $\mathfrak{B}$ as the one given by equation 2.4. The question is: how to obtain such representation? It was argued in [58] that in the context of linear differential systems the general form of a system with latent variables is given by the model

$$R\left(\frac{d}{dt}\right) w = M\left(\frac{d}{dt}\right) \ell \tag{2.6}$$

This representation is called a *latent variable representation* or *hybrid representation*. We want to obtain a method to actually compute a representation of the manifest behavior associated with this full behavior. This method is called the process of elimination, see [58], chapter 6. This process involves *row compression* of polynomial matrices and the notion of unimodularity.

Definition 7 *Let $U(\xi) \in \mathbb{R}^{g\times g}[\xi]$. Then $U(\xi)$ is said to be a unimodular polynomial matrix if there exists a polynomial matrix $V(\xi) \in \mathbb{R}^{g\times g}[\xi]$ such that $V(\xi)U(\xi) = I$. Equivalently, $det(U(\xi))$ is equal to a nonzero constant.*

Theorem 2 *Let $\mathfrak{B} \in \mathcal{L}^{\mathbb{w}}$ be represented by the latent variable representation 2.6 where $R \in \mathbb{R}^{g\times w}[\xi]$ and $M \in \mathbb{R}^{g\times \ell}[\xi]$. Let $U \in \mathbb{R}^{g\times g}[\xi]$ be a unimodular matrix such that*

$$UM = \begin{bmatrix} M_1 \\ 0 \end{bmatrix}. \tag{2.7}$$

where $M_1 \in \mathbb{R}^{\bullet\times \ell}[\xi]$ has full row rank. Partition

$$UR = \begin{bmatrix} R_1 \\ R_2 \end{bmatrix}.$$

accordingly. Then a kernel representation of $\mathfrak{B}$ is given by $R_2\left(\frac{d}{dt}\right) w = 0$.

The transformation using U applied to M is called row compression.

2.4 Equivalent representations of behaviors

A behavior $\mathfrak{B} \in \mathfrak{L}^{\mathfrak{w}}$ can have more than one representation. For instance, we already defined manifest and latent variable representations for a given behavior (and studied how to get the latter from the former, via elimination). Once we have such a kernel representation, it would be interesting to find equivalent representations for it. Then we speak about equivalent kernel representations of a given behavior.

Theorem 3 *Let $R_1(\xi) \in \mathbb{R}^{g\times \mathfrak{w}}[\xi]$ and $U(\xi) \in \mathbb{R}^{g\times g}[\xi]$. Define $R_2(\xi) = U(\xi)R_1(\xi)$. Denote the behaviors associated with $R_1(\xi)$ and $R_2(\xi)$ by $\mathfrak{B}_1$ and $\mathfrak{B}_2$, respectively. Then:*

a. $\mathfrak{B}_1 \subseteq \mathfrak{B}_2$.

b. If, in addition, $U(\xi)$ is unimodular, then $\mathfrak{B}_1 = \mathfrak{B}_2$.

The converse version is as follows, see [58].

Theorem 4 *The full row rank polynomial matrices $R_1(\xi),\ R_2(\xi) \in \mathbb{R}^{g\times \mathfrak{w}}[\xi]$ represent the same behavior $\mathfrak{B}$ if and only if there exists a unimodular matrix $U(\xi) \in \mathbb{R}^{g\times g}[\xi]$ such that $R_1(\xi) = U(\xi)R_2(\xi)$.*

Theorem 5 *Let $\mathfrak{B}_1, \mathfrak{B}_2 \in \mathfrak{L}^{\mathfrak{w}}$ be represented by kernel representations $R_1\left(\frac{d}{dt}\right) w = 0$ and $R_2\left(\frac{d}{dt}\right) w = 0$, respectively. Then, $\mathfrak{B}_1 \subseteq \mathfrak{B}_2$ if and only if there exists a $D \in \mathbb{R}^{\bullet\times\bullet}[\xi]$ such that $D(\xi)R_1(\xi) = R_2(\xi)$.*

Definition 8 *Let $\mathfrak{B} \in \mathfrak{L}^{\mathfrak{w}}$ and let $R \in \mathbb{R}^{\mathfrak{p}\times \mathfrak{w}}[\xi]$. The kernel representation $R\left(\frac{d}{dt}\right) w = 0$ is said to be a minimal kernel representation of $\mathfrak{B}$ if, whenever $R_1 \in \mathbb{R}^{g\times \mathfrak{w}}[\xi]$ induces a kernel representation of $\mathfrak{B}$ then $p \leq g$, i.e.,* $\operatorname{rowdim}(R) \leq \operatorname{rowdim}(R_1)$.

Theorem 6 *Let $\mathfrak{B} \in \mathfrak{L}^{\mathfrak{w}}$ and let $R \in \mathbb{R}^{g\times \mathfrak{w}}$ induce a kernel representation of the behavior $\mathfrak{B}$. The kernel representation $R\left(\frac{d}{dt}\right) w = 0$ is minimal if and only if the polynomial matrix R has full row rank.*

2.5 Observability and detectability

A very important notion in control systems is observability. Since the behavioral approach considers a system to interact with its environment via its terminal variables, we would like to extend the well known concept of observability (and later on the one of detectability) contemplating our more general viewpoint. More concretely, knowing the values that a certain set of variables, say w_1 (the observed variables) takes, we are interested in obtaining the values of another set of variables, say w_2 (the to-be-deduced variables) from w_1. In this context we consider dynamical systems with signals space given in terms of a Cartesian product $\mathbb{W}_1 \times \mathbb{W}_2$. Thus we consider systems of the form $\Sigma = (\mathbb{T}, \mathbb{W}_1 \times \mathbb{W}_2, \mathfrak{B})$ (see figure 2.7). This motivates the following definition, which is valid for general behavioral systems.

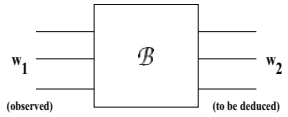

Figure 2.7: *Observability.*

Definition 9 *Consider the system $\Sigma = (\mathbb{T}, \mathbb{W}_1 \times \mathbb{W}_2, \mathfrak{B})$. Assume that trajectories in $\mathfrak{B}$ are partitioned as (w_1, w_2) with $w_i : \mathbb{R} \to \mathbb{W}_i$, $i = 1, 2$. We say that w_2 is observable from w_1 if $(w_1, w_2), (w_1, w_2') \in \mathfrak{B}$ implies $w_2 = w_2'$.*

If we consider systems with latent variables, very often the manifest variables will have the interpretation of the observed ones, and the latent variables (the auxiliary ones introduced during the modelling process) will be interpreted as the to-be-deduced variables.
Now, we are interested in getting a test for observability. Consider the system represented by $R_1(\frac{d}{dt})w_1 + R_2(\frac{d}{dt})w_2 = 0$. The question is: under which conditions on $R_1(\xi)$, $R_2(\xi) \in \mathbb{R}^{\bullet\times\bullet}[\xi]$ is w_2 observable from w_1? The following theorem gives an answer to this question:

Theorem 7 *Let $R_1(\xi) \in \mathbb{R}^{g\times \mathtt{w}_1}[\xi]$ and $R_2(\xi) \in \mathbb{R}^{g\times \mathtt{w}_2}[\xi]$. Let $\mathfrak{B}$ be the behavior represented by $R_1(\frac{d}{dt})w_1 + R_2(\frac{d}{dt})w_2 = 0$. Then the variable w_2 is observable from w_1 if and only if $rank\,(R_2(\lambda)) = \mathtt{w}_2$ for all $\lambda \in \mathbb{C}$, equivalently, $R_2(\lambda)$ has full column rank for all $\lambda \in \mathbb{C}$.*

Relaxing the constraint of full, row rank for all $\lambda \in \mathbb{C}$ to full row rank for all $\lambda \in \overline{\mathbb{C}}^+$ (the closed right half complex plane) in the last definition, we obtain the concept of detectability from our viewpoint. Theorem 8 below will provide the way of testing this property.

Definition 10 *Let $\Sigma = (\mathbb{R}, \mathbb{W}_1 \times \mathbb{W}_2, \mathfrak{B})$ be a linear differential system. The trajectories in $\mathfrak{B}$ are partitioned as (w_1, w_2) with $w_i : \mathbb{R} \to \mathbb{W}_i$,$i = 1, 2$. We say that w_2 is detectable from w_1 if $(w_1, w_2), (w_1, w_2') \in \mathfrak{B}$ implies $\lim_{t\to\infty}(w_2 - w_2')(t) = 0$.*

Theorem 8 *Let $R_1(\xi) \in \mathbb{R}^{g\times \mathtt{w}_1}[\xi]$ and $R_2(\xi) \in \mathbb{R}^{g\times \mathtt{w}_2}[\xi]$. Let $\mathfrak{B}$ be the behavior defined by $R_1\left(\frac{d}{dt}\right)\mathtt{w}_1 + R_2\left(\frac{d}{dt}\right)\mathtt{w}_2 = 0$. Then the variable w_2 is detectable from w_1 if and only if $rank\,(R_2(\lambda)) = \mathtt{w}_2$ for all $\lambda \in \overline{\mathbb{C}}^+$, equivalently, $R_2(\lambda)$ has full column rank for all λ in the closed right half complex plane.*

2.6 Controllability and stabilizability

An important concept in the analysis and synthesis of open dynamical systems is the concept of controllability, which deals with modifying the conduct a system exhibits. Suppose initially our system works in some given mode of operation, considered as undesirable. Then, the possibility of transferring our system from that mode to another one, referred

to as a desired mode, reflects the ability of controlling the plant we have at hand. This property of controllability was originally introduced in the context of state space systems. It is one of the central notions of control theory.
Let us recall the classical definition we have for this concept. The system described by $\dot{x} = f(x,u)$ (where f is a given function from $\mathbb{R}^n \times \mathbb{R}^m$ to $\mathbb{R}^n$) is said to be controllable if for all $a, b \in \mathbb{R}^n$ there exists $T > 0$ and a a function u from $[0,T]$ to $\mathbb{R}^m$ such that the solution x of $\dot{x} = f(x,u), x(0) = a$ yields $x(T) = b$. As is well known, if the system is linear and time invariant, i.e., it is given by $\dot{x} = Ax + Bu$ for appropriate matrices A and B, then the controllability test is checking that the matrix $[B\ AB\ A^2B\ \cdots A^{n-1}B]$ has full row rank. Although generalizations exist (for time varying, nonlinear systems for instance), all the notions of controllability discussed above have the important drawback of being representation dependent. We should however realize that a system may be uncontrollable either for the intrinsic reason that the control signal u can not affect the system variables, or because the state was chosen inefficiently. The behavioral approach offers the following definition of controllability in terms of trajectories in the system behavior rather than in terms of one of its possible representations.

Definition 11 *Let $\Sigma = (\mathbb{R}, \mathbb{R}^{\mathbb{w}}, \mathfrak{B})$ be a linear differential system. Σ is said to be controllable if for all $w_1,\ w_2 \in \mathfrak{B}$ there exists $T \in \mathbb{R}$, $T \geq 0$ and $w \in \mathfrak{B}$ such that $w(t) = w_1(t)$ for $t < 0$ and $w(t) = w_2(t-T)$ for $t \geq T$. Thus controllability refers to the ability to switch from any one trajectory in the behavior to any other one, allowing some time delay.*

Of course, if the system happens to be given in terms of a particular representation, we would like to have tests in order to decide whether the system is controllable. Also we would like to know whether controllable systems admit a special representation where this property becomes evident. For linear differential systems, the following theorem [89] deals with the aspects just mentioned.

Theorem 9 *Let $\Sigma = (\mathbb{R}, \mathbb{R}^{\mathbb{w}}, \mathfrak{B}) \in \mathcal{L}^{\mathbb{w}}$. Let $R(\frac{\mathrm{d}}{\mathrm{dt}})w = 0$ be a kernel representation of $\mathfrak{B}$, with $R(\xi) \in \mathbb{R}^{\bullet \times \mathbb{w}}[\xi]$. Then following statements are equivalent:*

a. The system Σ is controllable.

b. The polynomial matrix R has the property that $\operatorname{rank}(R(\lambda)) = \operatorname{rank}(R)$ for all $\lambda \in \mathbb{C}$.

c. There exists an integer ℓ and a polynomial matrix $M(\xi) \in \mathbb{R}^{\mathbb{w} \times \ell}[\xi]$ such that $\mathfrak{B}$ is the image of the differential operator $M(\frac{\mathrm{d}}{\mathrm{dt}})$, that is $\mathfrak{B} = M\left(\frac{d}{dt}\right) \mathfrak{C}^\infty(\mathbb{R}, \mathbb{R}^\ell)$.

A remarkable point of theorem 9 is that a controllable system allows a representation in terms of a latent variable of the form

$$w = M\left(\frac{d}{dt}\right)\ell \tag{2.8}$$

Equation 2.8 is called an *image representation* of $\mathfrak{B}$.

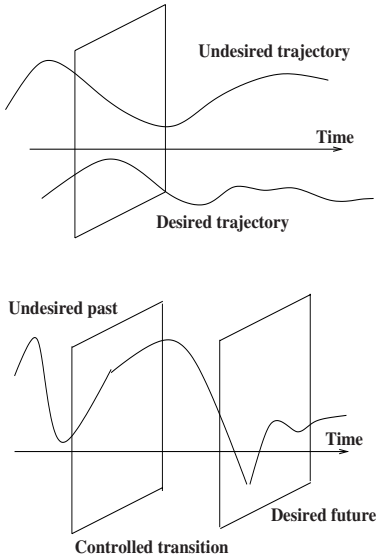

Figure 2.8: *Controllability.*

We now turn to the notion of stabilizability. This notion is concerned with the situation that we are on a given trajectory of the given behavior $\mathfrak{B}$ and we want switch to a trajectory that asymptotically tends to zero, while remaining on a trajectory within the behavior. The possibility of doing this for every trajectory in the behavior is a less severe condition on our system behavior than that of controllability.

Definition 12 *Let $\mathfrak{B} \in \mathcal{L}^{\mathfrak{w}}$. This system is called stabilizable if for every trajectory $w \in \mathfrak{B}$, there exists a trajectory $w_1 \in \mathfrak{B}$ with the property that $w_1(t) = w(t)$, $t \leq 0$ and $lim_{t\to\infty} w_1(t) = 0$.*

Of course, we also want a mean to test stabilizability of a given behavior $\mathfrak{B}$ in terms of its kernel representations. The way to do that is as follows.

Theorem 10 *Let $R(\xi) \in \mathbb{R}^{g\times \mathfrak{w}}[\xi]$. The behavior $\mathfrak{B}$ represented by $R\left(\frac{d}{dt}\right) w = 0$ is stabilizable if and only if* $\operatorname{rank}(R(\lambda)) = \operatorname{rank}(R)$ *for all $\lambda \in \overline{\mathbb{C}^+}$.*

2.7 Autonomous behaviors

As we know, controllability deals with the possibility to switch from any trajectory within the behavior to any other trajectory within that behavior. In general, for a given behavior, this is not possible. The extreme case of this is when for every given trajectory, its

future continuation is completely determined by its past. In that case we talk about an autonomous behavior.

Definition 13 *A linear differential system* $\Sigma = (\mathbb{R}, \mathbb{R}^{\mathtt{w}}, \mathfrak{B})$ *is called autonomous if for all* $w_1,\ w_2 \in \mathfrak{B}$ *we have:* $w_1(t) = w_2(t),\ t \leq 0 \Rightarrow\ w_1 = w_2$.

Assume now that system behavior is represented by a kernel representation. We would like to have conditions on the underlying polynomial matrix for the behavior to be autonomous. Such a condition is provided in the following theorem:

Theorem 11 *Let* $\mathfrak{B} \in \mathcal{L}^w$ *and let* $R(\xi) \in \mathbb{R}^{\bullet \times \mathtt{w}}[\xi]$ *induce a kernel representation* $R(\frac{d}{dt})w = 0$ *of* $\mathfrak{B}$. *Then the following statements are equivalent:*

a. $\mathfrak{B}$ *is autonomous,*

b. $\operatorname{rank}(R(\xi)) = \mathtt{w}$, *i.e.* $R(\xi)$ *has full column rank,*

c. $\mathfrak{B}$ *is a finite dimensional vector space.*

When $R(\frac{d}{dt})w = 0$ is a minimal kernel representation of $\mathfrak{B}$ then any of the latter statements is equivalent to the existence of a square polynomial matrix $R(\xi)$ with $\det((R(\xi))) \neq 0$. For autonomous behaviors we have the notion of *characteristic polynomial*. Let $\mathfrak{B}$ be autonomous and let $R(\xi)$ be a $\mathtt{w} \times \mathtt{w}$ polynomial matrix with $\det(R(\xi)) \neq 0$ such that $\mathfrak{B}$ is represented by $R(\frac{\mathrm{d}}{\mathrm{dt}})w = 0$. Of course, for any non-zero $c \in \mathbb{R}$, the polynomial matrix $cR(\xi)$ also yields a kernel representation of $\mathfrak{B}$. Hence we can choose $R(\xi)$ such that $\det(R(\xi))$ has leading coefficient equal to 1. This monic polynomial is denoted by $\chi_{\mathfrak{B}}(\xi)$ and is called *the characteristic polynomial of* $\mathfrak{B}$. $\chi_{\mathfrak{B}}(\xi)$ depends only on $\mathfrak{B}$, and not on the polynomial matrix $R(\xi)$ we used to define it. If $R_1(\xi)$, $R_2(\xi)$ both represent $\mathfrak{B}$ minimally then there exists a unimodular $U(\xi)$ such that $R_2(\xi) = U(\xi)R_1(\xi)$. Consequently, if $\det(R_1(\xi))$ and $\det(R_2(\xi))$ are monic then $\det(R_1(\xi)) = \det(R_2(\xi))$. Finally, the roots of the characteristic polynomial are called the *poles* of the autonomous behavior $\mathfrak{B}$.

2.8 Defining inputs and outputs

As we have pointed out up to now, the behavioral point of view is a representation free approach and hence is not subjected to any special framework to explain the way a system interacts with its environment. Nevertheless, as a particular case, a subset of the system variables may be defined as inputs while another subset of variables may be considered as the outputs of the system. In this way, the input/output paradigm becomes (as expected) a special case of our more general set up. Due to the fact that the latter approach is very important, we want to find the link between the two points of view. For instance, the behavioral approach takes into account the possibility of having unconstrained system variables, called *free variables* in our argot. These occur if the system does not impose any restriction on them. Such variables will be called *inputs* of the system. From section

2.7, we can see that behaviors represented by a square full row rank polynomial matrix are autonomous. In general, a polynomial matrix that represents the system will be nonsquare. This means that there are more system variables than equations, in other words we have an underdetermined system of equations. As a consequence, some of the variables remain unconstrained, they will be inputs. Once the inputs are chosen, together with the initial conditions they will determine the values of the remaining variables of the system. These are then the *outputs*. The following definition summarizes these ideas (see [58]).

Definition 14 *Let $\Sigma = (\mathbb{R}, \mathbb{R}^{\mathtt{w}}, \mathfrak{B})$ be a linear differential system. Partition the signal space as $\mathbb{R}^{\mathtt{w}} = \mathbb{R}^{\mathtt{m}} \times \mathbb{R}^{\mathtt{p}}$ and partition w correspondingly as $w = [w_1, w_2]$. This partition is called an input/output partition if:*

a. *w_1 is free, i.e., for all $w_1 \in \mathfrak{C}^\infty(\mathbb{R}, \mathbb{R}^{\mathtt{m}})$, there exists a $w_2 \in \mathfrak{C}^\infty(\mathbb{R}, \mathbb{R}^{\mathtt{p}})$ such that $[w_1, w_2] \in \mathfrak{B}$.*

b. *w_2 does not contain any further free components, i.e., given w_1, none of the components of w_2 can be chosen freely.*

If these conditions hold then we also say: w_1 is maximally free. If both conditions above are satisfied, then w_1 is called an input variable (and often denoted as u) and w_2 is called an output variable (often denoted as y).

If u is maximally free then y does not contain any free components. In this case, y is called often called *bound*. On the other hand, when we have a kernel representation of a dynamical system, the following theorem gives conditions for an input/output partition of w.

Theorem 12 *Let $R(\xi) \in \mathbb{R}^{g \times \mathtt{w}}[\xi]$ induce a kernel representation of $\mathfrak{B}$. Let $w = [u, y]$ be a partition of w and let $R(\xi) = [Q(\xi) \;\; P(\xi)]$ be a corresponding partition of $R(\xi)$. Then,*

a. *u is free if and only if rank$([Q(\xi) \;\; P(\xi)]) =$ rank$(P(\xi))$;*

b. *once u is fixed, y is bound if and only if $P(\xi)$ has full column rank, rank$(P(\xi)) =$ dim(y);*

c. *$w = [u, y]$ is an i/o (input/output) partition if and only if rank$(R(\xi)) =$ rank$(P(\xi)) =$ coldim$(P(\xi))$.*

If we choose $R(\xi)$ such that $R\left(\frac{d}{dt}\right) w = 0$ is a minimal kernel representation, then $w = [u, y]$ is an input/output partition if and only if $P(\xi)$ is square and nonsingular. In that case the system is represented by $Q\left(\frac{d}{dt}\right) u + P\left(\frac{d}{dt}\right) y = 0$. Then $-P(\xi)^{-1}Q(\xi)$ defines the transfer function of $\mathfrak{B}$. Inputs and outputs determine a set of invariants linked to a given behavior $\mathfrak{B}$, see [58]. These invariants are defined as follows:

a. $\mathtt{w}(\mathfrak{B})$ is the number of components of the system variable w of $\mathfrak{B}$.

b. $\mathtt{m}(\mathfrak{B})$ is the number of input variables in any input/output partition of the system variables. This number is called *the input cardinality* of $\mathfrak{B}$.

c. $\mathtt{p}(\mathfrak{B})$ is the number of outputs in any input/output partition of the system variable w. This number is called *the output cardinality* of $\mathfrak{B}$. Obviously, $\mathtt{w}(\mathfrak{B}) = \mathtt{p}(\mathfrak{B}) + \mathtt{m}(\mathfrak{B})$.

Later on in this book we will need to be able to compute the output cardinality of a system behavior represented by a latent variable representation. Assume that the behavior $\mathfrak{B} \in \mathcal{L}^{\mathtt{w}}$ is represented by

$$R\left(\frac{d}{dt}\right) w = M\left(\frac{d}{dt}\right) \ell,$$

where ℓ is a latent variable. Then we have the following (see [8]):

Lemma 1 $\mathtt{p}(\mathfrak{B}) = rank([R \;\; M]) - rank(M)$.

Of course image representations (see 2.8) are a particular case that the latter lemma applies to.

2.9 Controllable part of a behavior

Definition 15 *Let $\mathfrak{B} \in \mathcal{L}^{\mathtt{w}}$. The controllable part of a behavior $\mathfrak{B}$ is defined as the behavior $\mathfrak{B}_{cont} \in \mathcal{L}^{\mathtt{w}}$ satisfying the following three properties:*

a. $\mathfrak{B}_{cont}$ is controllable.

b. $\mathfrak{B}_{cont} \subseteq \mathfrak{B}$.

c. If $\mathfrak{B}_1 \in \mathcal{L}^w$ is controllable and if $\mathfrak{B}_1 \subseteq \mathfrak{B}$ then $\mathfrak{B}_1 \subseteq \mathfrak{B}_{cont}$.

It can be proven that for every linear differential behavior such $\mathfrak{B}_{cont}$ exists, and that it is unique. Every behavior $\mathfrak{B} \in \mathcal{L}^{\mathtt{w}}$ can be written as a direct sum of the controllable part and an autonomous part, see [58]:

Theorem 13 *Let $R(\xi) \in \mathbb{R}^{g \times \mathtt{w}}[\xi]$ be of full row rank and let $\mathfrak{B}$ be the behavior defined by*

$$R\left(\frac{d}{dt}\right) w = 0$$

Then there exists an autonomous subbehavior $\mathfrak{B}_{aut}$ of $\mathfrak{B}$ such that

$$\mathfrak{B} = \mathfrak{B}_{aut} \oplus \mathfrak{B}_{cont}$$

The poles of $\mathfrak{B}_{aut}$ are exactly those values $\lambda \in \mathbb{C}$ for which $\operatorname{rank}(R(\lambda)) < g$. $R(\xi)$ admits a factorization $R(\xi) = D(\xi)R_1(\xi)$ with $D(\xi)$ square and nonsingular, and $R_1(\lambda)$ full row rank for all $\lambda \in \mathbb{C}$. For any such factorization, the controllable part $\mathfrak{B}_{cont}$ is represented by $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$.

We note that the controllable part is unique, however there are in general *many* autonomous subbehaviors $\mathfrak{B}_{aut}$ such that $\mathfrak{B} = \mathfrak{B}_{aut} \oplus \mathfrak{B}_{cont}$.
Traditionally, this theorem is proved with the help of the Smith form, [58]. The latter provides a way to compute the associated polynomial matrices representing the behaviors just described. However, as we shall show in the coming chapters, this popular decomposition is not numerically reliable. Since in this work we are interested in numerical consequences of our polynomial approach, we can replace the Smith form method by so called column compressions and *embeddings*. The latter term will be formally defined in chapter 5.

2.10 Interconnection of dynamical systems

We now discuss interconnection of dynamical systems. As we have shown, we do not need to choose any special structure, like an input/output structure, to define our dynamical systems. Rather, since our systems interact with their environment via terminals, interconnecting systems via those terminals will be important. More concretely, let us consider the following two dynamical systems Σ_1 and Σ_2 given by

$$\Sigma_1 = (\mathbb{T}, \mathbb{W}_1 \times \mathbb{W}_2, \mathfrak{B}_1)\,, \quad \Sigma_2 = (\mathbb{T}, \mathbb{W}_2 \times \mathbb{W}_3, \mathfrak{B}_2) \tag{2.9}$$

with common axis time $\mathbb{T}$ and common factor $\mathbb{W}_2$ in the signal spaces. The interconnection of Σ_1 and Σ_2 is defined as the dynamical system

$$\Sigma_1 \wedge_{w_2} \Sigma_2 = (\mathbb{T}, \mathbb{W}_1 \times \mathbb{W}_2 \times \mathbb{W}_3, \mathfrak{B}) \tag{2.10}$$

with behavior $\mathfrak{B}$ is given by

$$\mathfrak{B} = \{(w_1, w_2, w_3) \mid (w_1, w_2) \in \mathfrak{B}_1,\ (w_2, w_3) \in \mathfrak{B}_2\} \tag{2.11}$$

The shared variable w_2 is referred to as *interconnection variable*. The system variable of the interconnection of Σ_1 and Σ_2 is (w_1, w_2, w_3). Sometimes, after interconnection, the interconnection variable w_2 is considered as a latent variable, and we are only interested in the behavior of the two remaining variables w_1 and w_3. This behavior can be obtained by eliminating w_2:

$$\mathfrak{B}_{w_1,w_3} = \{(w_1, w_3) \mid \exists\ w_2 \text{ such that } (w_1, w_2, w_3) \in \mathfrak{B}\}\,. \tag{2.12}$$

The ideas exposed here will be helpful to introduce, in the coming section, the notion of control by behavioral interconnection.

2.10.1 Control as interconnection

As we explained at the beginning of this chapter, the behavioral approach considers a system to interact with its environment via terminal variables. This interaction helps to define a system behavior as the set of trajectories permitted by the physical laws of the system.

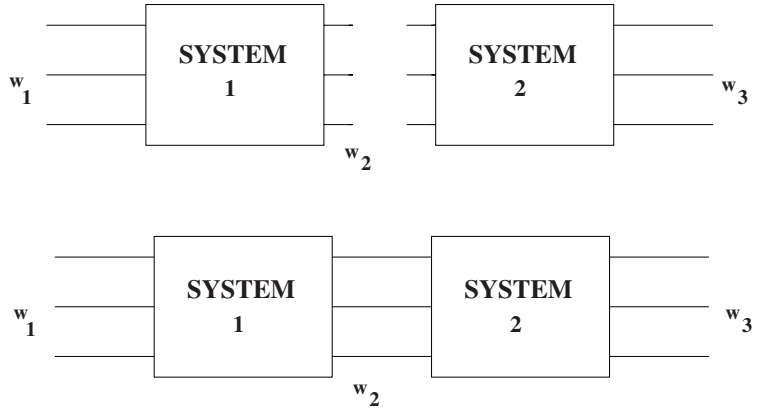

Figure 2.9: *Interconnection of systems.*

Nevertheless, we - as designers - do not want to be passive observers of what is happening with our system dynamics. Rather, we would like to modify the behavior of our system so that it satisfies our imposed requirements. The way to alter the behavior of our system is by means of another system (referred to as the controller) which is connected to our given system (the plant). Notice that we do not specify any special type of interconnection, like feedback. The way the systems interact is via the terminal variables. The terminals of the plant consist of *to be controlled variables* w and the *control terminals* c. The controller has only one type of variables which are precisely the control terminal variables c, see figure 2.10. Before interconnecting our plant to the controller, the trajectories w and c obey the restrictions imposed by the plant behavior. The space of all these trajectories is what we call *the full plant behavior*, defined formally as

$$\mathcal{P}_{\text{full}} = \big\{(w,c) \in \mathfrak{C}^{\infty}(\mathbb{R}, \mathbb{R}^{\mathfrak{w}+k}) \mid (w,c) \text{ satisfies the plant equations}\big\} \tag{2.13}$$

The *manifest plant behavior*, denoted by $(\mathcal{P}_{\text{full}})_w$, defines the to be controlled variables w. It is defined as

$$(\mathcal{P}_{\text{full}})_w = \{(w) \in \mathfrak{C}^{\infty}(\mathbb{R}, \mathbb{R}^{\mathfrak{w}}) | \exists\ c \text{ such that } (w,c) \in \mathcal{P}_{\text{full}}\} \tag{2.14}$$

In this work we deal with linear differential systems. Therefore we assume $\mathcal{P}_{\text{full}} \in \mathcal{L}^{\mathfrak{w}+k}$. The manifest plant behavior is obtained by eliminating c from $\mathcal{P}_{\text{full}}$, i.e., $(\mathcal{P}_{\text{full}})_w := \Pi_w(\mathcal{P}_{\text{full}})$. From the elimination theorem (Theorem 1) we can conclude that also $(\mathcal{P}_{\text{full}})_w \in \mathcal{L}^{w}$. On the

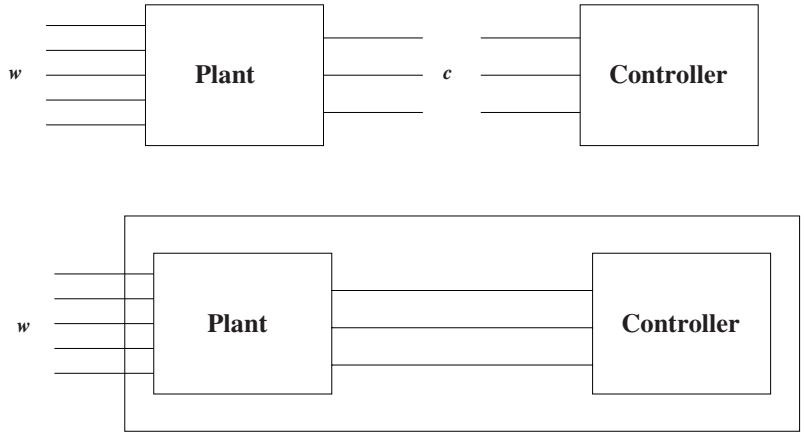

Figure 2.10: *Controller interconnection.*

other hand, as we mentioned above the controller has only one type of terminal variables, c. We define the *controller behavior* $\mathcal{C} \in \mathcal{L}^k$ as

$$\mathcal{C} = \big\{c \in \mathfrak{C}^\infty(\mathbb{R}, \mathbb{R}^k) \mid c \text{ satisfies the controller equations}\big\} \tag{2.15}$$

Once the blocks are interconnected, the restrictions imposed on c by the controller will be transmitted to the variables w. Choosing the controller in such a way that the variables w have a desired behavior in the interconnected resultant box (lower part of figure 2.10) is the basic problem of control within our perspective. Then when the systems interact, we obtain a new system in which c satisfies both the laws of the plant and the laws required by the controller. This means that c will be governed by $\mathcal{P}_{\text{full}}$ and $\mathcal{C}$. This fact determines a new behavior which appears as the result of interconnecting the controller to the plant and it is called the *full controlled behavior*. This interconnection behavior is given by

$$\mathcal{K}_{\text{full}}(\mathcal{C}) = \big\{(w, c) \in \mathfrak{C}^\infty(\mathbb{R}, \mathbb{R}^{\mathtt{w}+k}) \mid (w, c) \in \mathcal{P}_{\text{full}} \text{ and } c \in \mathcal{C}\big\} \tag{2.16}$$

The latter behavior represents our controlled plant. Since the interconnection only takes place through the variable c and not through the whole system variable (w, c), we call this *partial interconnection through c*. As we mentioned in this section before, after interconnection, the control variable is often considered as a latent variable. We might get rid of this variable by applying the elimination theorem. This then defines the *manifest control behavior*, defined as follows:

$$(\mathcal{K}_{\text{full}}(\mathcal{C}))_w = \{w \in \mathfrak{C}^\infty(\mathbb{R}, \mathbb{R}^{\mathtt{w}}) \mid \exists\ c \text{ such that } (w, c) \in \mathcal{K}_{\text{full}}(\mathcal{C})\} \tag{2.17}$$

With the definitions given above, we can now formulate the general control problem from the behavioral point of view:

Problem 8 *Given the full plant behavior* $\mathcal{P}_{\text{full}}$*:*

a. *Describe a set of specifications on the controlled plant, namely, the desired properties of the manifest controlled behavior (the desired controlled behavior).*

b. *Find a controller behavior* $\mathcal{C}$ *such that the corresponding manifest controlled behavior* $(\mathcal{K}_{\text{full}}(\mathcal{C}))_w$ *satisfies these specifications (limit of performance).*

In fact, the main idea is to make $(\mathcal{K}_{\text{full}}(\mathcal{C}))_w$ equal to a behavior $\mathcal{K}$ that is desired. If $\mathcal{K} \in \mathcal{L}^{\mathtt{w}}$, then if a controller behavior $\mathcal{C}$ exists such that $\mathcal{K} = (\mathcal{K}_{\text{full}}(\mathcal{C}))_w$ is accomplished, i.e the desired behavior becomes equal to the manifest controlled behavior, then we say that $\mathcal{C}$ *implements* $\mathcal{K}$ *by partial interconnection through c* or, equivalently, that $\mathcal{K}$ is *implemented by* $\mathcal{C}$ *by partial interconnection through c*. If a behavior $\mathcal{K} \in \mathcal{L}^{\mathtt{w}}$ has the property that there exists a controller $\mathcal{C}$ such that $\mathcal{K} = (\mathcal{K}_{\text{full}}(\mathcal{C}))_w$, then we call $\mathcal{K}$ *implementable by partial interconnection (through c)*.
In the following we will study the question which behaviors $\mathcal{K} \in \mathcal{L}^{\mathtt{w}}$ are implementable by partial interconnection, i.e. for which behaviors $\mathcal{K}$ there exists a controller $\mathcal{C} \in \mathcal{L}^{\mathtt{c}}$ such that $\mathcal{K} = (\mathcal{K}_{\text{full}}(\mathcal{C}))_w$. We shall see that the answer to this question is very intuitive. It will depend only on the manifest plant behavior $(\mathcal{P}_{\text{full}})_w$ and on the behavior consisting of the plant trajectories with the control variables set to zero. This behavior is referred to as *the hidden behavior* is denoted by $\mathcal{N}$. It is defined as follows:

Definition 16 *Let* $\mathcal{P}_{\text{full}} \in \mathcal{L}^{\mathtt{w}+k}$*. The hidden behavior* $\mathcal{N} \in \mathcal{L}^{\mathtt{w}}$ *is the behavior consisting of the to-be- controlled variable trajectories that can occur when the control variables are set to zero:*

$$\mathcal{N} = \{w \in \mathfrak{C}^{\infty}(\mathbb{R}, \mathbb{R}^{\mathtt{w}}) \mid (w, 0) \in \mathcal{P}_{\text{full}}\} \tag{2.18}$$

As we see from 2.17, $\mathcal{N} \subseteq (\mathcal{P}_{\text{full}})_w$. Of course, there are many behaviors that are wedged in between $\mathcal{N}$ and $(\mathcal{P}_{\text{full}})_w$. Obviously, any manifest controlled behavior $(\mathcal{K}_{\text{full}}(\mathcal{C}))_w$ must be contained in the manifest plant behavior $(\mathcal{P}_{\text{full}})_w$. Also, any manifest controlled behavior $(\mathcal{K}_{\text{full}}(\mathcal{C}))_w$ must contain the hidden behavior $\mathcal{N}$, since if the controller receives no information about what is happening in the plant, $\mathcal{N}$ must remain possible in the controlled behavior, independently of the controller we have. Now, it turns out that also the converse holds. This fact is called the *controller implementability theorem*, which is stated as follows (see [91]).

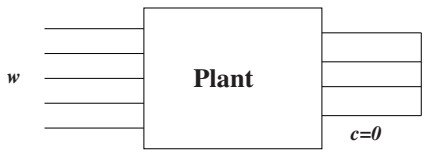

Figure 2.11: *The hidden behavior.*

Theorem 14 *Let $\mathcal{P}_{\text{full}} \in \mathcal{L}^{\mathtt{w}+k}$ be the full plant behavior, $(\mathcal{P}_{\text{full}})_w \in \mathcal{L}^{\mathtt{w}}$ the manifest plant behavior and $\mathcal{N} \in \mathcal{L}^{\mathtt{w}}$ the hidden behavior. Then $\mathcal{K} \in \mathcal{L}^{\mathtt{w}}$ is implementable by a controller $\mathcal{C} \in \mathcal{L}^{\mathtt{c}}$ acting on the control variables if and only if $\mathcal{N} \subseteq \mathcal{K} \subseteq (\mathcal{P}_{\text{full}})_w$.*

As we saw, the latter theorem gives conditions for implementability of desired behavior $\mathcal{K}$. This facts allows us to rephrase the general control problem (posed above as problem 8) as

Problem 9 *Given the full plant behavior $\mathcal{P}_{\text{full}}$:*

a. *Describe a set of specifications on the controlled plant, namely, the desired properties of the manifest controlled behavior $\mathcal{K}$ (the desired controlled behavior).*

b. *Find a controlled behavior $\mathcal{K}$ that satisfies these specifications and that is implementable by partial interconnection, i.e., $\mathcal{N} \subseteq \mathcal{K} \subseteq (\mathcal{P}_{\text{full}})_w$.*

In a sense the controller implementability theorem characterizes the *limits of performance* of the given full plant behavior $\mathcal{P}_{\text{full}}$: it exactly tells which controlled system behaviors can be obtained.

Chapter 3

Full Interconnection Issues

As we explained in chapter 2, the behavioral approach considers systems that interact with their environment through their terminals. After a controller is interconnected to the plant, the variables living on the terminals have to obey both the laws of the plant and the controller. The plant and the controller may or may not have an input/output structure when viewed from the interfacing variables. From a general point of view, the behavioral approach is not restricted to an input/output scheme for modelling and control. An important special case when interconnecting systems (e.g. plant and controller) is the so called *full interconnection case*, i.e., the situation that all the variables are available for interconnection. In this case, all terminals are used to interconnect the systems.

In this chapter we will study a number of synthesis problems that occur in the behavioral approach to control. In particular, we will establish algorithms for the numerical computation of controllers that achieve pole placement and stabilization by behavioral *full* interconnection. These synthesis problems were studied before in [9]. We stress that in the present chapter we only deal with the *full interconnection case*. In chapter 4 we will then deal with the partial interconnection case.

However, before discussing these synthesis problems, in this chapter we will first recall the notions of implementability and regular implementability by full interconnection. Given a full plant behavior, a given behavior is called *implementable* if it can be achieved as controlled behavior by interconnecting the plant with a controller. It is called regularly implementable if the controller that achieves the desired behavior does not re-impose restrictions that are already present in the plant behavior. We will present a new condition for regular implementability of a given behavior. Also, numerical algorithms will be presented to check regular implementability.

An important issue is also the problem of finding, for a given regularly implementable subbehavior of the plant behavior, *a parametrization* of all controllers that regularly implement this subbehavior. In the present chapter we will establish such parametrization, again for the full interconnection case. Also, an algorithm will be given to compute such a parametrization. The extension of this parametrization and of the related algorithm to the partial interconnection case will be given in chapter 4.

Finally, again for the full interconnection case, we will establish a parametrization of all

R.Z. Yoe: *Modelling and Control of Dynamical Systems: Numerical Implementation in a Behavioral Framework*, Studies in Computational Intelligence (SCI) **124**, 35–58 (2008)
www.springerlink.com

controllers that stabilize a given plant behavior, thus establishing the behavioral version of the Youla parametrization of all stabilizing controllers that was originally obtained in [93]. We will also present an algorithm to compute such parametrization. Clearly we will need to stack matrices frequently. To save space we will denote that by col(P,Q) when we provide a theorem, lemma or their corresponding proof. In our algorithms and programs we denote that by a semicolon in line with the MATLAB notation:

$$[P;Q] := \begin{bmatrix} P \\ Q \end{bmatrix}.$$

3.1 Implementability

Implementability deals with the issue which system behaviors can be achieved ('implemented') by interconnecting a given system behavior with a controller, and is thus concerned with the limits of performance of a given plant In the contexts of synthesis of dissipative systems [91], pole placement and stabilization , an important role is also played by the notion of *regular implementability*. A given behavior is called regularly implementable if it can be achieved by a controller behavior that does not impose restrictions on the control variable that are already present in the plant, equivalently, the number of outputs of the associated full controlled behavior is equal to the sum of the number of outputs of the plant and the number of outputs of the controller. In [91], for a given plant behavior a characterization was given of all implementable behaviors and in [9] a characterization was given of all regularly implementable behaviors.
In this section we will briefly review the notions of implementability and regular implementability for the full interconnection case. We will also present a new condition for regular implementability. Finally, we will establish two algorithms to check whether a a given desired subbehavior is regularly implementable, and if it is, to compute a suitable controller.
Let $\mathcal{P} \in \mathcal{L}^q$ be a plant behavior. A controller for $\mathcal{P}$ is a system behavior $\mathcal{C} \in \mathcal{L}^q$. The *full interconnection* of $\mathcal{P}$ and $\mathcal{C}$ is defined as the system which has the intersection $\mathcal{P} \cap \mathcal{C}$ as its behavior. This *controlled behavior* is again an element of $\mathcal{L}^q$. The full interconnection is called *regular* if

$$\mathtt{p}(\mathcal{P} \cap \mathcal{C}) = \mathtt{p}(\mathcal{P}) + \mathtt{p}(\mathcal{C}).$$

Let $\mathcal{K} \in \mathcal{L}^q$ be a given behavior, to be interpreted as a 'desired' behavior. If $\mathcal{K}$ can be achieved as controlled behavior, i.e. if there exists $\mathcal{C} \in \mathcal{L}^q$ such that $\mathcal{K} = \mathcal{P} \cap \mathcal{C}$, then we call $\mathcal{K}$ *implementable by full interconnection* (with respect to $\mathcal{P}$). If $\mathcal{K}$ can be achieved by regular interconnection, i.e. if there exists $\mathcal{C}$ such that $\mathcal{K} = \mathcal{P} \cap \mathcal{C}$ and $\mathtt{p}(\mathcal{P} \cap \mathcal{C}) = \mathtt{p}(\mathcal{P}) + \mathtt{p}(\mathcal{C})$, then we call $\mathcal{K}$ *regularly implementable by full interconnection*.
Obviously, a given $\mathcal{K} \in \mathcal{L}^q$ is implementable by full interconnection with respect to $\mathcal{P}$ if and only if $\mathcal{K} \subseteq \mathcal{P}$. Indeed, if $\mathcal{K} \subseteq \mathcal{P}$ then with 'controller' $\mathcal{C} = \mathcal{K}$ we have $\mathcal{K} = \mathcal{P} \cap \mathcal{C}$. Thus, if $\mathcal{P}$ and $\mathcal{K}$ have minimal kernel representations $R(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ and $K(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$, respectively, then $\mathcal{K}$ is implementable with respect to $\mathcal{P}$ if and only if there exists a polynomial matrix

F such that $R = FK$ (see [58]). Before going deeper into this, we need to provide. We now propose a new condition for regular implementability, which says that this property is equivalent with the existence of a polynomial matrix F with, in addition, $F(\lambda)$ full row rank for all $\lambda \in \mathbb{C}$. However, before doing that, we need to review a very important concept we work with: *minimal left/right annihilators.*

3.1.1 Minimal Annihilators of a Polynomial Matrix

If M is a polynomial matrix, then the polynomial matrix R is called a minimal left annihilator (MLA) of M if $\mathrm{im}(M) = \ker(R)$. For a given polynomial matrix R, the polynomial matrix M is called a minimal right annihilator (MRA) of R if $\mathrm{im}(M) = (\ker(R))_{\mathrm{cont}}$. The following useful fact is well known:

Theorem 15 *Let R and M be polynomial matrices. R is an MLA of M if and only if $RM = 0$, $R(\lambda)$ has rank independent of λ for $\lambda \in \mathbb{C}$ and* $\mathrm{rank}(R) = \mathrm{rowdim}(M) - \mathrm{rank}(M)$.

In this paper, in many places we will use, for a given M, a MLA of R *with full row rank*. If the given M has full row rank, say q, then for consistency we *define* such full row rank MLA as the 'void' matrix R with 0 rows and q columns. Likewise we often use, for a given R, a MRA with the property that $M(\lambda)$ has full column rank for all λ. If R has full column rank q we define such M to be the void matrix with q rows and 0 columns. In that case, if K is a given matrix with q columns, then KM is again void. Finally, we use the convention that if R is void, with zero rows and q columns, then a full column rank MRA if given by the $q \times q$ identity matrix I_q.
We will also need some facts on the representation of sums of behaviors. It is well known that the space of all linear differential systems $\mathcal{L}^q$ is closed under addition. Suppose that $\mathfrak{B}_1, \mathfrak{B}_2 \in \mathcal{L}^{\mathfrak{q}}$, where $\mathfrak{B}_1$ and $\mathfrak{B}_2$ have kernel representations $R_1(\frac{\mathrm{d}}{\mathrm{dt}})w = 0$ and $R_2(\frac{\mathrm{d}}{\mathrm{dt}})w = 0$, respectively. The problem to find a kernel representation of $\mathfrak{B}_1 + \mathfrak{B}_2$ was solved in [65] (see also [11]):

Proposition 10 : *Let $[S_1 \;\; S_2]$ be a MLA of* $\mathrm{col}(R_1, R_2)$. *Then the polynomial matrix $S_1R_1 = -S_2R_2$ yields a kernel representation of $\mathfrak{B}_1 + \mathfrak{B}_2$*

In the following theorem we now give a new condition for regular implementability.

Theorem 11 : *Let $\mathcal{P}, \mathcal{K} \in \mathcal{L}^q$. Let $R(\frac{\mathrm{d}}{\mathrm{dt}})w = 0$ and $K(\frac{\mathrm{d}}{\mathrm{dt}})w = 0$ be minimal kernel representations of $\mathcal{P}$ and $\mathcal{K}$, respectively. Then the following statements are equivalent:*

1. *$\mathcal{K}$ is regularly implementable w.r.t. $\mathcal{P}$ by full interconnection,*
2. *there exists a polynomial matrix F with $F(\lambda)$ full row rank for all $\lambda \in \mathbb{C}$, such that $R = FK$,*
3. *$\mathcal{K} + \mathcal{P}_{\mathrm{cont}} = \mathcal{P}$.*

Here, $\mathcal{P}_{\rm cont}$ denotes the controllable part of $\mathcal{P}$.

Proof : The equivalence of 1. and 3. was proven in [9]. Here, we will give a new proof, passing through the condition 2.
(1. $\Rightarrow$ 2.) Suppose there exists a polynomial matrix C such that

$$\left[\begin{array}{c} R(\frac{\rm d}{{\rm d}t}) \\ C(\frac{\rm d}{{\rm d}t}) \end{array} \right] w = 0$$

is a minimal kernel representation of $\mathcal{K}$. Then there exists a unimodular U such that $\mathrm{col}(R, C) = UK$. This implies $R = U_1 K$, with U_1 consisting of the upper rows of U.
(2. $\Rightarrow$ 1.) Assume $R = FK$. Let U_2 be such that $\mathrm{col}(F, U_2)$ is unimodular. Define $C = U_2 K$. Then

$$\left[\begin{array}{c} R(\frac{\rm d}{{\rm d}t}) \\ C(\frac{\rm d}{{\rm d}t}) \end{array} \right] w = 0$$

is a minimal kernel representation of $\mathcal{K}$, so $\mathcal{K}$ is regularly implemented by the controller $C(\frac{\rm d}{{\rm d}t})w = 0$.
(2. $\Rightarrow$ 3.) Assume that $R = FK$. Factorize $R = DR_1$ with $\det(D) \neq 0$ and $R_1(\lambda)$ full row rank for all λ. We want to prove that $\ker(R_1) + \ker(K) = \ker(R)$. We first claim that $(D, -F)$ is an MLA of $\mathrm{col}(R_1, K)$. Indeed, $DR_1 - FK = 0$ and $(D(\lambda), -F(\lambda))$ has full row rank for all λ. It remains to prove that $\mathrm{rowdim}(D, -F) = \mathrm{coldim}(D, -F) - \mathrm{rank}(\mathrm{col}(R_1, K))$. On the one hand, since D is square, we have $\mathrm{rowdim}(D, -F) = \mathrm{rowdim}(R_1)$. On the other hand,

$$\left[\begin{array}{cc} D & 0 \\ 0 & I \end{array} \right] \left[\begin{array}{c} R_1 \\ K \end{array} \right] = \left[\begin{array}{c} F \\ I \end{array} \right] K,$$

which implies that $\mathrm{rank}(\mathrm{col}(R_1, K)) = \mathrm{rank}(K) = \mathrm{rowdim}(K)$. Hence $\mathrm{rowdim}(R_1) = \mathrm{rowdim}(R_1, K) - \mathrm{rowdim}(K) = \mathrm{coldim}(D, -F) - \mathrm{rank}(\mathrm{col}(R_1, K))$. The proof is then completed by noting that, by proposition 10, $DR_1 = FK = R$ yields a kernel representation of $\ker(R_1) + \ker(K)$.
(3. $\Rightarrow$ 2.) Assume that $\ker(R_1) + \ker(K) = \ker(R)$. Let (W_1, W_2) be a full row rank MLA of $\mathrm{col}(R_1, K)$. By proposition 10 $W_1 R_1 = -W_2 K$ yields a kernel representation of $\ker(R)$. We claim that $W_2 K$ in fact yields a *minimal* kernel representation. Indeed, let p be a polynomial row vector such that $pW_2K = 0$. Then also $pW_1R_1 = 0$. Since both K and R_1 have full row rank, this implies $p(W_1, W_2) = 0$, so $p = 0$, which proves the claim. As a consequence, there exists a unimodular U such that $R = UW_2K$. Define $F := UW_2$. We will prove that $F(\lambda)$ has full row rank for all λ. Suppose, for some λ and some complex row vector η, $\eta U(\lambda)W_2(\lambda) = 0$. Then it follows that $\eta U(\lambda)W_1(\lambda)R_1(\lambda) = 0$, which implies $\eta U(\lambda)W_1(\lambda) = 0$. Since $(W_1(\lambda), W_2(\lambda))$ has full row rank, this implies $\eta U(\lambda) = 0$, so $\eta = 0$. □

Given a minimal kernel representation $R(\frac{\rm d}{{\rm d}t})w = 0$ of the plant behavior $\mathcal{P}$, and a minimal kernel representation $K(\frac{\rm d}{{\rm d}t})w = 0$ of the desired behavior $\mathcal{K}$, we would like to have an algorithm to decide whether $\mathcal{K}$ is regularly implementable by full interconnection. In the

following we propose two possible algorithmic implementations of theorem 11 to check regular implementability of a given behavior, and to compute a controller that regularly implements it. The first algorithm is based on the equivalence of 1. and 2. in the theorem. Note that if K has full row rank and the polynomial equation $R = FK$ has a solution F, then it is unique.

Algorithm 1 Input: full row rank polynomial matrices R and K.

1. Solve the equation $R = FK$. If no solution exists, $\mathcal{K}$ is not regularly implementable. If a solution F exists, it is unique.

2. (Column compression) Compute a unimodular U such that $FU = [F_1\ 0]$, with F_1 full column rank.

3. Check if F_1 is unimodular. If not, then $\mathcal{K}$ is not regularly implementable, so stop. If F_1 is unimodular go to step 4 to compute a controller.

4. Compute W such that $[F;W]$ is unimodular.

5. Put $C = WK$. Then the controller $\ker(C)$ regularly implements $\mathcal{K}$.

The latter algorithm is coded in MATLAB as it is shown below.

Listing 1

```
function [C,flag]=behimpl(R,K)
%[C,flag]=behimpl(R,K) tests whether ker(K) is implementable
% by regular full interconnection as a subbehavior of ker(R).
% If that is the case, a controller C is calculated % and the flag is set to 1.
% R and K are frr (full row rank matrices)

% Step 1
% xab is a standard MATLAB function which solves the polynomial
% matrix equation XA=B
F=xab(K,R);

% Step 2
[Fc,r,U,UI]=colred(F);
% (Fc stores [F1  |  0])
% The following command separates the compressed (non zero matrix)
% F1 of Fc=[F1  |  0]

F1=separate0(Fc);
```

```
% Step 3
if isunimod(F1)==1
disp('The behavior K is reg.  implementable by full interconnection')

% Step 4
W=embedding(F);

% Step 5
C=W*K;
else
error('The behavior ker(K) is not reg.  implementable by full interconnection')

%%%%% Function to separate the non zero matrix Fc of F1 %%%%%

function [F1,Zero]=separate0(Fc)

% [F1,Zero]=separate0(Fc)
% In Fc=[F1; 0] or [F1 0] this command
% extracts F1 from Fc giving
% also the matrix 0 called here Zero.

% REMARK: Conjugate transpose is denoted by A'
% and non-conjugate transpose is denoted by A.'

[m,n]=size(Fc);

%%%%%%%%%%%%%
% m>n vertical, m<n horizontal
if m<n
vert=0;
else
Fc=Fc.';
vert=1;
end
%%%%%%%%%%%%%

[m,n]=size(Fc);

c=0;
for i=1:m,
if (Fc(i,:))~=0
c=c+1;
else
```

```
end
end

F1=Fc((1:c),:);

if vert==1
F1=F1';
else

end

if vert==0
Zero=zeros(m-c,n);
else
Zero=zeros(m-c,n).';
end
```

In the listing above we only included the essential commands. In the actual implementation we check whether K and R have full row rank. In fact the final algorithm will also be able to do implementation by partial interconnection as explained in chapter 4.

Example 12 *Let us consider the following full row rank polynomial matrices $R(\xi)$ and $K(\xi)$, given respectively by*

$$R = \begin{bmatrix} -15 - 0.87s - 0.13s^2 - 3.5s^3 & 9.9 + 6.4s + 29s^2 - 12s^3 & 1.6 + 7.9s - 5.8s^2 + 5.3s^3 \\ 17 + 8s + 8.8s^2 - 2.2s^3 & 3 - 5.2s - 1.6s^2 - 0.98s^3 & 10 + 4.3s - 0.26s^2 - 3.8s^3 \end{bmatrix}$$

$$K = \begin{bmatrix} -12 + 7.6s + 12s^2 - 10s^3 & 17 - 4.9s + 1.7s^2 + 3.5s^3 & -2.5 - 1.5s - 12s^2 - 0.22s^3 \\ 6.2 + 19s + 0.82s^2 + 16s^3 & -3.8 - 13s - 7.2s^2 - 5.6s^3 & 6.2 - 13s - 1.2s^2 - 11s^3 \\ -1.2 - 2.3s - 0.94s^2 + 0.12s^3 & -0.26 + 1.2s + 0.17s^2 - 0.037s^3 & 1 + 1.6s + 1.3s^2 + 0.52s^3 \end{bmatrix}$$

We do not show all the matrices computed by the program. After running our MATLAB file we obtain C:

Running 1
>> C=behimpl(R,K)
The behavior ker(K) is regularly implementable by full interconnection

Polynomial matrix in s: 1-by-3, degree: 12

$C =$

$$C = \begin{bmatrix} -16 - 4s - 4s^2 - 1.4s^3 & 20 - 8.7s + 26s^2 - 8.7s^3 & -6.8 + 8.6s - 6.8s^2 + 5.9s^3 \end{bmatrix}$$

Our second algorithm is based on the equivalence between 2. and 3. in theorem 11 We first explain the underlying theory. Given a minimal kernel representation $R(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ of the plant behavior $\mathcal{P}$, and a minimal kernel representation $K(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ of the desired behavior $\mathcal{K}$, we compute a factorization $R = DR_1$, where D is a square, nonsingular polynomial matrix and where $R_1(\lambda)$ has full row rank for all $\lambda \in \mathbb{C}$. Then $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ is a minimal kernel representation of $\mathcal{P}_{\mathrm{cont}}$. Clearly, condition 3. of theorem 11 is then equivalent with

$$\ker(K(\tfrac{\mathrm{d}}{\mathrm{d}t})) + \ker(R_1(\tfrac{\mathrm{d}}{\mathrm{d}t})) = \ker(R(\tfrac{\mathrm{d}}{\mathrm{d}t})). \tag{3.1}$$

In order to check this condition, we compute a kernel representation of the left-hand side (3.1) in this equation. For this we compute a full row rank MLA $[X_1 \;\; X_2]$ of the stacked polynomial matrix $[K; R_1]$. Then $X_1K = -X_2R_1$ and the left hand side of (3.1) has kernel representation $\ker(X_2(\frac{\mathrm{d}}{\mathrm{d}t})R_1(\frac{\mathrm{d}}{\mathrm{d}t}))w = 0$. As a consequence, $\mathcal{K}$ is regularly implementable if and only if

$$\ker(D(\tfrac{\mathrm{d}}{\mathrm{d}t})R_1(\tfrac{\mathrm{d}}{\mathrm{d}t})) = \ker(X_2(\tfrac{\mathrm{d}}{\mathrm{d}t})R_1(\tfrac{\mathrm{d}}{\mathrm{d}t})). \tag{3.2}$$

Now, we have the following lemma:

Lemma 2 *Condition (3.2) holds if and only if X_2D^{-1} is a unimodular polynomial matrix.*

Proof : ($\Rightarrow$) DR_1 has full row rank. Also, X_2R_1 has full row rank. Indeed, let p be a polynomial row vector such that $pX_2R_1 = 0$. Then also $pX_1K = 0$. Since both K and R_1 have full row rank, this implies $p(X_1, X_2) = 0$, so $p = 0$, which proves the claim. If (3.2) holds, then there exists a unimodular polynomial matrix U such that $UDR_1 = X_2R_1$. Since $R_1(\lambda)$ has full row rank for all λ this now implies that $UD = X_2$, so X_2D^{-1} is a unimodular polynomial matrix.
($\Leftarrow$) Assume that $X_2 = UD$ from some unimodular U. Then $X_2R_1 = UDR_1$, so (3.2) holds. □

It can easily be shown that if $U := X_2D^{-1}$ is unimodular, then $R = FK$, with F defined by $F := -U^{-1}X_1$, and $F(\lambda)$ has full row rank for all $\lambda \in \mathbb{C}$. This leads to the following algorithm to check regular implementability and to compute a required controller:

Algorithm 2 `Input: full row rank polynomial matrices` R `and` K.

1. `Compute a factorization` $R = DR_1$, `with` D `square and nonsingular, and` $R_1(\lambda)$ `full row rank for all` λ.

2. `Compute a full row rank MLA` $[X_1 \;\; X_2]$ `of` $[K; R_1]$.

3. `Compute the solution` U `of` $X_2 = UD$. `Check whether` U `is unimodular. If it is not then` $\mathcal{K}$ `is not regularly implementable, otherwise it is. In that case go to step 4 to compute a controller.`

4. `Compute the solution` F `of` $UF = X_1$.

5. `Compute` W `such that` $[F; W]$ `is unimodular`

6. Compute $C = WK$. The controller $\ker(C)$ regularly implements $\mathcal{K}$.

In order to simplify the coding of the latter algorithm, we first provide the listing corresponding only to step 1. In this step, the decomposition $R = DR_1$ means that we separate the controllable behavior of a given one. That program is called `behctrb.m` (from "controllable behavior").

Listing 2

```
function [D,R1]=behctrb(R)

% [D,R1]=behctrb(R)
% Computation of the controllable part of a given behavior.
% This command decomposes R=D*R1, R with full row rank (frr),
% R1 frr and D nonsingular.

% RM=0, i.e., M=MRA(R) then
M=null(R);
R1=(null(M'))';
% Since R=D*R1 and we have R and R1 we can determine D as
D=xab(R1,R);
[mR,nR]=size(R);
if D*R1-R =zeros(mR,nR)
error('There is something wrong with R=F*K')
else
end
```

Next, we continue with the remaining steps of the first algorithm which is based in the equivalence of statements 2 and 3 of theorem 11.

Listing 3

```
function C=behimplmla(R,K)

% C=behimplmla_2(R,K)
% Given R,K (min.ker.rep.  of the plant beh.  and cont.  beh.  resp)
% behimplmla tests whether ker(K) is implementable by
% regular full interconnection (irfi).
% If that is the case, a controller C is computed
% (based in the equivalence of statements 2 and 3
% of theorem 11).

% Step 1
[D,R1]=behctrb(R);
```

```
[mR,nR]=size(R);
[mR1,nR1]=size(R1);
[mD,nD]=size(D);

[mK,nK]=size(K);

% Step 2
S=[K;R1];
[mS,nS]=size(S);
X=(null(S'))';
[mX,nX]=size(X);

% X1 and X2 are obtained from X=[X1 | X2]

nX1=mK;
nX2=nX-nX1;
mX1=mX;
mX2=mX;

X1=X(:,1:nX1);
X2=X(:,nX1+1:nX1+nX2);

% Step 3
U=xab(D,X2);
if isunimod(U)==1
disp('ker(K) is regularly implementable by reg.full interconnection')

% Step 4
F=axb(U,X1);

% Step 5
W=embedding(F)

% Step 6
C=W*K

else
error('ker(K) is not regularly implementable by reg.full interconnection')
end
```

An example illustrates the ideas explained up to now.

Example 13 *Let us consider the following full row rank matrices $R(\xi)$ and $K(\xi)$ defined as it is shown below.*

$$R = \begin{bmatrix} 2.5 - 8.6s + 11s^2 - 3s^3 + 12s^4 & 10 + 21s - 6.6s^2 - 11s^3 + 9.5s^4 & -4 - 3.8s + 4.8s^2 + 4.4s^3 + 3.8s^4 \\ 11 + 8.6s - 12s^2 + 4s^3 - 5.8s^4 & -9.8 + 1.2s + 0.68s^2 - 5.3s^3 + 5.4s^4 & 6.8 + 5.4s - 5.1s^2 - 13s^3 - 6.1s^4 \end{bmatrix}$$

$$K = \begin{bmatrix} 0.86 + 6.9s + 21s^2 - 0.029s^3 - 0.9s^4 & -2.5 - 8.7s + 4.2s^2 - 1.3s^3 + 5.4s^4 & 8.8 - 13s + 8.7s^2 - 8s^3 - 7.5s^4 \\ -7.5 - 3.1s - 15s^2 + 8.3s^3 - 6.1s^4 & 9.6 + 20s + 2.9s^2 - 3.9s^3 + 5.8s^4 & -11 + 0.61s + 0.12s^2 - 1.7s^3 - 6.9s^4 \\ 1 + 0.68s + 0.37s^2 - 0.32s^3 & 0.21 + 0.52s - 0.18s^2 + 0.61s^3 & -0.67 + 0.28s - 0.73s^2 - 0.8s^3 \end{bmatrix}$$

The program computes C showing that

Running 2
>> C=behimplmla(R,K)

ker(K) is regularly implementable by reg.full interconnection

Polynomial matrix in s: 1-by-3, degree: 3

$$C = \begin{bmatrix} -16 - 4s - 4s^2 - 1.4s^3 & 20 - 8.7s + 26s^2 - 8.7s^3 & -6.8 + 8.6s - 6.8s^2 + 5.9s^3 \end{bmatrix}$$

3.2 Stabilization and pole placement by regular full interconnection

This section deals with the synthesis problems of stabilization and pole placement by regular full interconnection. We will give algorithms to compute, for a given plant behavior, controllers that achieve pole placement and stabilization. Both problems require the computation of a unimodular embedding.
We will first discuss the problem of pole placement by regular full interconnection. This problem is defined as follows.
Let $\mathcal{P} \in \mathcal{L}^q$ be a given plant behavior. Let $r(\xi)$ be a monic real polynomial. Find a controller behavior $\mathcal{C}$ such that the controlled behavior $\mathcal{K} = \mathcal{P} \cap \mathcal{C}$ is autonomous, has characteristic polynomial $\chi_{\mathcal{K}} = r$, and the interconnection is regular.
It was proven in [9] that for every monic real polynomial $r(\xi)$ there exists such controller $\mathcal{C}$ if and only if the plant behavior $\mathcal{P}$ is controllable and $\mathtt{p}(\mathcal{P}) < q$.
Assume now that $\mathcal{P}$ is represented by the minimal kernel representation $R(\frac{\mathrm{d}}{\mathrm{dt}})w = 0$, with $R(\xi)$ a real polynomial matrix. We assume that $\mathcal{P}$ is controllable, equivalently $R \in \mathcal{U}$. Obviously, $\mathtt{p}(\mathcal{P}) < q$ if and only if the number of rows of R is less than q. Now let $r(\xi)$ be an arbitrary real monic polynomial. In the following we describe an algorithm to compute a required controller:

Algorithm 3 `Input: The polynomial matrix` $R(\xi)$ `and the real monic polynomial` $r(\xi)$

1. `Compute a real polynomial matrix` $C_1(\xi)$ `such that` $[R; C_1]$ `is unimodular.`

2. `Output:` $C(\xi)$ `is the polynomial matrix obtained by multiplying one of the rows of` $C_1(\xi)$ `by` $r(\xi)$`.`

Define the controller behavior $\mathcal{C} \in \mathcal{L}^{\mathbb{q}}$ as the system represented by $C(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$. The controlled system behavior $\mathcal{K} = \mathcal{P} \cap \mathcal{C}$ is then represented by

$$\begin{bmatrix} R(\frac{\mathrm{d}}{\mathrm{d}t}) \\ C(\frac{\mathrm{d}}{\mathrm{d}t}) \end{bmatrix} w = 0,$$

Clearly, $\det(R; C) = r$, so the controlled system $\mathcal{K}$ is autonomous and $\chi_{\mathcal{K}} = r$.
Below we provide the MATLAB code of the m-file that computes a required controller.

Listing 4

```
function C=behplace(R,r)
%C=behplace(R,r) places poles by means of a suitable controller.
% Data, R(ξ) ∈ ℝ^{m×n}, r(ξ)
C1 =embedding(R)
% Compute the controller C (for any i ≤ m)
C = r * C1(i,:)
% Testing the resulting placement...
if (det([R;C])-r)≅0
error('Wrong placement')
else
disp('Pole allocated at '),roots(det([R;C]))
end
```

Next, we provide an example.

Example 14 *Let the plant behavior be represented by* $R(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$*, where* $R(\xi)$ *is the polynomial matrix given by*

$$R(\xi) = \begin{bmatrix} 11+1 & 9.5\xi+2 & 3\xi+3 \\ 1.4\xi+2.5 & 3\xi+1.7 & 2.7\xi+7.6 \end{bmatrix}, \quad r(\xi) = \xi + 1$$

Let the desired polynomial be given by $r(\xi) = \xi + 1$*. We want to compute the controller representation* $C(\xi)$ *to allocate the desired pole at* $\xi = -1$*.*

The corresponding output is shown below:

Running 3
>> C=behplace(R,r)

R =

1 + 11s 2 + 9.5s 3 + 3s
2.5 + 1.4s 1.7 + 3s 7.6 + 2.7s

C1 =

-0.7549 -0.6310 -0.1787

r =

1 + s

C =

-0.75 - 0.75s -0.63 - 0.63s -0.18 - 0.18s

Pole allocated at

ans =

-1.0000

>>

We will now turn to the stabilization problem. The problem of stabilization by regular full interconnection is formulated as follows. Let $\mathcal{P} \in \mathcal{L}^q$ be a given plant behavior. Find a controller behavior $\mathcal{C}$ such that the controlled behavior $\mathcal{K} = \mathcal{P} \cap \mathcal{C}$ is autonomous, its characteristic polynomial $\chi_{\mathcal{K}}$ is Hurwitz, and the interconnection is regular. It was proven in [9] that there exists such controller $\mathcal{C}$ if and only if the plant behavior $\mathcal{P}$ is stabilizable. Assume that $\mathcal{P}$ is represented by the minimal kernel representation $R(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$, with $R(\xi)$ a real polynomial matrix. Assume that $\mathcal{P}$ is stabilizable. This is equivalent to the condition that $R(\lambda)$ has full row rank for all $\lambda \in \mathbb{C}^+$, equivalently, $R \in \mathcal{M}$. In the following we describe an algorithm to compute a required controller:

Algorithm 4 `Input:` `The polynomial matrix` $R(\xi)$.

1. `Factorize` $R(\xi) = G(\xi)R_1(\xi)$, `with` G `Hurwitz and` $R_1 \in \mathcal{M}$.

2. `Compute a real polynomial matrix` $C(\xi)$ `such that` $[R_1; C]$ `is unimodular.`

The controller behavior $\mathcal{C} \in \mathcal{L}^{\mathbb{q}}$ represented by $C(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ stabilizes the system: clearly, $\det[R; C] = \det(G)\det[R_1; C]$, so the controlled system $\mathcal{K}$ is autonomous. The characteristic polynomial $\chi_{\mathcal{K}}$ is then a scalar multiple of $\det(G)$ so $\mathcal{K}$ is stable.
Below we provide the code of the relevant m-file:

Listing 5

```
function C=behstab(R)
% C=behstab(R) finds a stabilizing controller.
% Data:   R(ξ)
% The following command factorizes R as R = GR_1
[G, R_1]=behctrb(R)
C=embedding(R_1)
```

The corresponding running is given below.

Example 2: Stabilization by regular full interconnection. Let us see how the latter works with the following system represented by the polynomial matrix $R(\xi) = [-12 - 2\xi \quad 6 + 7\xi + \xi^2]$.

Running 4

$[G, R1]$=*behctrb(R)*

R =

$-12 - 2s \quad 6 + 7s + s^2$

G =

12 + 2s

R1 =

-1 0.5 + 0.5s

C =

0 -1

>>

The latter implies that there is an uncontrollable (but stabilizable) mode at $\xi = -6$. The controller is represented by the constant polynomial matrix $C = [0 \quad -1]$.

3.3 All regularly implementing controllers

In this section we will establish, for a given plant $\mathcal{P} \in \mathcal{L}^q$ and a given regularly implementable behavior $\mathcal{K} \in \mathcal{L}^q$, a parametrization of all controllers $\mathcal{C} \in \mathcal{L}^q$ that regularly implement $\mathcal{K}$ by full interconnection. This problem was considered before in [40] for the case that the plant behavior $\mathcal{P}$ is controllable, and the given subbehavior $\mathcal{K}$ is autonomous. The approach in [40] is heavily based on the use of image representations for $\mathcal{P}$. Here, we will establish a parametrization for arbitrary $\mathcal{P}$ and arbitrary (regularly implementable) $\mathcal{K}$.

In the following, let $\mathcal{K} \in \mathcal{L}^q$. Let K be a real polynomial matrix such that $K(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ is a (not necessarily minimal) kernel representation of $\mathcal{K}$. Our first goal is to find a condition on the polynomial matrix K that is necessary and sufficient for $\mathcal{K}$ to be regularly implementable. In order to formulate such condition, let $R(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ be a minimal kernel representation of $\mathcal{P}$, and let M be MRA of R such that $M(\lambda)$ has full column rank for all $\lambda \in \mathbb{C}$. Next, consider the polynomial matrix KM, and let Q be a full row rank MLA of KM. Finally, let W be a polynomial matrix such that $\mathrm{col}(Q, W)$ is unimodular. Note that the number of rows of W is equal to $\mathrm{rank}(KM)$. The following lemma gives necessary and sufficient conditions, in terms of the representing polynomial matrix K, for regular implementability of $\mathcal{K}$:

Lemma 3 *$\mathcal{K}$ is regularly implementable by full interconnection w.r.t. $\mathcal{P}$ if and only if*

$$\begin{bmatrix} R(\frac{\mathrm{d}}{\mathrm{d}t}) \\ W(\frac{\mathrm{d}}{\mathrm{d}t})K(\frac{\mathrm{d}}{\mathrm{d}t}) \end{bmatrix} w = 0 \tag{3.3}$$

is a minimal kernel representation of $\mathcal{K}$. A controller that regularly implements $\mathcal{K}$ is then represented by the minimal kernel representation $W(\frac{\mathrm{d}}{\mathrm{d}t})K(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$.

Proof : Factor $R = DR_1$, with D square and nonsingular and $R_1(\lambda)$ full row rank for all λ. Then $\mathcal{P}_{\mathrm{cont}} = \ker(R_1)$. Let S be such that $\mathrm{col}(R_1, S)$ is unimodular. Define

$$\begin{bmatrix} R_1 \\ S \end{bmatrix}^{-1} = \begin{bmatrix} N_1 & M_1 \end{bmatrix}.$$

Then we have $\mathrm{im}(M_1) = \ker(R_1) = \mathrm{im}(M)$, so there exists a unimodular Z such that $M_1 = MZ$. It follows that $N_1R_1 + MZS = I_q$. We claim that there exists a polynomial matrix T such that $TR = QK$. In order to prove this, we will show that $\ker(R) \subseteq \ker(QK)$. Indeed, let w be such that $R(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$. Since $\mathcal{P} = \mathcal{K} + \mathcal{P}_{\mathrm{cont}}$, there exist $w_1 \in \mathcal{K}$ and $w_2 \in \mathcal{P}_{\mathrm{cont}}$ such that $w = w_1 + w_2$. Hence (omitting the symbol $\frac{\mathrm{d}}{\mathrm{d}t}$ in the equations below),

$$QKw = QK(w_1 + w_2) = QKw_2 = QK(N_1R_1 + MZS)w_2 = QKN_1R_1w_2 = 0.$$

Next, note that

$$\begin{bmatrix} I_p & 0 \\ -T & Q \\ 0 & W \end{bmatrix} \begin{bmatrix} R \\ K \end{bmatrix} = \begin{bmatrix} R \\ 0 \\ WK \end{bmatrix}.$$

Note that the left-most matrix in this equation is unimodular. Thus we have: $w \in \mathcal{K}$ if and only if $K(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ and $R(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ if and only if $W(\frac{\mathrm{d}}{\mathrm{d}t})K(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ and $R(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$. This proves that (3.3) is indeed a kernel representation of $\mathcal{K}$.
Finally, we show that the representation (3.3) is minimal. Indeed,

$$\begin{bmatrix} R \\ WK \end{bmatrix} \begin{bmatrix} N_1 & MZ \end{bmatrix} = \begin{bmatrix} D & 0 \\ WKN_1 & WKMZ \end{bmatrix}.$$

It is easily seen that, by construction, WKM has full row rank. Since D is nonsingular, we conclude that $\mathrm{col}(R, WK)$ must have full row rank as well. □

We will now apply lemma 3 to establish the main result of this section. It gives, for a given regularly implementable subbehavior $\mathcal{K}$ of $\mathcal{P}$, a parametrization of all controllers that regularly implement $\mathcal{K}$.

Theorem 15 : *Let $\mathcal{P} \in \mathcal{L}^q$, with minimal kernel representation $R(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$. Let $\mathcal{K} \in \mathcal{L}^q$ be regularly implementable by full interconnection and let K be a polynomial matrix such that $K(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ is a kernel representation of $\mathcal{K}$. Construct a polynomial matrix W as follows:*

1. *Choose a MRA M of R such that $M(\lambda)$ has full column rank for all λ,*
2. *Choose a full row rank MLA Q of KM,*
3. *Choose W such that $\mathrm{col}(Q, W)$ is unimodular.*

Then for any $\mathcal{C} \in \mathcal{L}^{\mathfrak{q}}$ represented by $C(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ the following statements are equivalent:

1. *$\mathcal{C}$ has minimal kernel representation $C(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ and regularly implements $\mathcal{K}$,*
2. *there exists a polynomial matrix F and a unimodular polynomial matrix U such that*

$$C = FR + UWK. \tag{3.4}$$

Proof : (2. ⇒ 1.) First note that since $\mathcal{K}$ is regularly implementable, by lemma 3 the polynomial matrix $\mathrm{col}(R, WK)$ has full row rank. Since

$$\begin{bmatrix} I_p & 0 \\ F & U \end{bmatrix} \begin{bmatrix} R \\ WK \end{bmatrix} = \begin{bmatrix} R \\ FR + UWK \end{bmatrix}, \tag{3.5}$$

this implies that also $C = FR + UWK$ has full row rank, so $C(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ is a minimal representation of $\mathcal{C}$. It also follows from (3.5) that $\mathcal{C}$ implements $\mathcal{K}$. Clearly, the interconnection of $\mathcal{P}$ and $\mathcal{C}$ is regular.
(1. ⇒ 2.) Assume that C has full row rank, and $\mathcal{C}$ regularly implements $\mathcal{K}$. Then both

$$\begin{bmatrix} R(\frac{\mathrm{d}}{\mathrm{d}t}) \\ C(\frac{\mathrm{d}}{\mathrm{d}t}) \end{bmatrix} w = 0 \text{ and } \begin{bmatrix} R(\frac{\mathrm{d}}{\mathrm{d}t}) \\ W(\frac{\mathrm{d}}{\mathrm{d}t})K(\frac{\mathrm{d}}{\mathrm{d}t}) \end{bmatrix} w = 0$$

are a minimal representation of $\mathcal{K}$. Consequently, there exists a unimodular polynomial matrix

$$V = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix}$$

such that $V\mathrm{col}(R, WK) = \mathrm{col}(R, C)$. As in the proof of lemma 3, let S be such that $\mathrm{col}(R_1, S)$ is unimodular, and let N_1 and Z be such that $\mathrm{col}(R_1, S)^{-1} = [N_1 \quad MZ]$, with Z unimodular. Then we have

$$\begin{bmatrix} R \\ WK \end{bmatrix} \begin{bmatrix} N_1 & MZ \end{bmatrix} = \begin{bmatrix} D & 0 \\ WKN_1 & WKMZ \end{bmatrix}.$$

and

$$\begin{bmatrix} R \\ C \end{bmatrix} \begin{bmatrix} N_1 & MZ \end{bmatrix} = \begin{bmatrix} D & 0 \\ CN_1 & CMZ \end{bmatrix}.$$

This implies $V_{12}WKMZ = 0$. Since WKM has full row rank, $V_{12} = 0$. Also, $V_{11}D = V_{11}$ so $V_{11} = I_p$. Thus

$$V = \begin{bmatrix} I_p & 0 \\ V_{21} & V_{22} \end{bmatrix}.$$

It follows that V_{22} is unimodular. We also have

$$\begin{bmatrix} I_p & 0 \\ V_{21} & V_{22} \end{bmatrix} \begin{bmatrix} R \\ WK \end{bmatrix} = \begin{bmatrix} R \\ C \end{bmatrix},$$

from which $C = V_{21}R + V_{22}WK$. This completes the proof of the theorem □

Summarizing the above, for a given plant $\mathcal{P}$ with kernel representation $R(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ and a given regularly implementable subbehavior $\mathcal{K} \subseteq \mathcal{P}$ with kernel representation $K(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ (not necessarily minimal!) a parametrization of all controllers that regularly implement $\mathcal{K}$ is obtained in the following steps:

1. find a MRA M of R such that $M(\lambda)$ has full column rank for all λ,
2. find a full row rank MLA Q of KM,
3. find W such that $\mathrm{col}(Q, W)$ is unimodular,
4. the controllers $\mathcal{C} \in \mathcal{L}^q$ that regularly implement $\mathcal{K}$ are parameterized by

$$\mathcal{C} = \{w \in \mathcal{L}_1^{\mathrm{loc}}(\mathbb{R}, \mathbb{R}^q) \mid (FR + UWK)(\tfrac{\mathrm{d}}{\mathrm{d}t})w = 0\}$$

with F ranging over all polynomial matrices with p columns and r rows, and U ranging over all unimodular $r \times r$ polynomial matrices. Here, $r := \mathrm{rank}(KM)$.

This leads to the following MATLAB code:

Listing 6

```
function W=behallimpl(R,K)
% W=behallimpl(R,K)
% All implementing controllers:  The full interconnection case.

% Checking whether ker(K) is regularly implementable by rfi:
C=behimpl(R,K)

% M is fcr:
M=null(R)

% Q=MLA(K*M)
T=K*M;
Q=(null(T'))'
W=embedding(Q)
disp('The set of all parameterizing controllers is (FR+UWK)(d/dt)w=0')
```

In order to illustrate the latter computation, let us consider the following example.

Example 16 Let us consider matrices R and K given as

$$R = [\xi + 1 \;\; \xi + 2], \quad K = \begin{bmatrix} \xi + 3 & \xi + 4 \\ \xi + 5 & \xi + 6 \end{bmatrix}$$

The corresponding W is

Running 5

D =

0.71

R1 =

1.4 + 1.4s 2.8 + 1.4s

W =

0 1

The set of all parameterizing controllers is (FR+UWK)(d/dt)w=0

3.4 All stabilizing controllers

In this section we will consider the problem of parametrizing *all* stabilizing controller behaviors for $\mathcal{P}$. Our result in this section generalizes the result from [40] that was obtained under the assumption that $\mathcal{P}$ can be represented by an image representation, equivalently, $\mathcal{P}$ is controllable.
Assume that $\mathcal{P}$ is represented by the minimal kernel representation $R(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$. Assume that $\mathcal{P}$ is stabilazable, equivalently $R(\lambda)$ has full row rank for all $\lambda \in \mathbb{C}^+ = \{\lambda \in \mathbb{C} \mid \mathrm{Re}(\lambda) \geq 0\}$. The following corollary of theorem 15 yields a parametrization of all stabilizing controllers for the stabilizable plant $\mathcal{P}$:

Corollary 1 *Let $\mathcal{P} \in \mathcal{L}^{\mathfrak{q}}$ be stabilizable. Let $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ be a minimal kernel representation of the controllable part $\mathcal{P}_{\mathrm{cont}}$. Let C_0 be such that $\mathrm{col}(R_1, C_0)$ is unimodular. Then for any $\mathcal{C} \in \mathcal{L}^{\mathfrak{q}}$ represented by the kernel representation $C(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ the following statements are equivalent:*

1. *$\mathcal{P} \cap \mathcal{C}$ is autonomous and stable, the interconnection is regular and the kernel representation $C(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ is minimal,*

2. *there exist a polynomial matrix F and a Hurwitz polynomial matrix D such that $C = FR_1 + DC_0$.*

Proof : Let $R(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ be a minimal kernel representation of $\mathcal{P}$. Then $R = D_1 R_1$, with D_1 Hurwitz. (2. $\Rightarrow$ 1.) Assume $C = FR_1 + DC_0$. We have

$$\begin{bmatrix} R \\ C \end{bmatrix} = \begin{bmatrix} D_1 & 0 \\ F & D \end{bmatrix} \begin{bmatrix} R_1 \\ C_0 \end{bmatrix}.$$

This implies that $\mathrm{col}(R, C)$ has full row rank, so the interconnection of $\mathcal{P}$ and $\mathcal{C}$ is regular. Also, for some nonzero constant c, $\det \mathrm{col}(R, C) = c\det(D_1)\det(D)$, so the interconnection is autonomous and stable.
(1. $\Rightarrow$ 2.) Denote $\mathcal{P} \cap \mathcal{C}$ by $\mathcal{K}$, and define $K := \mathrm{col}(R, C)$. Then $K(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ is a kernel representation of $\mathcal{K}$. Let M be a MRA of R such that $M(\lambda)$ has full column rank for all λ, and let Q be a full row rank MLA of KM. Let W be such that $\mathrm{col}(Q, W)$ is unimodular. Then according to theorem 3, $\mathcal{K}$ has a minimal kernel representation

$$\begin{bmatrix} R(\frac{\mathrm{d}}{\mathrm{d}t}) \\ W(\frac{\mathrm{d}}{\mathrm{d}t})K(\frac{\mathrm{d}}{\mathrm{d}t}) \end{bmatrix} w = 0.$$

Note that since $\mathcal{K}$ is autonomous, $\mathrm{col}(R, WK)$ is in fact square and nonsingular. Let N_1 and M_1 be such that

$$\begin{bmatrix} R_1 \\ C_0 \end{bmatrix}^{-1} \begin{bmatrix} N_1 & M_1 \end{bmatrix}.$$

Then $\mathrm{im}(M_1) = \ker(R_1) = \mathrm{im}(M)$, so there exists a unimodular matrix Z such that $M_1 = MZ$. This implies that $N_1R_1 + MZC_0 = I_q$. We then also have

$$\begin{bmatrix} R \\ WK \end{bmatrix} = \begin{bmatrix} N_1 & MZ \end{bmatrix} = \begin{bmatrix} D_1 & 0 \\ WKN_1 & WKMZ \end{bmatrix},$$

so $WKMZ$ is square and

$$\det \begin{bmatrix} R \\ WK \end{bmatrix} = \det(D_1)\det(WKMZ).$$

Since $\mathcal{K}$ is stable, $\mathrm{col}(R, WK)$ is Hurwitz, so $WKMZ$ is Hurwitz. Finally, since $\mathcal{C}$ regularly implements $\mathcal{K}$, by theorem 15 there exists a polynomial matrix F' and a unimodular U such that $C = F'R + UWK$. Hence

$$C = F'R + UWK(N_1R_1 + MZC_0) = (F'D_1 + UWKN_1)R_1 + UWKMZC_0.$$

The proof of the theorem is then completed by taking $F := F'D_1 + UWKN_1$ and $D := UWKMZ$. □

If, in the latter proof, we assume that $\mathcal{P}$ is controllable, then we can take $R = R_1$, and we recover the parametrization of all stabilizing controllers that was obtained in [40].
Below we establish an m-file that computes a parametrization of all stabilizing controllers.

Listing 7

```
function [R1, D, C0]=behallstab(R)
% [R1, D, C0]=behallstab(R)
% All stabilizing controllers:  The full interconnection case

[D,R1]=behcntr(R)
% Is D Hurwitz?
ra=roots(det(D))

c=0;
for i=1:length(ra)
if real(ra(i))<0
c=c+1;
else
end
end

if c==length(ra)
disp('OK: D is Hurwitz')
```

```
else
error('Matrix D is not Hurwitz')
end
C0=embedding(R1)
disp('Then a parameterization of all the stabilizing controllers for the')
disp('stabilizable plant behavior ker(P) is C=F*R1+D*C0')
disp('where the polynomial matrix F is the free parameter')
```

Example 17 *The plant is represented by* $R(\xi) = [1 - \xi - \xi^2 + \xi^3 \quad 2 - \xi - \xi^2]$.

The family of all stabilizing controllers for this plant is computed as follows.

Running 6

>> R

R =

$1 - s - s^2 + s^3 \quad 2 - s - s^2$

>> [R1, D, C0]=behallstab(R)

*ker(K) is implementable w.r.t ker(P) according to R=F*K*

D =

-2.6 + 2.6s

R1 =

$-0.38 + 0.38s^2 \quad -0.76 - 0.38s$

ra =

1.0000

??? Error using ==> behallstab
Matrix D is not Hurwitz

>>

As we see, matrix $R(s)$ is not suitable for finding a family of stabilizing controllers. Now, let us examine an example that does work which is the following.

Example 18 *The system is again* $R = [-12 - 2s \quad 6 + 7s + s^2]$.

After running the corresponding program we obtain the following computations:

Running 7
>>R

R =

$-12 - 2s \quad 6 + 7s + s^2$

>> $[R1, D, C0]$=*behallstab(R)*

*ker(K) is implementable w.r.t ker(P) according to R=F*K*

D =

12 + 2s

R1 =

$-1 \quad 0.5 + 0.5s$

ra =

-6.0000

OK: D is Hurwitz

C0 =

>> *C0*=$[0 \quad -1]$

C0 =

$0 \quad -1$

*D*C0=*

$0 \quad -12-2s$

*Then a parametrization of all the stabilizing controllers for the stabilizable plant behavior ker(P) is C=F*R1+D*C0 where the polynomial matrix F is the free parameter*

*C=F**$[-1 \quad 0.5+0.5s]+[0 \quad -12-2s]$

>>

To conclude this section, as another corollary of theorem 15 we will, for a given monic real polynomial $r(\xi)$, give a parametrization of *all* controllers that achieve this desired characteristic polynomial.
If $\mathcal{P}$ is represented by the minimal kernel representation $R(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ then $\mathcal{P}$ is controllable if and only if $R(\lambda)$ has full row rank for all $\lambda \in \mathbb{C}$. The condition $p(\mathcal{P}) < q$ is then equivalent to the condition that the number of rows of R is less than q. The proof of the following corollary is completely analogous to that of the previous one:

Corollary 2 *Let $\mathcal{P} \in \mathcal{L}^{\mathbb{q}}$ be controllable and let $R(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ be a minimal kernel representation of $\mathcal{P}$. Let $r(\xi)$ be a monic real polynomial. Let C_0 be such that $\mathrm{col}(R, C_0)$ is unimodular. Then for any $\mathcal{C} \in \mathcal{L}^{\mathbb{q}}$ represented by the kernel representation $C(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ the following statements are equivalent:*

1. *$\chi_{\mathcal{P}\cap\mathcal{C}} = r$, the interconnection of $\mathcal{P} \cap \mathcal{C}$ is regular and the kernel representation $C(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ is minimal,*
2. *there exist a polynomial matrix F and a square polynomial matrix D with the property that $\det(D) = cr$ for some constant $c \neq 0$, such that $C = FR + DC_0$.*

3.5 Summary

In this chapter important synthesis problems were defined and solved. Starting with the pole placement and stabilization problems, a parametrization of all regularly implementing and stabilizing controllers is offered for the full interconnection case. This result from a behavioral point of view, resembles the Youla parametrization of all stabilizing controllers from the classical control theory. We have seen that since the Behavioral Approach considers as particular case of modelling and control a matrix polynomial point of view, analysis and synthesis of control systems are based on three main operations: finding minimal

left/right annihilators, row/column compressions and embeddings. Nevertheless, we have not talked about numerical consequences. This part will be considered later.

Chapter 4

Partial Interconnection Issues

In this chapter we shall continue the development we started in the previous chapter. Instead of full interconnection, we will now deal with the case of partial interconnection. We will start with reviewing the notions of implementability and regular implementability. Next we will deal with the problems of pole placement and stabilization by partial regular interconnection. We will set up algorithms to compute regular controllers for a given plant that achieve pole placement or stabilization. Next, we show how to implement these algorithms in MATLAB. An important part of this chapter deals with the problem of parametrizing (for the partial interconnection case) all controllers that regularly implement a given desired behavior. We will establish such parametrization, first under certain observability assumptions, and afterwards also in the general case. Also for this parametrization problem, we will give numerical algorithms. Finally, we will address the problem of parametrizing for a given plant, all controllers that stabilize the given plant by regular partial interconnection. Also for this problem we give the corresponding MATLAB codes and some examples runned with comments. We conclude the chapter with a summary.

4.1 Regular implementability by partial interconnection

In subsection 2.10.1 we discussed the notion of implementability by partial interconnection, which deals with the question which system behaviors are achievable by interconnecting a given plant with a controller behavior. Then, in section 3.1 we treated implementability and regular implementability for the special case of *full interconnection*. In the present section we will study the notion of regular implementability by partial interconnection and generalize the results of section 3.1 to the partial interconnection case. For a given full plant behavior, we will give characterizations of all regularly implementable behaviors. Let $\mathcal{K} \in \mathcal{L}^{\mathtt{w}}$ be a given behavior, which should be interpreted as a 'desired' behavior. If $\mathcal{K}$ can be achieved by *regular* partial interconnection, i.e. if there exists $\mathcal{C}$ such that $\mathcal{K} = (\mathcal{K}_{\text{full}}(\mathcal{C}))_w$ and

$$\mathtt{p}(\mathcal{K}_{\text{full}}(\mathcal{C})) = \mathtt{p}(\mathcal{P}_{\text{full}}) + \mathtt{p}(\mathcal{C}),$$

R.Z. Yoe: *Modelling and Control of Dynamical Systems: Numerical Implementation in a Behavioral Framework*, Studies in Computational Intelligence (SCI) **124**, 59–93 (2008)
www.springerlink.com

then we call $\mathcal{K}$ *regularly implementable by partial interconnection*. Conditions for regular implementability were given in [9]:

Proposition 19 : *Let $\mathcal{P}_{\text{full}} \in \mathcal{L}^{\mathtt{w}+k}$. $\mathcal{K} \in \mathcal{L}^{\mathtt{w}}$ is regularly implementable by partial interconnection through c if and only if $\mathcal{N} \subseteq \mathcal{K} \subseteq (\mathcal{P}_{\text{full}})_w$ and $\mathcal{K}$ is regularly implementable w.r.t. $(\mathcal{P}_{\text{full}})_w$ by full interconnection.*

4.2 Pole placement and stabilization by regular partial interconnection

In subsection 3.2 we have discussed pole placement and stabilization by regular full interconnection. In the present section we will discuss these synthesis problems in the more general context of partial interconnection. Again, we will establish algorithms to compute, for a given full plant behavior, controllers that achieve the design specifications. This section is subdivided into two subsections. The first subsection deals with the problem of pole placement, in the second subsection we treat the stabilization problem.

4.2.1 Pole placement by regular partial interconnection

The problem of pole placement by regular partial interconnection through c is formulated as follows: given a real monic polynomial $r(\xi)$, find a controller $\mathcal{C} \in \mathcal{L}^k$ such that the manifest controlled behavior $\mathcal{K}$ is autonomous, has characteristic polynomial $\chi_{\mathcal{K}} = r$, and the interconnection is regular. Necessary and sufficient conditions for the existence of such controller for any given $r(\xi)$ can be expressed in terms of the manifest plant behavior and *hidden behavior* associated with the full plant $\mathcal{P}_{\text{full}}$.

It was shown in [9] that for every real monic polynomial r there exists a required controller $\mathcal{C}$ if and only if $(\mathcal{P}_{\text{full}})_w$ is controllable, $\mathtt{p}((\mathcal{P}_{\text{full}})_w) < \mathtt{w}$ and $\mathcal{N} = 0$.

In the following we will establish an algorithm to compute, for a given r, a required polynomial matrix C (representing a controller $\mathcal{C}$) in case that the full plant behavior $\mathcal{P}_{\text{full}}$ is given by a minimal kernel representation. In the algorithm we have to solve a unimodular embedding problem twice.

Let $\mathcal{P}_{\text{full}}$ be represented by $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w + R_2(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$, with $[R_1\ R_2]$ full row rank. Let U be a unimodular matrix such that

$$UR_2 = \begin{bmatrix} R_{12} \\ 0 \end{bmatrix}, \tag{4.1}$$

such that R_{12} has full row rank. Let R_{11} and R_{21} be obtained by partitioning the product UR_1 as in (4.1):

$$UR_1 = \begin{bmatrix} R_{11} \\ R_{21} \end{bmatrix},$$

Evidently, $\mathcal{P}_{\text{full}}$ is then also represented by the minimal kernel representation

$$\begin{bmatrix} R_{11}(\frac{\mathrm{d}}{\mathrm{d}t}) & R_{12}(\frac{\mathrm{d}}{\mathrm{d}t}) \\ R_{21}(\frac{\mathrm{d}}{\mathrm{d}t}) & 0 \end{bmatrix} \begin{bmatrix} w \\ c \end{bmatrix} = 0$$

From this representation it is clear that the manifest plant behavior $(\mathcal{P}_{\text{full}})_w$ is represented minimally by $R_{21}(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ (since R_{12} has full row rank), and that the hidden behavior $\mathcal{N}$ is represented by

$$\begin{bmatrix} R_{11}(\frac{\mathrm{d}}{\mathrm{d}t}) \\ R_{21}(\frac{\mathrm{d}}{\mathrm{d}t}) \end{bmatrix} w = 0. \tag{4.2}$$

Assume that $(\mathcal{P}_{\text{full}})_w$ is controllable. Then $R_{21}(\lambda)$ has full row rank for all λ, so R_{21} can be embedded into a unimodular polynomial matrix. Also, assume that $\mathcal{N} = 0$. Then $[R_{11}(\lambda); R_{21}(\lambda)]$ has full column rank for all λ, so $[R_{11}; R_{21}]$ can be embedded into a unimodular matrix as well. Now choose polynomial matrices U_{12} and U_{22} such that

$$\begin{bmatrix} R_{11} & U_{12} \\ R_{21} & U_{22} \end{bmatrix} \tag{4.3}$$

is unimodular. Next, solve the polynomial equation (unknowns X and Y)

$$\begin{bmatrix} R_{11} & U_{12} \\ R_{21} & U_{22} \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} R_{12} \\ 0 \end{bmatrix}. \tag{4.4}$$

Then we have

$$\begin{bmatrix} R_{11} & U_{12} \\ R_{21} & U_{22} \end{bmatrix} \begin{bmatrix} I & X \\ 0 & Y \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & 0 \end{bmatrix},$$

so $\mathcal{P}_{\text{full}}$ also has the minimal kernel representation

$$\begin{bmatrix} I & X(\frac{\mathrm{d}}{\mathrm{d}t}) \\ 0 & Y(\frac{\mathrm{d}}{\mathrm{d}t}) \end{bmatrix} \begin{bmatrix} w \\ c \end{bmatrix} = 0.$$

From this, note that $(w, c) \in \mathcal{P}_{\text{full}}$ implies $w = -X(\frac{\mathrm{d}}{\mathrm{d}t})c$. Next, let C_0 be such that $[R_{21}; C_0]$ is unimodular, and let C_1 be any polynomial matrix obtained by multiplying one of the rows of C_0 by the desired polynomial r. Then of course $\det[R_{21}; C_1] = r$. Define now the controller behavior $\mathcal{C} \in \mathcal{L}^k$ as the behavior represented by $C(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$, with C defined by $C := C_1 X$. We claim that the corresponding manifest controlled behavior $\mathcal{K}$ is then represented by

$$\begin{bmatrix} R_{21}(\frac{\mathrm{d}}{\mathrm{d}t}) \\ C_1(\frac{\mathrm{d}}{\mathrm{d}t}) \end{bmatrix} w = 0,$$

and that the interconnection of $\mathcal{P}_{\text{full}}$ with $\mathcal{C}$ is regular.
Indeed, let $w \in \mathcal{K}$. Then there exists c such that $(w, c) \in \mathcal{P}_{\text{full}}$ and $c \in \mathcal{C}$. Hence, $C(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$, so $C_1(\frac{\mathrm{d}}{\mathrm{d}t})X(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$. Also, $(w, c) \in \mathcal{P}_{\text{full}}$, so $w = -X(\frac{\mathrm{d}}{\mathrm{d}t})c$. This yields $C_1(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$. Also, $R_{21}(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$. Conversely, assume that $R_{21}(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ and $C_1(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$. There exists c such that $(w, c) \in \mathcal{P}_{\text{full}}$. Hence $w = -X(\frac{\mathrm{d}}{\mathrm{d}t})c$ so $C_1(\frac{\mathrm{d}}{\mathrm{d}t})X(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$, equivalently, $c \in \mathcal{C}$. Thus we obtain $(w, c) \in \mathcal{K}_{\text{full}}$, so $w \in \mathcal{K}$. We conclude that $\chi_{\mathcal{K}} = r$. It can also be shown that the interconnection is regular. This leads to the following algorithm that takes the polynomial matrices representing the full plant behavior and the desired polynomial as input, and produces as output a polynomial matrix representing a desired controller:

Algorithm 5 Input: The polynomial matrices $R_1(\xi)$ and $R_2(\xi)$ and the real polynomial $r(\xi)$

1. Compute a unimodular matrix U and a full row rank polynomial matrix R_{12} such that $UR_2 = [R_{12}; 0]$
2. Compute polynomial matrices R_{11} and R_{21} such that $UR_1 = [R_{11}; R_{21}]$
3. Compute polynomial matrices U_{12} and U_{22} such that (4.3) is unimodular
4. Compute polynomial matrices X and Y that solve the polynomial equation (4.4).
5. Compute a polynomial matrix C_0 such that $[R_{21}; C_0]$ is unimodular. Compute C_1 as the polynomial matrix obtained by multiplying one of the rows of C_0 by $r(\xi)$.
6. Output: $C = C_1X$.

Below the MATLAB code of an m-file that computes a required controller is given. The program which computes the controller is a direct implementation of the steps indicated in the algorithm. We note that we take advantage of the MATLAB command *xab*, i.e., $X = xab(A, B)$ that computes a solution for the matrix equation $XA = B$ (in addition the command *axb* solves $AX = B$ for X).

Listing 8

```
function C=behplace(R1,R2,r)
% C=behplace_pi(R1,R2,r)
% Algorithm to compute a pole placement
% controller by partial interconnection.

% Step 1:  Compute a unimodular matrix U and a frr R12
% s.t.  U*R2=[R12;0]

[m,n]=size([R1 R2]);

% We start considering a polynomial matrix
% given as:  R=[R1 R2],i.e., R is a kernel
% representation given by R1(d/dt)*w+R2(d/dt)*c=0.
% R1 has frr but R2 does not.

% Rowcompression of R2:
```

```
[R2rowred,rR2rowred,U,UI]=rowred(R2);
% R2rowred=[R12;0],i.e., R2 does not have frr

% Step 2:  Compute R11, R21 s.t U*R1=[R11;R21]

% The same transformation defined by U above
% is applied to R1,i.e., we obtain U*R1

UR1=U*R1;
[mUR1,nUR1]=size(UR1);

%R1=[R11;R21];

% We retrieve R11,R12,R21,R22=Zero:
[R12,Zero]=separate0(R2rowred);

[mR12,nR12]=size(R12);
[mZero,nZero]=size(Zero);

R11=UR1(1:mR12,:);
R21=UR1(mR12+1:m,:);

if isempty(R21) =1
else
error('R21 is empty')
end

% Step 3:  Compute polynomial matrices U12,U22 s.t.  [R11 U12;R21; U22],i.e,
% that means Uemb=embedding([R11;R21])=embedding(U*R1) where
% Uemb will be partitioned as =[Uemb12;Uemb22].
% The embedded matrix looks like [U*R1;Uemb].

% REMARK: In the papers and in the book the transpose of the
% matrices shown below is used.  That is why we have to transpose.

% Transposition without conjugating is done as M.'

% Wemb is the embedded matrix by Q.
% Uemb is the unimodular rowreducing matrix.

[Wemb,Q,Uemb,detUemb,deteWemb]=embedding(UR1);
U12_22=Q.';
```

```
% Step 4:  Solve [U*R1;U12_22]*Z=[R12;0] where [R12;0]=U*R2=R2rowred
% i.e., we solve Wemb*Z=R2rowred
% Z is obtained here.
Z=xab(Wemb.',R2rowred);

[mR2rowred,nR2rowred]=size(R2rowred);

if Wemb*Z.'-R2rowred.' =zeros(nR2rowred,mR2rowred)
error('xab equation not satisfied')
else
end

[mZ,nZ]=size(Z);

% We partition Z=[X;Y]

mX=mR12; %nR11
%nX=nR12
X=Z(1:mX,:);
%mY=mZ-mX;
%nY=nX;
Y=Z(mX+1:mZ,:);

% Step 5:  C0=embedding(R21) and C1=C0(i,:)*r;
[W_R21C0,Q_R21C0,U_R21C0,detU_R21C0,deteW_R21C0]=embedding(R21);
C0=Q_R21C0;
[mC0,nC0]=size(C0);

% We construct C1 as follows.  We compute C0(i,:)*r where i<=mC0.
% Let us take i=1.  Then C1=[C0(1,:)*r;C0((2:mC0),:)].
C1=[C0(1,:)*r;C0((2:mC0),:)];

% The controller is given by C.
C=C1*X.';

% REMARK: It has to be true that:  det([R21;C1])=r
disp('Poles placed at:')
roots(det([R21;C1]))
```

Note that the algorithm has the same name as the one for full interconnection in chapter 3. The actual algorithm combines the two by counting the number of input arguments. We illustrate the algorithm by applying it to a simple example.

Example 20 *Consider the system given by* $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w + R_2(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ *where the polynomial matrices* R_1 *and* R_2 *are defined by*

$$R_1(\xi) = \begin{bmatrix} \xi+1 & \xi+2 & \xi+3 \\ \xi+4 & \xi+5 & \xi+6 \end{bmatrix}, \quad R_2(\xi) = \begin{bmatrix} \xi+7 & \xi+8 & \xi+9 \\ 3.1\xi+22 & 3.1\xi+25 & 3.1\xi+28 \end{bmatrix}$$

Let the desired polynomial be given by $r(\xi) = \xi + 77$.

Running 8
R2rowred =

$$\begin{matrix} 22+3.1s & 25+3.1s & 28+3.1s \\ 0 & 0 & 0 \end{matrix}$$

rR2rowred =

1

Constant polynomial matrix: 2-by-2

U =

$$\begin{matrix} 0 & 1 \\ 1 & -0.32 \end{matrix}$$

R12 =

$$22+3.1s \quad 25+3.1s \quad 28+3.1s$$

Zero =

$$0 \quad 0 \quad 0$$

R11 =

$$4+s \quad 5+s \quad 6+s$$

R21 =

$$-0.27+0.68s \quad 0.41+0.68s \quad 1.1+0.68s$$

Wemb =

$$\begin{array}{lll} 4+s & 5+s & 6+s \\ -0.27+0.68s & 0.41+0.68s & 1.1+0.68s \\ 1-s & -1s & -1s \end{array}$$

Q =

$$\begin{array}{lll} 1-s & -1s & -1s \end{array}$$

Uemb =

$$\begin{array}{lll} s & 1-s & 0 \\ s & -1s & 1 \\ 1 & -2 & 1 \end{array}$$

Constant polynomial matrix: 1-by-1

detUemb =

1

Constant polynomial matrix: 1-by-1

deteWemb =

3

C0 =

$$\begin{array}{lll} 1-s & -1s & -1s \\ 1 & 1 & 0 \end{array}$$

C1 =

$$\begin{array}{lll} 77-76s-s^2 & -77s-s^2 & -77s-s^2 \\ 1 & 1 & 0 \end{array}$$

C =

$$\begin{array}{l} 2.2e+03+2.7e+02s+3.1s^2 \\ -71-8.3s \end{array}$$

Poles placed at:

ans =

-77.0000

>>

4.2.2 Stabilization by regular partial interconnection

We now consider the problem of stabilization by regular partial interconnection. This problem is formulated as follows: given $\mathcal{P}_{\text{full}} \in \mathcal{L}^{\mathbb{w}+k}$, find a controller $\mathcal{C} \in \mathcal{L}^k$ such that the corresponding manifest controlled behavior $(\mathcal{K}_{\text{full}}(\mathcal{C}))_w$ is stable, and the interconnection is regular. This problem was studied extensively in [9], and it was shown that for the full plant behavior $\mathcal{P}_{\text{full}}$ such controller exists if and only if the manifest plant behavior $(\mathcal{P}_{\text{full}})_w$ is stabilizable and the hidden behavior $\mathcal{N}$ is stable.
In this subsection we will establish an algorithm to compute a required polynomial matrix C (representing a controller $\mathcal{C}$) in case that $\mathcal{P}_{\text{full}}$ is represented by the minimal kernel representation $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w + R_2(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$.
Let U be a unimodular matrix that leads to the polynomial matrices R_{11}, R_{12} and R_{21} as in the previous subsection. Again, $\mathcal{P}$ is represented by $R_{21}(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ and $\mathcal{N}$ by (4.2). We assume that $\mathcal{P}$ is stabilizable and $\mathcal{N}$ is stable. Then $R_{21}(\lambda)$ has full row rank for all $\lambda \in \mathbb{C}^+$ and $[R_{11}(\lambda); R_{21}(\lambda)]$ has full column rank for all $\lambda \in \mathbb{C}^+$. Hence there exists a Hurwitz polynomial matrix G and polynomial matrices R'_{11} and R'_{21} such that

$$\begin{bmatrix} R_{11} \\ R_{21} \end{bmatrix} = \begin{bmatrix} R'_{11} \\ R'_{21} \end{bmatrix} G \tag{4.5}$$

and such that $[R'_{11}(\lambda); R'_{21}(\lambda)]$ has full column rank for all $\lambda \in \mathbb{C}$. Choose polynomial matrices U_{12} and U_{22} such that

$$\begin{bmatrix} R'_{11} & U_{12} \\ R'_{21} & U_{22} \end{bmatrix} \tag{4.6}$$

is unimodular. Next, solve the polynomial equation (unknowns X and Y)

$$\begin{bmatrix} R'_{11} & U_{12} \\ R'_{21} & U_{22} \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} R_{12} \\ 0 \end{bmatrix}. \tag{4.7}$$

Then we have

$$\begin{bmatrix} R'_{11} & U_{12} \\ R'_{21} & U_{22} \end{bmatrix} \begin{bmatrix} G & X \\ 0 & Y \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & 0 \end{bmatrix},$$

so $\mathcal{P}_{\text{full}}$ also has the minimal kernel representation

$$\begin{bmatrix} G & X(\frac{\mathrm{d}}{\mathrm{d}t}) \\ 0 & Y(\frac{\mathrm{d}}{\mathrm{d}t}) \end{bmatrix} \begin{bmatrix} w \\ c \end{bmatrix} = 0.$$

From this, note that $(w,c) \in \mathcal{P}_{\text{full}}$ implies $G(\frac{\mathrm{d}}{\mathrm{d}t})w = -X(\frac{\mathrm{d}}{\mathrm{d}t})c$. Next, let C_0 be such that $[R'_{21}; C_0]$ is Hurwitz (such C_0 exists since $R'_{21}(\lambda)$ has full row rank for all $\lambda \in \mathbb{C}^+$, see subsection 3.2). Define now the controller behavior $\mathcal{C} \in \mathcal{L}^k$ as the behavior represented by $C(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$, with C defined by $C := C_0X$. Similar as in the previous subsection, it can be shown that the corresponding manifest controlled behavior $\mathcal{K}$ is then represented by

$$\begin{bmatrix} R_{21}(\frac{\mathrm{d}}{\mathrm{d}t}) \\ (C_0G)(\frac{\mathrm{d}}{\mathrm{d}t}) \end{bmatrix} w = 0,$$

and that the interconnection of $\mathcal{P}_{\text{full}}$ with $\mathcal{C}$ is regular. Since $\det[R_{21}; C_0G] = \det[R'_{21}; C_0] \det(G)$, $\mathcal{K}$ is stable. Summarizing, this leads to the following algorithm:

Algorithm 6 Input: The polynomial matrices $R_1(\xi)$ and $R_2(\xi)$.

1. Compute a unimodular matrix U and a full row rank polynomial matrix R_{12}. such that $UR_2 = [R_{12}; 0]$.

2. Compute polynomial matrices R_{11} and R_{21} such that $UR_1 = [R_{11}; R_{21}]$.

3. Compute a Hurwitz polynomial G and polynomial matrices R'_{11} and R'_{21} such that (4.5) holds and such that $[R'_{11}(\lambda); R'_{21}(\lambda)]$ has full column rank for all $\lambda \in \mathbb{C}$.

4. Compute polynomial matrices U_{12} and U_{22} such that the matrix in equation (4.6) is unimodular.

5. Compute polynomial matrices X and Y that solve the polynomial equation (4.7).

6. Compute a polynomial matrix C_0 such that $[R'_{21}; C_0]$ is Hurwitz.

7. Output: $C = C_0X$.

We now give the code of the m-file that computes a required controller.

Listing 9

```
function C=behstab(R1,R2)
% C=behstab_pi(R1,R2)
% Algorithm to compute a stabilizing
% controller by partial interconnection.

% R1, R2 are given horizontally but they are transposed for the
% computations.

% Step 1:  Compute a unimodular matrix U and a frr R12
% s.t.  U*R2=[R12;0];
```

```
% Data:  R1, R2
[m,n]=size([R1  R2]);

% We start considering a polynomial matrix
% given as:  R=[R1 R2],i.e., R is a kernel
% representation given by R1(d/dt)*w+R2(d/dt)*c=0.
% R1 has frr but R2 does not.

% Rowcompression of R2:
% R2rowred=[R12;0],i.e., R2 does not have frr

[R2rowred,rR2rowred,U,UI]=rowred(R2);
[mR2rowred,nR2rowred]=size(R2rowred);
% Step 2:  Compute R11, R21 s.t U*R1=[R11;R21]

% The same transformation defined by U above
% is applied to R1,i.e., we obtain U*R1

UR1=U*R1;
[mUR1,nUR1]=size(UR1);

%R1=[R11;R21];

% We retrieve R11,R12,R21,R22=0=Zero
[R12,Zero]=separate0(R2rowred);

[mR12,nR12]=size(R12);
[mZero,nZero]=size(Zero);

R11=UR1(1:mR12,:);
[mR11,nR11]=size(R11);

R21=UR1(mR12+1:m,:);

if isempty(R21)~=1
else
error('R21 is empty')
end

% Step 3:  We decompose:  R1=Rprima*G where Rprima=[Rprima11;Rprima21]
% where Rprima is fcr and G is a non sing.  matrix.
% We transpose the equation given above which yields:
```

```
% R.'=G.'*Rprima.'

% (because the papers and book computations are done in terms of columns)

[G, Rprima]=ctrbbeh(R1);

[mRprima, nRprima]=size(Rprima);

% Step 4:  U12_22=embedding(Rprima),i.e.,
[Wemb, Q, Uemb, detUemb, deteWemb]=embedding(Rprima);
U12_22emb=Q.';

% Step 5:  Z=axb(Wemb,R2rowred.');

Zz=xab(Wemb.',R2rowred);
if Wemb*Zz.'-R2rowred.'~=zeros(nR2rowred,mR2rowred)
error('xab equation is not satisfied')
else
end

[mZz,nZz]=size(Zz);

% We partition Z=[X;Y]
nX=nR12;

% Partitioning Rprima=[Rprima11;Rprima21]
mRprima11=mR12;
mRprima21=mZero;
%nRprima11=mX

% Step 6:  C0=embedding(Rprima21) s.t [Rprima21;C0] is Hurwitz
% First, we have to retrieve Rprima11,Rprima21:

% Since R1=Rprima*G , mR11=mRprima11 and nR11=nRprima11
mRprima11=mR11;
nRprima11=nR11;

[mR1, nR1]=size(R1);

Rprima11=Rprima(1:mR11,1:nR11);
Rprima21=Rprima(mR11+1:mR1,:);
```

```
[Wemb21,Q,Uemb21,detUemb21,deteWemb21]=embedding(Rprima21);
C0=Q.';

mX=nRprima;

X=Zz(1:mRprima11,:)

%mY=mZ-mX
%Y=Zz(mX+1:mZz,:)
Y=Zz(mRprima11+1:mZz,:);

% Step 7:  C=C0.'*X
C=C0.'*X.'
disp('Stabilized poles at:')
roots(det([R21;(C0*G).']))
```

4.3 All regularly implementing controllers: the observable case

In this section and the subsequent one we deal with the problem of parametrizing all controllers that regularly implement a given behavior. The problem that we study is formulated as follows.
Let $\mathcal{P}_{\text{full}}$ be represented minimally by $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w + R_2(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$. Let $\mathcal{K} \in \mathcal{L}^{\mathbb{w}}$ be a desired behavior, represented minimally by $K(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$. Then the problem is: *give a parametrization, in terms of the polynomial matrices R_1, R_2 and K of* all *polynomial matrices C such that the controller $C(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ regularly implements $\mathcal{K}$.*

Example 21 : Consider the plant behavior $\mathcal{P}_{\text{full}}$ with manifest variable $w = (w_1, w_2)$ and control variable $c = (c_1, c_2)$ represented by

$$\begin{aligned} w_1 + \dot{w}_2 + \dot{c}_1 + c_2 &= 0 \\ c_1 + c_2 &= 0 \end{aligned}$$

Clearly, $(\mathcal{P}_{\text{full}})_w = \mathfrak{C}^\infty(\mathbb{R}, \mathbb{R}^2)$. For the desired behavior $\mathcal{K}$ we take $\mathcal{K} = \{(w_1, w_2) \mid w_1 + \dot{w}_2 = 0\}$. The following controller regularly implements $\mathcal{K}$ through c: $\mathcal{C} = \{(c_1, c_2) \mid \dot{c}_1 + c_2 = 0\}$. Also every controller represented by $kc_1 + c_2 = 0$, with $k \neq 1$, regularly implements $\mathcal{K}$. We would like to find a parametrization of *all* 1×2 polynomial matrices $C(\xi) = [C_1(\xi) \quad C_2(\xi)]$ such that $\mathcal{C} = \{(c_1, c_2) \mid C_1(\frac{\mathrm{d}}{\mathrm{d}t})c_1 + C_2(\frac{\mathrm{d}}{\mathrm{d}t})c_2 = 0\}$ regularly implements $\mathcal{K}$.

We will first assume that in the full plant behavior $\mathcal{P}_{\text{full}}$, c is observable from w. Starting from this assumption, we will in the present section establish a parametrization. Then, in the next section we will lift the observability assumption and describe a parametrization for the general case.
First, let us recall some notation. If $\mathcal{K}_{\text{full}}$ is a subbehavior of $\mathcal{P}_{\text{full}}$, then $(\mathcal{K}_{\text{full}})_w$ denotes the behavior obtained from $\mathcal{K}_{\text{full}}$ by eliminating c. Likewise, $(\mathcal{K}_{\text{full}})_c$ is obtained by eliminating w from $\mathcal{K}_{\text{full}}$. Suppose now that $\mathcal{K}$ is implementable through c w.r.t. $\mathcal{P}_{\text{full}}$. Associated with $\mathcal{K}$ an important role will be played by the subbehavior $\mathcal{L}_{full}$ of $\mathcal{P}_{\text{full}}$ defined as the interconnection of $\mathcal{P}_{\text{full}}$ and $\mathcal{K}$ through w.
We will now first consider the problem of finding one controller $\mathcal{C} \in \mathcal{L}^k$ that implements $\mathcal{K}$. We will derive a representation of such controller in terms of representations of $\mathcal{P}_{\text{full}}$ and $\mathcal{K}$. Let $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w + R_2(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ and $K(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ be kernel representations of $\mathcal{P}_{\text{full}}$ and $\mathcal{K}$, respectively. The behavior $\mathcal{L}_{full}$ introduced above is then represented by the kernel representation

$$\begin{bmatrix} R_1(\frac{\mathrm{d}}{\mathrm{d}t}) & R_2(\frac{\mathrm{d}}{\mathrm{d}t}) \\ K(\frac{\mathrm{d}}{\mathrm{d}t}) & 0 \end{bmatrix} \begin{bmatrix} w \\ c \end{bmatrix} = 0. \tag{4.8}$$

Also note that $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ is a representation of the hidden behavior $\mathcal{N}$. Since $\mathcal{N} \subseteq \mathcal{K}$ we have that, for all w, $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ implies $K(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$. As a consequence, there exists a polynomial matrix F such that $K = FR_1$. Now define a controller behavior $\mathcal{C}^* \in \mathcal{L}^k$ by

$$\mathcal{C}^* := \ker(FR_2(\tfrac{\mathrm{d}}{\mathrm{d}t})) \tag{4.9}$$

This controller behavior indeed implements $\mathcal{K}$:

Lemma 4 $\mathcal{K} = (\mathcal{K}_{\text{full}}(\mathcal{C}^*))_w$.

Proof. Let $\mathcal{L}_{full} \in \mathcal{L}^{\mathbb{w}+k}$ be the interconnection of $\mathcal{P}_{\text{full}}$ and $\mathcal{K}$ through w. We have

$$\begin{bmatrix} R_1 & R_2 \\ 0 & FR_2 \end{bmatrix} = \begin{bmatrix} I_1 & 0 \\ F & -I_2 \end{bmatrix} \begin{bmatrix} R_1 & R_2 \\ K & 0 \end{bmatrix},$$

with I_1 and I_2 identity matrices of appropriate dimensions. Consequently, $\mathcal{L}_{full}$ is equal to the full controlled behavior $\mathcal{K}_{\text{full}}(\mathcal{C}^*)$. Also, since $\mathcal{K} \subseteq \mathcal{P}$, it is easily seen that $\mathcal{K}$ is obtained by eliminating c from $\mathcal{L}_{full}$. Thus we conclude that $\mathcal{K} = (\mathcal{K}_{\text{full}}(\mathcal{C}^*))_w$.

Remark 22 : The controller $\mathcal{C}^*$ represented by $(FR_2)(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ *contains* the 'canonical controller' that was studied in [66], [90] and [8]. Indeed, the behavior $\mathcal{L}_{full}$ above is equal to the interconnection of $\mathcal{P}_{\text{full}}$ and $\mathcal{K}$ through the variable w. By definition, the canonical controller is obtained from this interconnection by eliminating w. This can be done by choosing a unimodular matrix V such that $VR_1 = \text{col}(R_{11}, 0)$ with R_{11} full row rank, and to partition $VR_2 = \text{col}(R_{12}, R_{22})$. Then

$$\begin{bmatrix} R_{11}(\frac{\mathrm{d}}{\mathrm{d}t}) & R_{12}(\frac{\mathrm{d}}{\mathrm{d}t}) \\ 0 & R_{22}(\frac{\mathrm{d}}{\mathrm{d}t}) \\ 0 & (FR_2)(\frac{\mathrm{d}}{\mathrm{d}t}) \end{bmatrix} \begin{bmatrix} w \\ c \end{bmatrix} = 0$$

is a kernel representation of $\mathcal{L}_{full}$. Since R_{11} has full row rank, a representation of the canonical controller is therefore given by :

$$\begin{bmatrix} R_{22}(\frac{\mathrm{d}}{\mathrm{d}t}) \\ (FR_2)(\frac{\mathrm{d}}{\mathrm{d}t}) \end{bmatrix} c = 0.$$

The following lemma states that if c is observable from w, and if a given subbehavior of the manifest plant behavior is obtained by elimination of c from a subbehavior of $\mathcal{P}_{\text{full}}$, then this subbehavior of $\mathcal{P}_{\text{full}}$ is unique:

Lemma 5 *Let $\mathcal{P}_{\text{full}} \in \mathcal{L}^{\mathbb{w}+k}$ with system variable (w, c). Assume that c is observable from w. Let $\mathcal{K}^1_{\text{full}}, \mathcal{K}^2_{\text{full}} \in \mathcal{L}^{\mathbb{w}+k}$ be subbehaviors of $\mathcal{P}_{\text{full}}$. Then we have: $(\mathcal{K}^1_{\text{full}})_w = (\mathcal{K}^2_{\text{full}})_w$ if and only if $\mathcal{K}^1_{\text{full}} = \mathcal{K}^2_{\text{full}}$.*

Proof : Assume $(w, c) \in \mathcal{K}^1_{\text{full}}$. Then $w \in (\mathcal{K}^1_{\text{full}})_w = (\mathcal{K}^2_{\text{full}})_w$, so there exists c' such that $(w, c') \in \mathcal{K}^2_{\text{full}}$. Thus, $(w, c - c') = (w, c) - (w, c') \in \mathcal{P}_{\text{full}}$, so $c = c'$. It follows that $(w, c) \in \mathcal{K}^2_{\text{full}}$. □

As a consequence of the previous lemma we have that if in $\mathcal{P}_{\text{full}}$, c is observable from w, if $\mathcal{K}$ is implementable through c, and if $\mathcal{C}$ is a controller that implements $\mathcal{K}$, then the corresponding full controlled behavior $\mathcal{K}_{\text{full}}(\mathcal{C})$ is in fact equal to $\mathcal{L}_{full}$, the interconnection of $\mathcal{P}_{\text{full}}$ and $\mathcal{K}$ through w:

Lemma 6 *Let $\mathcal{P}_{\text{full}} \in \mathcal{L}^{\mathbb{w}+k}$ with system variable (w, c). Let $\mathcal{K} \in \mathcal{L}^{\mathbb{w}}$ be implementable through c w.r.t. $\mathcal{P}_{\text{full}}$. Let $\mathcal{C}$ be a controller such that $\mathcal{K} = (\mathcal{K}_{\text{full}}(\mathcal{C}))_w$, Then we have:*

1. *$\mathcal{K}_{\text{full}}(\mathcal{C}) \subseteq \mathcal{L}_{full}$.*
2. *If c is observable from w then $\mathcal{K}_{\text{full}}(\mathcal{C}) = \mathcal{L}_{full}$.*

Proof : (1.) If $(w, c) \in \mathcal{K}_{\text{full}}(\mathcal{C})$ then $w \in (\mathcal{K}_{\text{full}}(\mathcal{C}))_w = \mathcal{K}$ Also, $(w, c) \in \mathcal{P}_{\text{full}}$. It follows that $(w, c) \in \mathcal{L}_{full}$. (2.) Clearly $(\mathcal{K}_{\text{full}}(\mathcal{C}))_w = \mathcal{K} = (\mathcal{L}_{full})_w$. If c is observable from w then this implies $\mathcal{K}_{\text{full}}(\mathcal{C}) = \mathcal{L}_{full}$. □

For the special case that, in $\mathcal{P}_{\text{full}}$, c is observable from w, the following theorem reduces the problem of parameterizing all controllers that regularly implement $\mathcal{K}$ via interconnection through c with respect to $\mathcal{P}_{\text{full}}$ to that of parameterizing all controllers that regularly implement $(\mathcal{L}_{full})_c$ via *full interconnection* with respect to $(\mathcal{P}_{\text{full}})_c$:

Theorem 23 : *Let $\mathcal{P}_{\text{full}} \in \mathcal{L}^{\mathbb{w}+k}$, with system variable (w, c). Assume that c is observable from w. Let $\mathcal{K} \in \mathcal{L}^{\mathbb{w}}$ be regularly implementable through c. Let $\mathcal{L}_{full}$ be the interconnection of $\mathcal{P}_{\text{full}}$ and $\mathcal{K}$ through w. Let $\mathcal{C} \in \mathcal{L}^k$. Then the following two statements are equivalent:*

1. *$\mathcal{C}$ regularly implements $\mathcal{K}$ by interconnection through c,*
2. *$\mathcal{C}$ regularly implements $(\mathcal{L}_{full})_c$ via full interconnection w.r.t. $(\mathcal{P}_{\text{full}})_c$.*

Proof : Let $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w + R_2(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ be a minimal kernel representation of $\mathcal{P}_{\text{full}}$. Let V be unimodular such that $VR_1 = \text{col}(R_{11}, 0)$ with R_{11} full row rank. Partition $VR_2 = \text{col}(R_{12}, R_{22})$. Then $R_{22}(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ is a minimal kernel representation of $(\mathcal{P}_{\text{full}})_c$. $\mathcal{K}$ is implementable so there exists a polynomial matrix, say C^* (with C^* any polynomial matrix such that $C^*(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ is a kernel representation of the controller $\mathcal{C}^*$ given by (4.9)), such that

$$\begin{bmatrix} R_{11}(\frac{\mathrm{d}}{\mathrm{d}t}) & R_{12}(\frac{\mathrm{d}}{\mathrm{d}t}) \\ 0 & R_{22}(\frac{\mathrm{d}}{\mathrm{d}t}) \\ 0 & C^*(\frac{\mathrm{d}}{\mathrm{d}t}) \end{bmatrix} \begin{bmatrix} w \\ c \end{bmatrix} = 0$$

is a kernel representation of $\mathcal{L}_{full}$. Hence a kernel representation of $(\mathcal{L}_{full})_c$ is given by

$$\begin{bmatrix} R_{22}(\frac{\mathrm{d}}{\mathrm{d}t}) \\ C^*(\frac{\mathrm{d}}{\mathrm{d}t}) \end{bmatrix} c = 0.$$

(1. $\Rightarrow$ 2.) Assume that $\mathcal{C}$ regularly implements $\mathcal{K}$, i.e., $\mathcal{K}(\mathcal{C}) = \mathcal{K}$ and the interconnection is regular. Let $C(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ be a minimal representation of $\mathcal{C}$. Then the polynomial matrix

$$\begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \\ 0 & C \end{bmatrix} \tag{4.10}$$

has full row rank. We claim that $(\mathcal{P}_{\text{full}})_c \cap \mathcal{C} = (\mathcal{L}_{full})_c$. Indeed, let $R_{22}(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ and $C(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$. Since R_{11} has full row rank, there exists w such that

$$\begin{bmatrix} R_{11}(\frac{\mathrm{d}}{\mathrm{d}t}) & R_{12}(\frac{\mathrm{d}}{\mathrm{d}t}) \\ 0 & R_{22}(\frac{\mathrm{d}}{\mathrm{d}t}) \\ 0 & C(\frac{\mathrm{d}}{\mathrm{d}t}) \end{bmatrix} \begin{bmatrix} w \\ c \end{bmatrix} = 0, \tag{4.11}$$

so $(w, c) \in \mathcal{K}_{\text{full}}(\mathcal{C}) = \mathcal{L}_{full}$. This implies that $c \in (\mathcal{L}_{full})_c$. Conversely, let $c \in (\mathcal{L}_{full})_c$. Then there exists w such that $(w, c) \in \mathcal{L}_{full} = \mathcal{K}_{\text{full}}(\mathcal{C})$, so (4.11) holds. This implies that $R_{22}(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ and $C(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$, so $c \in (\mathcal{P}_{\text{full}})_c \cap \mathcal{C}$. Obviously, $\text{col}(R_{22}, C)$ has full row rank, so the full interconnection of $(\mathcal{P}_{\text{full}})_c$ and $\mathcal{C}$ is regular.
(2. $\Leftarrow$ 1.) Conversely, assume $\mathcal{C}$ regularly implements $(\mathcal{L}_{full})_c$ w.r.t. $(\mathcal{P}_{\text{full}})_c$ by full interconnection, and $C(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ is a minimal kernel representation of $\mathcal{C}$. Then

$$\begin{bmatrix} R_{22}(\frac{\mathrm{d}}{\mathrm{d}t}) \\ C(\frac{\mathrm{d}}{\mathrm{d}t}) \end{bmatrix} c = 0$$

is a minimal kernel representation of $(\mathcal{L}_{full})_c$. We claim that $\mathcal{C}$ regularly implements $\mathcal{K}$ through c. Indeed, $(w, c) \in \mathcal{K}_{\text{full}}(\mathcal{C})$ if and only if $R_{11}(\frac{\mathrm{d}}{\mathrm{d}t})w + R_{12}(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$, $R_{22}(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ and $C(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$. Since $\mathcal{C}$ implements $(\mathcal{L}_{full})_c$ w.r.t. $(\mathcal{P}_{\text{full}})_c$, this is equivalent with $R_{11}(\frac{\mathrm{d}}{\mathrm{d}t})w + R_{12}(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$, $R_{22}(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ and $C^*(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$, which, in turn, is equivalent with $(w, c) \in \mathcal{L}_{full}$. Hence $\mathcal{K}_{\text{full}}(\mathcal{C}) = \mathcal{L}_{full}$, which implies $(\mathcal{K}_{\text{full}}(\mathcal{C}))_w = (\mathcal{L}_{full})_w = \mathcal{K}$. In addition, (4.10) has full row rank so the interconnection is regular. □

A parametrization of all controllers that regularly implement $(\mathcal{L}_{full})_c$ by full interconnection w.r.t. $(\mathcal{P}_{\text{full}})_c$ can be obtained by applying theorem 15. In this way we will now establish, for the case that in $\mathcal{P}_{\text{full}}$ c is observable from w, a parametrization of all controllers that regularly implement a given $\mathcal{K}$ w.r.t. $\mathcal{P}_{\text{full}}$. Let $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w + R_2(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ and $K(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$ be minimal kernel representations of $\mathcal{P}_{\text{full}}$ and $\mathcal{K}$, respectively. We will compute representations of $(\mathcal{P}_{\text{full}})_c$ and $(\mathcal{L}_{full})_c$. As before, let V be unimodular such that $VR_1 = \text{col}(R_{11}, 0)$ with R_{11} full row rank. Partition $VR_2 = \text{col}(R_{12}, R_{22})$. Then $(\mathcal{P}_{\text{full}})_c$ has minimal kernel representation

$$R_{22}(\tfrac{\mathrm{d}}{\mathrm{d}t})c = 0. \tag{4.12}$$

Clearly, the hidden behavior $\mathcal{N}$ is represented by $R_{11}(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$, so there exists a polynomial matrix F_1 such that $K = F_1R_{11}$. As in the proof of lemma 4, $\mathcal{L}_{full}$ is therefore represented by

$$\begin{bmatrix} R_{11}(\frac{\mathrm{d}}{\mathrm{d}t}) & R_{12}(\frac{\mathrm{d}}{\mathrm{d}t}) \\ 0 & R_{22}(\frac{\mathrm{d}}{\mathrm{d}t}) \\ 0 & (F_1R_{12})(\frac{\mathrm{d}}{\mathrm{d}t}) \end{bmatrix} \begin{bmatrix} w \\ c \end{bmatrix} = 0,$$

so a kernel representation of $(\mathcal{L}_{full})_c$ is given by

$$\begin{bmatrix} R_{22}(\frac{\mathrm{d}}{\mathrm{d}t}) \\ (F_1R_{12})(\frac{\mathrm{d}}{\mathrm{d}t}) \end{bmatrix} c = 0. \tag{4.13}$$

A parametrization of all controllers $\mathcal{C}$ that regularly implement $(\mathcal{L}_{full})_c$ by full interconnection w.r.t $(\mathcal{P}_{\text{full}})_c$ is now obtained as follows. First, let M be a MRA of R_{22} with $M(\lambda)$ full column rank for all λ. According to theorem 15, the next step is to choose a full row rank MLA of

$$\begin{bmatrix} R_{22} \\ F_1R_{12} \end{bmatrix} M.$$

Since $R_{22}M = 0$, such MLA can be chosen of the form

$$\begin{bmatrix} I & 0 \\ 0 & Q_2 \end{bmatrix},$$

with Q_2 a full row rank MLA of $F_1R_{12}M$. The next step is to choose $W = (W_1, W_2)$ such that

$$\begin{bmatrix} I & 0 \\ 0 & Q_2 \\ W_1 & W_2 \end{bmatrix}$$

is unimodular. Obviously, here it suffices to choose $W_1 = 0$ and W_2 such that $\text{col}(Q_2, W_2)$ is unimodular. Summarizing, writing Q instead of Q_2, the following steps lead to a desired parametrization:

1. choose a MRA M of R_{22} with $M(\lambda)$ full column rank for all λ,
2. choose a full row rank MLA Q of $F_1R_{12}M$,

3. choose W such that $\mathrm{col}(Q, W)$ is unimodular,

4. a parametrization of all controllers that regularly implement $(\mathcal{L}_{full})_c$ by full interconnection w.r.t $(\mathcal{P}_{\mathrm{full}})_c$, equivalently that regularly implement $\mathcal{K}$ through c w.r.t. $\mathcal{P}_{\mathrm{full}}$, is then given by $C = GR_{22} + UWF_1R_{12}$, where G ranges over all polynomial matrices and U ranges over all unimodular polynomial matrices, of course of suitable dimensions.

Of course, our ultimate goal is to obtain a parametrization in terms of the original system parameters R_1, R_2 and K. This goal is achieved in the following theorem:

Theorem 24 : *Let $\mathcal{P}_{\mathrm{full}} \in \mathcal{L}^{\mathtt{w}+k}$, with minimal kernel representation $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w + R_2(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$. Assume that c is observable from w. Let $\mathcal{K} \in \mathcal{L}^{\mathtt{w}}$, with minimal kernel representation $K(\frac{\mathrm{d}}{\mathrm{d}t})w = 0$, be regularly implementable through c w.r.t. $\mathcal{P}_{\mathrm{full}}$. Construct polynomial matrices V_1, V_2, F_1 and W as follows:*

1. *let V_2 be a full row rank MLA of R_1,*

2. *choose V_1 such that $\mathrm{col}(V_1, V_2)$ is unimodular,*

3. *let M be a MRA of V_2R_2 with $M(\lambda)$ full column rank for all λ,*

4. *let F_1 be such that $K = F_1V_1R_1$,*

5. *let Q be a full row rank MLA of $F_1V_1R_2M$,*

6. *choose W such that $\mathrm{col}(Q, W)$ is unimodular.*

Then for any $\mathcal{C} \in \mathcal{L}^k$ represented by $C(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ the following statements are equivalent:

1. *$\mathcal{C}$ has minimal kernel representation $C(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ and regularly implements $\mathcal{K}$ through c with respect to $\mathcal{P}_{\mathrm{full}}$,*

2. *there exists a polynomial matrix G and a unimodular U such that*

$$C = (UWF_1V_1 + GV_2)R_2.$$

Proof : If V_2 is a full row rank MLA of R_1 then the unimodular matrix $V = \mathrm{col}(V_1, V_2)$ satisfies $VR_1 = \mathrm{col}(V_1R_1, 0)$ with V_1R_1 full row rank. Also $VR_2 = \mathrm{col}(V_1R_2, V_2R_2)$. The proof then follows by identifying $R_{11} = V_1R_1$, $R_{12} = V_1R_2$ and $R_{22} = V_2R_2$. □

In order to implement the latter theorem, we have the following code.

Listing 10

```
function [W,F1,V1,V2]=behallimpl(R1,R2,K)
% [W,F1,V1,V2]=behallimpl(R1,R2,K)
% All controllers that regularly implement a given behavior ker(K):
% The observable case.
% There exists a polynomial matrix G and a unimodular U such that
% C=(U*W*F1*V1+G*V2)*R2
% Assume R1, R2 and K to be given.

V2=(null(R1'))'
V1=embedding(V2)
M=null(V2*R2)

F1=xab(V1*R1,K)
Q=(null(F1*V1*R2*M)')'
W=embedding(Q)

disp('C(d/dt)c=0 regularly implements ker(K) by partial interconnection iff')
disp('there exists a polynomial matrix G and a unimodular U such that')
disp('C=(U*W*F1*V1+G*V2)R2 ')
```

4.4 All regularly implementing controllers: the nonobservable case

We will now treat the nonobservable case. Again consider the system $\mathcal{P}_{\text{full}}$ represented by the minimal kernel representation $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w + R_2(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$. We will no longer assume that c is observable from w. We will however show that the general case can be reduced to the observable case. This reduction requires two steps. First, we will reduce the general case to the case that R_2 has full column rank. Next we will reduce the latter to the case that $R_2(\lambda)$ has full column rank for all λ, i.e. the observable case.

1. *Reduction to the case that R_2 has full column rank.* Let V be a unimodular matrix such that
$$R_2 = \begin{bmatrix} \tilde{R}_2 & 0 \end{bmatrix} V,$$
with $\tilde{R}_2$ full column rank k'. Define the new system $\mathcal{P}'_{\text{full}} \in \mathcal{L}^{\mathbb{w}+k'}$ as the system (with control variable c') represented by
$$R_1(\tfrac{\mathrm{d}}{\mathrm{d}t})w + \tilde{R}_2(\tfrac{\mathrm{d}}{\mathrm{d}t})c' = 0.$$

2. *Reduction to the observable case.* Assume now that in $\mathcal{P}_{\text{full}}$ the matrix R_2 has full column rank. Let L be a square, nonsingular polynomial matrix such that $R_2 = \tilde{R}_2 L$,

with $\tilde{R}_2(\lambda)$ full column rank for all $\lambda \in \mathbb{C}$. Define the new system $\mathcal{P}'_{\text{full}}$ as the system (with control variable c') represented by

$$R_1(\tfrac{\mathrm{d}}{\mathrm{d}t})w + \tilde{R}_2(\tfrac{\mathrm{d}}{\mathrm{d}t})c' = 0.$$

In the system $\mathcal{P}'_{\text{full}}$, c' is observable from w.

It will turn out that, in both reduction steps, $\mathcal{K} \in \mathcal{L}^{\mathbb{w}}$ is regularly implementable through c w.r.t. $\mathcal{P}_{\text{full}}$, if and only if it is regularly implementable through c' w.r.t. $\mathcal{P}'_{\text{full}}$. Also, every controller that regularly implements $\mathcal{K}$ w.r.t $\mathcal{P}'_{\text{full}}$ will turn out to lead to a *set* of controllers that implement $\mathcal{K}$ w.r.t $\mathcal{P}_{\text{full}}$. In the following two subsections, we will treat the two reduction steps separately.

4.4.1 Reduction to the case that R_2 has full column rank

In this subsection the parametrization problem for the original plant $\mathcal{P}_{\text{full}}$ will be reduced to the parametrization problem for a plant $\mathcal{P}'_{\text{full}}$ in which the R_2-matrix has full column rank. In the following, let V be a unimodular matrix such that $R_2 = [\tilde{R}_2 \quad 0]V$, with $\tilde{R}_2$ full column rank $k' = \text{rank}(R_2)$. Let $\mathcal{P}'_{\text{full}}$ be the system represented by $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w + \tilde{R}_2(\frac{\mathrm{d}}{\mathrm{d}t})c' = 0$.

Theorem 25 : *Let $\mathcal{K} \in \mathcal{L}^{\mathbb{w}}$. Then $\mathcal{K}$ is regularly implementable through c w.r.t. $\mathcal{P}_{\text{full}}$ if and only if $\mathcal{K}$ is regularly implementable through c' w.r.t. $\mathcal{P}'_{\text{full}}$. Let $\mathcal{C} \in \mathcal{L}^k$ be represented by the minimal kernel representation $C(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$. Then the following two statements are equivalent:*

1. *the controller $\mathcal{C}$ regularly implements $\mathcal{K}$ through c w.r.t $\mathcal{P}_{\text{full}}$,*
2. *there exist a polynomial matrix C_{11}, polynomial matrices C_{12} and C_{21} of full row rank, and a unimodular matrix U such that*

$$C = U \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & 0 \end{bmatrix} V, \tag{4.14}$$

and such that the controller $\mathcal{C}_{21} \in \mathcal{L}^{k'}$ represented by $C_{21}(\frac{\mathrm{d}}{\mathrm{d}t})c' = 0$ regularly implements $\mathcal{K}$ through c' w.r.t. $\mathcal{P}'_{\text{full}}$.

Proof : The statement that $\mathcal{K}$ is regularly implementable through c w.r.t. $\mathcal{P}_{\text{full}}$ if and only if $\mathcal{K}$ is regularly implementable through c' w.r.t. $\mathcal{P}'_{\text{full}}$ is easy to prove: $\mathcal{P}_{\text{full}}$ and $\mathcal{P}'_{\text{full}}$ share the same hidden behavior $\mathcal{N}$ and the same manifest plant behavior $(\mathcal{P}'_{\text{full}})_w = (\mathcal{P}_{\text{full}})_w$.
(1. $\Rightarrow$ 2.) If $\mathcal{C}$ regularly implements $\mathcal{K}$ then (omitting the $\frac{\mathrm{d}}{\mathrm{d}t}$'s) then $\mathcal{K} = (\mathcal{K}_{\text{full}}(\mathcal{C}))_w$ with $\mathcal{K}_{\text{full}}(\mathcal{C}) = \{(w, c) \mid R_1 w + R_2 c = 0 \text{ and } Cc = 0\}$ and

$$\begin{bmatrix} R_1 & R_2 \\ 0 & C \end{bmatrix}$$

full row rank. Partition $CV^{-1} = [V_1 \quad V_2]$ with the number of columns of V_1 equal to $k' = \text{rank}(R_2)$. Choose a unimodular matrix U such that $U^{-1}C_2 = \text{col}(C_{12}, 0)$ with C_{12} full row rank. Partition $U^{-1}C_1 = \text{col}(C_{11}, C_{21})$. Since $[C_1 \quad C_2]$ has full row rank, also C_{21} has full row rank. Moreover, (4.14) holds. We claim that the controller $\mathcal{C}_{21}$ represented by $C_{21}(\frac{\mathrm{d}}{\mathrm{d}t})c' = 0$ regularly implements $\mathcal{K}$ w.r.t. $\mathcal{P}'_{\text{full}}$. Indeed, denote by

$$\mathcal{K}'_{\text{full}}(\mathcal{C}_{21}) := \{(w, c_1) \mid R_1 w + \tilde{R}_2 c_1 = 0 \text{ and } C_{21}c_1 = 0\}.$$

the full controlled behavior of $\mathcal{P}'_{\text{full}}$ using the controller $\mathcal{C}_{21}$. Since C_{12} has full row rank,

$$\begin{aligned}
&\{w \mid \text{ there exists } c_1 \text{ s.t. } R_1 w + \tilde{R}_2 c_1 = 0,\ C_{21}c_1 = 0\} \\
&= \{w \mid \text{ there exists } c_1, c_2 \text{ s.t. } R_1 w + \tilde{R}_2 c_1 = 0,\ C_{11}c_1 + C_{12}c_2 = 0,\ C_{21}c_1 = 0\} \\
&= \{w \mid \text{ there exists } c_1, c_2 \text{ s.t. } R_1 w + R_2 V^{-1}\text{col}(c_1, c_2) = 0,\ U^{-1}CV^{-1}\text{col}(c_1, c_2) = 0\} \\
&= \{w \mid \text{ there exists } c \text{ s.t. } R_1 w + R_2 c = 0,\ Cc = 0\ \}.
\end{aligned}$$

Thus we obtain $(\mathcal{K}'_{\text{full}}(\mathcal{C}_{21}))_w = (\mathcal{K}_{\text{full}}(\mathcal{C}))_w = \mathcal{K}$. Finally, using the fact that C_{12} has full row rank,

$$\begin{bmatrix} R_1 & R_2 \\ 0 & C \end{bmatrix} \tag{4.15}$$

has full row rank if and only if

$$\begin{bmatrix} R_1 & \tilde{R}_2 \\ 0 & C_{21} \end{bmatrix} \tag{4.16}$$

has full row rank. Hence the interconnection of $\mathcal{C}_{12}$ and $\mathcal{P}'_{\text{full}}$ is regular if and only if the interconnection of $\mathcal{C}$ and $\mathcal{P}_{\text{full}}$ is regular.
(2. $\Rightarrow$ 1.) Conversely, if (4.14) holds then by reversing the above argument we see that if the controller $C_{21}c' = 0$ regularly implements $\mathcal{K}$ w.r.t $\mathcal{P}'_{\text{full}}$, then $Cc = 0$ regularly implements $\mathcal{K}$ w.r.t $\mathcal{P}_{\text{full}}$. □

4.4.2 Reduction to the observable case

In the previous subsection it was shown that our parametrization problem can be reduced to a problem for a plant behavior with R_2-matrix full column rank. In the present subsection we will reduce the full column rank case to the observable case. Let $\mathcal{P}_{\text{full}}$ be represented by the minimal kernel representation $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w + R_2(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$, with R_2 full column rank. Let L be square, nonsingular such that $R_2 = \tilde{R}_2 L$, with $\tilde{R}_2(\lambda)$ full column rank for all λ. Let $\mathcal{P}'_{\text{full}}$ be the (observable) system represented by $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w + \tilde{R}_2(\frac{\mathrm{d}}{\mathrm{d}t})c' = 0$.

Theorem 26 : *Let $\mathcal{K} \in \mathcal{L}^{\mathbf{w}}$. Then $\mathcal{K}$ is regularly implementable through c w.r.t. $\mathcal{P}_{\text{full}}$ if and only if $\mathcal{K}$ is regularly implementable through c' w.r.t. $\mathcal{P}'_{\text{full}}$. Let $\mathcal{C} \in \mathcal{L}^k$ be represented by the minimal kernel representation $C(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$. Then the following two statements are equivalent:*

1. *the controller $\mathcal{C}$ regularly implements $\mathcal{K}$ through c w.r.t $\mathcal{P}_{\text{full}}$,*
2. *the controller $\mathcal{C}'$ represented in latent variable representation (with latent variable ℓ) by*
$$\begin{bmatrix} I \\ 0 \end{bmatrix} c' = \begin{bmatrix} L(\frac{\mathrm{d}}{\mathrm{d}t}) \\ C(\frac{\mathrm{d}}{\mathrm{d}t}) \end{bmatrix} \ell \tag{4.17}$$
regularly implements $\mathcal{K}$ through c' w.r.t. $\mathcal{P}'_{\text{full}}$.

Proof : Again, it is easily verified that the hidden behavior and manifest plant behavior of $\mathcal{P}_{\text{full}}$ and $\mathcal{P}'_{\text{full}}$ coincide. This proves the first statement.
The full controlled behavior resulting from the interconnection of $\mathcal{P}_{\text{full}}$ and $\mathcal{C}$ is equal to

$$\begin{aligned}\mathcal{K}_{\text{full}}(\mathcal{C}) &= \{w \mid \text{ there exists } c \text{ such that } R_1 w + R_2 c = 0,\, Cc = 0\} \\ &= \{w \mid \text{ there exists } c \text{ such that } R_1 w + \tilde{R}_2 L c = 0,\, Cc = 0\}.\end{aligned}$$

Since L is nonsingular this behavior equals

$$\{w \mid \text{ there exists } c', c \text{ such that } R_1 w + \tilde{R}_2 c' = 0,\, c' = Lc,\, Cc = 0\},$$

which is equal to $\mathcal{K}'_{\text{full}}(\mathcal{C}')$, the full controlled behavior resulting from the interconnection of $\mathcal{P}'_{\text{full}}$ and $\mathcal{C}'$. Clearly, $\mathcal{C}$ implements $\mathcal{K}$ w.r.t $\mathcal{P}_{\text{full}}$ if and only if $\mathcal{K} = (\mathcal{K}_{\text{full}}(\mathcal{C}))_w$, while $\mathcal{C}'$ implements $\mathcal{K}$ w.r.t $\mathcal{P}'_{\text{full}}$ if and only if $\mathcal{K} = (\mathcal{K}_{\text{full}}(\mathcal{C}'))_w$. This shows that $\mathcal{C}$ implements $\mathcal{K}$ w.r.t $\mathcal{P}_{\text{full}}$ if and only if $\mathcal{C}'$ implements $\mathcal{K}$ w.r.t $\mathcal{P}'_{\text{full}}$.
Next, we will prove that the interconnection of $\mathcal{P}_{\text{full}}$ and $\mathcal{C}$ is regular if and only if the interconnection of $\mathcal{P}'_{\text{full}}$ and $\mathcal{C}'$ is regular. Note that $\mathcal{K}'_{\text{full}}(\mathcal{C}')$ has latent variable representation

$$\begin{bmatrix} R_1 & \tilde{R}_2 \\ 0 & I \\ 0 & 0 \end{bmatrix} \begin{bmatrix} w \\ c' \end{bmatrix} = \begin{bmatrix} 0 \\ L \\ C \end{bmatrix} \ell.$$

Hence the output cardinality of $\mathcal{K}'_{\text{full}}(\mathcal{C}')$ equals

$$p(\mathcal{K}'_{\text{full}}(\mathcal{C}')) = \operatorname{rank} \begin{bmatrix} R_1 & \tilde{R}_2 & 0 \\ 0 & I & L \\ 0 & 0 & C \end{bmatrix} - \operatorname{rank} \begin{bmatrix} 0 \\ L \\ C \end{bmatrix}.$$

Using elementary row and column operations and the fact that L is nonsingular, this can be shown to be equal to

$$\operatorname{rank} \begin{bmatrix} R_1 & R_2 \\ 0 & C \end{bmatrix} = p(\mathcal{K}_{\text{full}}(\mathcal{C})).$$

Also,

$$p(\mathcal{C}') = \operatorname{rank} \begin{bmatrix} I & L \\ 0 & C \end{bmatrix} - \operatorname{rank} \begin{bmatrix} L \\ C \end{bmatrix} = \operatorname{rank}(C) = p(\mathcal{C}).$$

Finally, $p(\mathcal{P}_{\text{full}}) = \operatorname{rank}[R_1 \quad R_2] = \operatorname{rank}[R_1 \quad \tilde{R}_2] = p(\mathcal{P}'_{\text{full}})$. This proves our claim. □

According to this theorem, a controller represented by $C(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ works for $\mathcal{P}_{\text{full}}$ if and only if the controller $c' = L(\frac{\mathrm{d}}{\mathrm{d}t})\ell$, $C(\frac{\mathrm{d}}{\mathrm{d}t})\ell = 0$ (with control variable c') works for the observable system $\mathcal{P}'_{\text{full}}$. What we are looking for here is a parametrization of *all* such polynomial matrices C. Now, we do already have a parametrization of all controllers $C'(\frac{\mathrm{d}}{\mathrm{d}t})c' = 0$ that work for $\mathcal{P}'_{\text{full}}$. Indeed, this parametrization was established in theorem 24. Hence the question arises under what conditions the latent variable representation $c' = L(\frac{\mathrm{d}}{\mathrm{d}t})\ell, C(\frac{\mathrm{d}}{\mathrm{d}t})\ell = 0$ and the kernel representation $C'(\frac{\mathrm{d}}{\mathrm{d}t})c' = 0$ represent the same behavior $\mathcal{C}'$. The answer to this is given in the following lemma:

Lemma 7 *Let L be a $k \times k$, square, nonsingular polynomial matrix. Let C and C' be polynomial matrices with k columns. Then the latent variable representation $c' = L(\frac{\mathrm{d}}{\mathrm{d}t})\ell, C(\frac{\mathrm{d}}{\mathrm{d}t})\ell = 0$ and the kernel representation $C'(\frac{\mathrm{d}}{\mathrm{d}t})c' = 0$ represent the same behavior if and only if* $\ker(C'L) = \ker(L) + \ker(C)$.

Proof : ($\Rightarrow$) Let $C'L\ell = 0$. Then $c' := L\ell \in \ker(C')$, so there exists ℓ' such that $c' = L(\frac{\mathrm{d}}{\mathrm{d}t})\ell', C(\frac{\mathrm{d}}{\mathrm{d}t})\ell' = 0$. Define $\ell'' := \ell - \ell'$ Then $\ell = \ell' + \ell'' \in \ker(C) + \ker(L)$. Conversely, let $C\ell = 0$. Define $c' = L\ell$. Then $C'c' = 0$ so $C'L\ell = 0$. ($\Leftarrow$) Assume $C'c' = 0$. Let ℓ be such that $c' = L\ell$. Then $C'L\ell = 0$ so there exists $\ell' \in \ker(C)$ and $\ell'' \in \ker(L)$ such that $\ell = \ell' + \ell''$. This implies that $c' = L\ell'$, while $C\ell' = 0$. Conversely, assume $c' = L\ell$ with $C\ell = 0$. Then clearly $C'L\ell = 0$ so $C'c' = 0$. □

Corollary 3 *The controller represented by $C(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ regularly implements $\mathcal{K}$ through c w.r.t. $\mathcal{P}_{\text{full}}$ if and only if there exists a polynomial matrix C' such that $C'(\frac{\mathrm{d}}{\mathrm{d}t})c' = 0$ regularly implements $\mathcal{K}$ through c' w.r.t. $\mathcal{P}'_{\text{full}}$ and C satisfies* $\ker(C) + \ker(L) = \ker(C'L)$.

Since we already have a parametrization of all polynomial matrices C' such that the controller $C'(\frac{\mathrm{d}}{\mathrm{d}t})c' = 0$ regularly implements $\mathcal{K}$ w.r.t. $\mathcal{P}'_{\text{full}}$, a parametrization of all controllers that implement $\mathcal{K}$ through c w.r.t. $\mathcal{P}_{\text{full}}$ can be obtained by parameterizing *for fixed C'* all polynomial matrices C such that $\ker(C) + \ker(L) = \ker(C'L)$. Such parametrization can be derived from the following theorem:

Theorem 27 : *Let L be a $k \times k$, square, nonsingular polynomial matrix. Let C and C' be full row rank polynomial matrices with k columns. Then* $\ker(C'L) = \ker(L) + \ker(C)$ *if and only if there exists a square, nonsingular polynomial matrix X such that $C'L = XC$ and*

$$\begin{bmatrix} -X(\lambda) & C'(\lambda) \end{bmatrix}$$

has full row rank for all $\lambda \in \mathbb{C}$.

Proof : ($\Rightarrow$) Let $\begin{bmatrix} -X & Y \end{bmatrix}$ be a full row rank MLA of $\mathrm{col}(C, L)$. Then by proposition 10, $\ker(XC) = \ker(YL) = \ker(C'L)$. We claim that XC has full row rank. Indeed, if p is a polynomial row vector such that $pXC = 0$ then also $pX = 0$. Since $XC = YL$ and L is nonsingular, also $pY = 0$. Since $\begin{bmatrix} -X & Y \end{bmatrix}$ has full row rank this yields $p = 0$. Thus XC

and $C'L$ yield minimal representations of the same behavior so there exists a unimodular U such that $C'L = UXC$. This implies $YL = U^{-1}C'L$ so, by the nonsingularity of L, $Y = U^{-1}C'$. Define $\tilde{X} = UX$. Then $[-\tilde{X} \quad C']$ is a full row rank MLA of $\mathrm{col}(C, L)$. This implies $C'L = \tilde{X}C$. Also, $[-\tilde{X}(\lambda) \quad C'(\lambda)]$ has full row rank for all λ. Finally, $\tilde{X}$ is square and nonsingular. Clearly, X has full row rank. Also,

$$\mathrm{rowdim}(X) = \mathrm{rowdim}(C) + \mathrm{rowdim}(L) - \mathrm{rank}[C \quad L].$$

Since L is nonsingular, $\mathrm{rank}[C \quad L] = \mathrm{rank}(L)$, so $\mathrm{rowdim}(X) = \mathrm{rowdim}(C)$. Of course, also $\mathrm{coldim}(X) = \mathrm{rowdim}(C)$, so X is square and nonsingular. The same then holds for $\tilde{X}$.
($\Leftarrow$) We have $XC - C'L = 0$ and $[-X(\lambda) \quad C'(\lambda)]$ has full row rank for all λ. It is easily verified that $\mathrm{rank}[X \quad C'] = \mathrm{coldim}(X, C') - \mathrm{rank}[C \quad L]$, so $[X \quad C']$ is a full row rank MLA of $\mathrm{col}(C, L)$. By proposition 10, this implies that $\ker(C) + \ker(L) = \ker(C'L)$ as desired. □

Corollary 4 *Let $\mathcal{P}_{\mathrm{full}}$ be represented by the minimal kernel representation $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w + R_2(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$, with R_2 full column rank. Let L be square, nonsingular such that $R_2 = \tilde{R}_2 L$, with $\tilde{R}_2(\lambda)$ full column rank for all λ. Let $\mathcal{P}'_{\mathrm{full}}$ be the (observable) system represented by $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w + \tilde{R}_2(\frac{\mathrm{d}}{\mathrm{d}t})c' = 0$. Let $\mathcal{C} \in \mathcal{L}^k$ be represented by the minimal kernel representation $C(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$. Then for every $\mathcal{K} \in \mathcal{L}^{\mathbb{q}}$ that is regularly implementable through c w.r.t. $\mathcal{P}_{\mathrm{full}}$ the following two statements are equivalent:*

1. *the controller $\mathcal{C}$ regularly implements $\mathcal{K}$ through c w.r.t $\mathcal{P}_{\mathrm{full}}$,*
2. *there exists a square, nonsingular polynomial matrix X and a full row rank polynomial matrix C' such that*
$$C = X^{-1}C'L,$$
where $(X(\lambda), C'(\lambda))$ has full row rank for all $\lambda \in \mathbb{C}$ and the controller $\mathcal{C}'$ represented by $C'(\frac{\mathrm{d}}{\mathrm{d}t})c' = 0$ regularly implements $\mathcal{K}$ through c' w.r.t. $\mathcal{P}'_{\mathrm{full}}$.

Thus, for any given full row rank C' that works for the observable system $\mathcal{P}'_{\mathrm{full}}$, a *set* of polynomial matrices C that work for $\mathcal{P}_{\mathrm{full}}$ is obtained by dividing $C'L$ by those nonsingular polynomial matrices X that have the the properties that $(X(\lambda), C'(\lambda))$ has full row rank for all λ, and the quotient $X^{-1}C'L$ is a polynomial matrix again.

4.5 All stabilizing controllers

In this section we return to the stabilization problem. We will solve the problem of parametrizing, for a given plant $\mathcal{P}_{\mathrm{full}}$, all stabilizing controllers. This will be done along the same lines as the parametrization of all regularly implementing controllers: we will first establish a parametrization under the condition that in $\mathcal{P}_{\mathrm{full}}$ c is observable from w. Then we will lift the assumption and treat the general case. For the observable case the following lemma is instrumental:

Lemma 8 *Let $\mathcal{P}_{\text{full}} \in \mathcal{L}^{\mathbb{w}+k}$ with system variable (w, c). Assume that c is observable from w. Assume that $(\mathcal{P}_{\text{full}})_w$ is stabilizable and thatw is detectable from c. Let $\mathcal{C} \in \mathcal{L}^k$. Then the following two statements are equivalent:*

1. *$\mathcal{C}$ stabilizes $\mathcal{P}_{\text{full}}$ through c,*
2. *$\mathcal{C}$ stabilizes $(\mathcal{P}_{\text{full}})_c$ by full interconnection.*

Proof : (1. $\Rightarrow$ 2.) $\mathcal{K} := (\mathcal{K}_{\text{full}}(\mathcal{C}))_w$ is stable and the interconnection is regular. Let $\mathcal{L}_{full}$ be the interconnection of $\mathcal{P}_{\text{full}}$ and $\mathcal{K}$ through w. Then, by observability, $\mathcal{K}_{\text{full}}(\mathcal{C}) = \mathcal{L}_{full}$. According to theorem 23, $\mathcal{C}$ regularly implements $(\mathcal{L}_{full})_c$ by full interconnection with $(\mathcal{P}_{\text{full}})_c$. We claim that $(\mathcal{L}_{full})_c$ is stable. Indeed, let $c \in (\mathcal{L}_{full})_c$. There exists w such that $(w, c) \in \mathcal{L}_{full} \subseteq \mathcal{P}_{\text{full}}$. Let R_2^+ be a polynomial left-inverse of R_2. Then we have $c = -R_2^+ R_1 w$. Hence $c(t) \to 0$ $(t \to \infty)$ (note that the components of w are products of polynomials and stable exponentials).
(2. $\Rightarrow$ 1.) Let $\mathcal{K} := (\mathcal{K}_{\text{full}}(\mathcal{C}))_w$. Let $\mathcal{L}_{full}$ be the interconnection of $\mathcal{P}_{\text{full}}$ and $\mathcal{K}$ through w. Then $\mathcal{L}_{full} = \mathcal{K}_{\text{full}}(\mathcal{C})$ and according to theorem 23, $\mathcal{C}$ regularly implements $(\mathcal{L}_{full})_c$ by full interconnection with $(\mathcal{P}_{\text{full}})_c$. Thus, $(\mathcal{L}_{full})_c$ is stable. We claim that $(\mathcal{L}_{full})_w$ is stable. Indeed, assume that $w \in (\mathcal{L}_{full})_w$. Then there exists c such that $(w, c) \in \mathcal{L}_{full}$. Note that $c \in (\mathcal{L}_{full})_c$, so the components of c are products of polynomials and stable exponentials. Now, (w, c) satisfies $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w + R_2(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$. By detectability of w from c, $R_1(\lambda)$ has full column rank for all $\lambda \in \mathbb{C}^+$. This implies that $w(t) \to 0$ $(t \to \infty)$. □

The following theorem then gives a parametrization of all stabilizing controllers for the observable case:

Corollary 5 *Let $\mathcal{P}_{\text{full}} \in \mathcal{L}^{\mathbb{w}+k}$ satisfy the assumptions of lemma 8. Let $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w + R_2(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ be a minimal kernel representation of $\mathcal{P}_{\text{full}}$. Construct polynomial matrices V_2, S and C_0 as follows:*

1. *let V_2 be a full row rank MLA of R_1,*
2. *factorize $V_2 R_2 = TS$ with T square, nonsingular and $S(\lambda)$ full row rank for all $\lambda \in \mathbb{C}$.*
3. *let C_0 be such that $\operatorname{col}(S, C_0)$ is unimodular*

Then for any $\mathcal{C} \in \mathcal{L}^k$ represented by the kernel representation $C(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ the following statements are equivalent:

1. *$\mathcal{C}$ stabilizes $\mathcal{P}_{\text{full}}$ through c and the kernel representation $C(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ is minimal,*
2. *there exist a polynomial matrix F and a Hurwitz polynomial matrix D such that $C = FS + DC_0$.*

Proof : This is an immediate corollary of lemma 1 and corollary 8. □

The code needed to implement the latter theorem is given below.

Listing 11

```
function C=behallstab(R1,R2)
% C=behallstab(R1,R2)
% All stabilizing controllers:  The partial interconnection case.
% It gives a parametrization of all stabilizing controllers
% for the observable case:  C=F*S+U*D*C0
% R1, R2 are supposed to be given.  [R1  R2] is frr
% Since C is a set of controllers, if such family is obtainable
% the boolean variable C is set to 1.  The parameters of this set C
% are F, U (unimodular), and D (Hurwitz).

if isfullrank([R1  R2])==0
error('[R1  R2] does not satisfy the frr property')
else
end

% V2 is a frr MLA of R1
V2=(null(R1'))'
% Factorize V2*R2=T*S where T is square nonsingular matrix and S is frr.
Vbar=V2*R2
[T,S]=behctrb(Vbar)

if isempty(S)==1,
warning('The set of all stabilizing controllers could not be found')
C=0;
else
C=1;
C0=embedding(S)
disp('Then there exists pol.matrices F, U (unimodular), and D (Hurwitz) s.t.')
disp(' C=F*S+U*D*C0')

end
```

Example 28 : Let us illustrate the above algorithm. Let $\mathcal{P}_{\text{full}}$ be represented by $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w + R_2(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ where the polynomial matrices R_1 and R_2 are given by

$$R_1(\xi) = \begin{bmatrix} 1 & \xi \\ 0 & 0 \end{bmatrix}, \quad R_2(\xi) = \begin{bmatrix} \xi & 1 \\ 1 & 1 \end{bmatrix}.$$

When we run the program we obtain the following result.

Running 9

>> R1

R1 =

1 s
0 0

>> R2

R2 =

s 1
1 1

>> C=behallstab(R1,R2)

Constant polynomial matrix: 1-by-2

V2 =

0 1

Constant polynomial matrix: 1-by-2

Vbar =

1 1

*ker(K) is implementable w.r.t ker(P) according to R=F*K*

Constant polynomial matrix: 1-by-1

T =

1.4

Constant polynomial matrix: 1-by-2

S =

0.71 0.71

C0 =

$0 \quad -1$

C =

1

Then there exist pol.matrices F, U (unimodular), and D (Hurwitz) s.t.
*C=F*S+U*D*C0*
>>

Example 29 Another example is the following. This time $\mathcal{P}_{\text{full}}$ is represented by

$$R_1 = \begin{bmatrix} 1 & \xi \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad R_2 = \begin{bmatrix} \xi & 1 \\ 1 & 1 \\ \xi+1 & \xi+2 \end{bmatrix}$$

The resulting computation is shown below.

Running 10

>> R1

R1 =

$1 \quad s$
$0 \quad 1$
$0 \quad 0$

>> R2

R2 =

$s \quad 1$
$1 \quad 1$
$1+s \quad 2+s$

>> C=behallstab(R1,R2)

Constant polynomial matrix: 1-by-3

V2 =

0 0 1

Vbar =

$1+s \quad 2+s$

*ker(K) is implementable w.r.t ker(P) according to R=F*K*

Constant polynomial matrix: 1-by-1

T =

0.71

S =

$1.4+1.4s \quad 2.8+1.4s$

C0 =

$1 \quad -1$

Then there exists pol.matrices F, U (unimodular), and D (Hurwitz) s.t.
*C=F*S+U*D*C0*

C =

1
>>

Thus we have obtained a parametrization of all stabilizing controllers for the observable case. In order to arrive at a parametrization for the general case, we can perform the same two reduction steps as in section 4.4. We will describe both steps separately now, the proofs are left to the reader.
The first step concerns the reduction of a general $\mathcal{P}_{\text{full}}$ to a full plant behavior $\mathcal{P}'_{\text{full}}$ with R_2-matrix full column rank. Let V be a unimodular matrix such that $R_2 = [\tilde{R}_2 \quad 0]V$, with $\tilde{R}_2$ full column rank. Let $\mathcal{P}'_{\text{full}}$ be represented by $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w + \tilde{R}_2(\frac{\mathrm{d}}{\mathrm{d}t})c' = 0$.

Corollary 6 *$(\mathcal{P}_{\text{full}})_w$ is stabilizable if and only if $(\mathcal{P}'_{\text{full}})_w$ is stabilizable, and in $\mathcal{P}_{\text{full}}$, w is detectable from c if and only in $\mathcal{P}'_{\text{full}}$, w is detectable from c'. Furthermore, if $\mathcal{C} \in \mathcal{L}^k$ is represented by the minimal kernel representation $C(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ then the following two statements are equivalent:*

1. *the controller $\mathcal{C}$ stabilizes $\mathcal{P}_{\text{full}}$ through c,*

2. *there exist a polynomial matrix* C_{11}*, polynomial matrices* C_{12} *and* C_{21} *of full row rank, and a unimodular matrix* U *such that*

$$C = U \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & 0 \end{bmatrix} V, \tag{4.18}$$

and such that the controller $\mathcal{C}_{21} \in \mathcal{L}^{k'}$ *represented by* $C_{21}(\frac{\mathrm{d}}{\mathrm{d}t})c' = 0$ *stabilizes* $\mathcal{P}'_{\text{full}}$ *through* c'*.*

The next step concerns the reduction of a full plant behavior $\mathcal{P}_{\text{full}}$ with full column rank R_2-matrix to a behavior $\mathcal{P}'_{\text{full}}$ in which the control variable c' is observable from w. Let L be square, nonsingular, such that $R_2 = L\tilde{R}_2$, with $\tilde{R}_2(\lambda)$ full column rank for all λ. Let $\mathcal{P}'_{\text{full}}$ be represented by $R_1(\frac{\mathrm{d}}{\mathrm{d}t})w + \tilde{R}_2(\frac{\mathrm{d}}{\mathrm{d}t})c' = 0$.

Corollary 7 $(\mathcal{P}_{\text{full}})_w$ *is stabilizable if and only if* $(\mathcal{P}'_{\text{full}})_w$ *is stabilizable, and in* $\mathcal{P}_{\text{full}}$*,* w *is detectable from* c *if and only in* $\mathcal{P}'_{\text{full}}$*,* w *is detectable from* c'*. Furthermore, if* $\mathcal{C} \in \mathcal{L}^k$ *is represented by the minimal kernel representation* $C(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ *then the following two statements are equivalent:*

1. *the controller* $\mathcal{C}$ *stabilizes* $\mathcal{P}_{\text{full}}$ *through* c*,*

2. *there exists a square, nonsingular polynomial matrix* X *and a full row rank polynomial matrix* C' *such that*

$$C = X^{-1}C'L,$$

where $[X(\lambda) \quad C'(\lambda)]$ *has full row rank for all* $\lambda \in \mathbb{C}$ *and the controller* $\mathcal{C}'$ *represented by* $C'(\frac{\mathrm{d}}{\mathrm{d}t})c' = 0$ *stabilizes* $\mathcal{P}'_{\text{full}}$ *through* c'*.*

4.6 Examples for the nonobservable case

In order to illustrate the theory developed in this chapter on the parametrization in the nonobervable case, in this section we will present some worked-out examples.

Example 30 : Let $\mathcal{P}_{\text{full}}$ with with manifest variable $w = (w_1, w_2)$ and control variable $c = (c_1, c_2)$ be represented by

$$\begin{aligned} w_1 + \dot{w}_2 + \dot{c}_1 + c_2 &= 0 \\ c_1 + c_2 &= 0 \end{aligned}$$

Clearly, $(\mathcal{P}_{\text{full}})_w = \mathfrak{C}^\infty(\mathbb{R}, \mathbb{R}^2)$. For $\mathcal{K}$ we take the behavior represented by $w_1 + \dot{w}_2 = 0$. $\mathcal{K}$ is regularly implementable through (c_1, c_2) w.r.t. $\mathcal{P}_{\text{full}}$. We have

$$R_1(\xi) = \begin{bmatrix} 1 & \xi \\ 0 & 0 \end{bmatrix} \text{ and } R_2(\xi) = \begin{bmatrix} \xi & 1 \\ 1 & 1 \end{bmatrix}.$$

R_2 has full column rank. We factorize $R_2 = \tilde{R}_2 L$ with $\tilde{R}_2 = I_2$, the 2×2 identity matrix, and $L = R_2$. The resulting system $\mathcal{P}'_{\text{full}}$ represented by $R_2 w + \tilde{R}_2 c' = 0$ is observable. We first parameterize all controllers that regularly implement $\mathcal{K}$ w.r.t. $\mathcal{P}'_{\text{full}}$. For this, we perform the steps described in theorem 4.3: $V_2 = [0 \quad 1]$, $V_1 = [1 \quad 0]$, $V_2 R_2 = [1 \quad 1]$ so $M = \text{col}(1, -1)$. Next, $V_1 R_1 = K = [1 \quad \xi]$, so $F_1 = 1$. We have $F_1 V_1 R_2 M = \xi - 1$. Thus Q, as full row rank MLA of $\xi - 1$, is void. We take $W = 1$. A parametrization of all full row rank controllers representations C' that regularly implement $\mathcal{K}$ w.r.t. $\mathcal{P}'_{\text{full}}$ is given by $C'(\xi) = [u \quad g(\xi)]$, with $0 \neq u \in \mathbb{R}$ and g an arbitrary polynomial with real coefficients.

The latter computations can of course also be performed using the MATLAB program *behallimpl* obtained and described before. The resulting running appears below.

Running 11

```
>> help behallimpl
[W,F1,K,V1,V2]=behallimpl(R1,R2)
All controllers that regularly implement a given behavior ker(K):
The observable case.
There exists a polynomial matrix G and a unimodular U such that
C=(U*W*F1*V1+G*V2)R2
Assume R1, R2 and K to be given.


>> [W,F1,K,V1,V2]=behallimpl(R1,R2)

Constant polynomial matrix:  1-by-2

V2 =

0    1

V1=embedding(V2)

>> V1=[1  0]

V1 =
cont_reg_obs
1    0

>> M=null(V2*R2)

[K,rank,F1,F1inv]=rowred(V1*R1,'bas')

Q=(null(F1*V1*R2*M)')'
```

```
W=embedding(Q)


C(d/dt)c=0 regularly implements ker(K) by partial interconnection iff
there exists a polynomial matrix G and a unimodular U such that
C=(U*W*F1*V1+G*V2)R2

Constant polynomial matrix:  2-by-1

M =

-0.71
0.71

K =

1   s

rank =

1
Constant polynomial matrix:  1-by-1

F1 =

1

Constant polynomial matrix:  1-by-1

F1inv =

1

NULL: No polynomial right null space

Q =

Empty polynomial matrix:  1-by-0
```

Continuing the example now, we parametrize all controllers C that regularly implement $\mathcal{K}$ w.r.t. the original full plant behavior $\mathcal{P}_{\text{full}}$. According to corollary 4, for any choice of $u \neq 0$ and polynomial g, we should find all nonzero polynomials $x(\xi)$ that divide $C'L =$

$[u\xi + g(\xi) \quad u + g(\xi)]$ such that $[x(\lambda) \quad u \quad g(\lambda)] \neq 0$ for all $\lambda \in \mathbb{C}$. Since $u \neq 0$, this constraint is automatically satisfied. Thus we only need to compute all common factors $x(\xi)$ of the polynomials $u\xi + g(\xi)$ and $u + g(\xi)$. It is easily seen that if λ is a common root of these two polynomials, then we must have $\lambda = 1$. There are now two possibilities:

1. $g(1) \neq -u$. In this case $u\xi + g(\xi)$ and $u + g(\xi)$ are coprime. The only common factor is then $x(\xi) = 1$.

2. $g(1) = -u$. In this case $x(\xi) = \xi - 1$ is a common factor. It is easily seen that $(\xi - 1)^2$ cannot be a common factor: for this, $\lambda = 1$ should be a common root of the derivatives $u + \frac{dg}{d\xi}(\xi)$ and $\frac{dg}{d\xi}(\xi)$, which is impossible since $u \neq 0$.

Thus we find that a parametrization of all controllers that regularly implement $\mathcal{K}$ for $\mathcal{P}_{\text{full}}$ is given by: $C(\xi) = [u\xi + g(\xi) \quad u + g(\xi))$, $u \neq 0$, g arbitrary polynomial, or $C(\xi) = [\frac{u\xi+g(\xi)}{\xi-1} \quad \frac{u+g(\xi)}{\xi-1}]$, $u \neq 0$ and g arbitrary polynomial such that $g(1) = -u$. Since $g(1) = -u$ if and only if there exists a polynomial h such that $g(\xi) = -u + h(\xi)(\xi - 1)$, the latter is equivalent with: $C(\xi) = [u + h(\xi) \quad h(\xi)]$, $u \neq 0$ and h arbitrary polynomial.

Example 31 : Let $\mathcal{P}_{\text{full}}$ with $w = (w_1, w_2, w_3)$ and $c = (c_1, c_2)$ be represented by

$$\begin{aligned} w_1 + \dot{c}_1 + \ddot{c}_2 &= 0 \\ \ddot{w}_2 + \dot{c}_1 + \ddot{c}_2 &= 0 \\ w_3 + \dot{c}_2 &= 0 \end{aligned}$$

Then

$$R_1(\xi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \xi^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } R_2(\xi) = \begin{bmatrix} \xi^2 & \xi \\ \xi^2 & \xi \\ 0 & \xi \end{bmatrix}.$$

R_1 has full row rank, R_2 has full column rank. $(\mathcal{P}_{\text{full}})_w$ is represented by $w_1 = \ddot{w}_2$. For the behavior to be regularly implemented we take $\mathcal{K}$ represented by $w_1 = \ddot{w}_2, \ddot{w}_1 = 0$. Hence

$$K = \begin{bmatrix} 1 & -\xi^2 & 0 \\ \xi^2 & 0 & 0 \end{bmatrix}.$$

Factor $R_2 = \tilde{R}_2 L$ with

$$\tilde{R}_2(\xi) = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \text{ and } L(\xi) = \begin{bmatrix} \xi^2 & 0 \\ 0 & \xi \end{bmatrix}.$$

The resulting system $\mathcal{P}'_{\text{full}}$ represented by $R_2 w + \tilde{R}_2 c' = 0$ is observable. We first parameterize all controllers that regularly implement $\mathcal{K}$ w.r.t. $\mathcal{P}'_{\text{full}}$. For this, we perform the steps

described in theorem 4.3: V_2 is void, so for V_1 we take I_3. $V_2\tilde{R}_2$ is void, so a suitable MRA is $M = I_2$. Take

$$F_1 = \begin{bmatrix} 1 & -1 & 0 \\ \xi^2 & 0 & 0 \end{bmatrix}.$$

Then $K = F_1V_1\tilde{R}_2$. A full row rank MLA of $F_1V_1\tilde{R}_2M$ is then $Q = [1 \quad 0]$. Take $W = [0 \quad 1]$. Then a parametrization of all full row rank controllers representations C' that regularly implement $\mathcal{K}$ w.r.t. $\mathcal{P}'_{\text{full}}$ is given by $C'(\xi) = u[\xi^2 \quad \xi^2]$, with $0 \neq u \in \mathbb{R}$.
Next we parameterize all controllers C that regularly implement $\mathcal{K}$ w.r.t. the original full plant behavior $\mathcal{P}_{\text{full}}$. We should find all nonzero polynomials $x(\xi)$ that divide $C'L = u[\xi^4 \quad \xi^3]$ such that $[x(\lambda) \quad \lambda^2 \quad \lambda^2] \neq 0$ for all $\lambda \in \mathbb{C}$. Among all possible factors $x(\xi) = 1, \xi, \xi^2, \xi^3$ only $x(\xi) = 1$ qualifies, so we conclude that the parametrization of all C's is given by $C(\xi) = u[\xi^4 \quad \xi^3]$ with $0 \neq u \in \mathbb{R}$.

Example 32 : Let $\mathcal{P}_{\text{full}}$ be represented by

$$\begin{aligned} w_1 + \dot{c} + c &= 0 \\ \ddot{w}_2 + \dot{c} + c &= 0. \end{aligned}$$

Then $(\mathcal{P}_{\text{full}})_w$ is represented by $w_1 = \ddot{w}_2$. Let $\mathcal{K}$ be represented by $w_1 = \ddot{w}_2, \ddot{w}_1 = 0$. We have

$$R_1 = \begin{bmatrix} 1 & 0 \\ 0 & \xi^2 \end{bmatrix}, R_2 = \begin{bmatrix} 1+\xi \\ 1+\xi \end{bmatrix}, \tilde{R}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, L(\xi) = 1 + \xi.$$

In $\mathcal{P}'_{\text{full}}$, represented by $R_1w + \tilde{R}_2c', = 0$, c' is observable from w. We parameterize all controllers C' that regularly implement $\mathcal{K}$ w.r.t. $\mathcal{P}'_{\text{full}}$. Performing the steps of theorem 4.3, we obtain V_2 void, $V_1 = I_2$, V_2R_2 void, $M = 1$. Furthermore,

$$K = \begin{bmatrix} 1 & -\xi^2 \\ \xi^2 & 0 \end{bmatrix}, F_1 = \begin{bmatrix} 1 & -1 \\ \xi^2 & 0 \end{bmatrix}.$$

We take $Q = [1 \quad 0]$ and $W = [0 \quad 1]$. The required parametrization is then $C'(\xi) = u\xi^2$, $0 \neq u \in \mathbb{R}$. To obtain a parametrization for the original plant $\mathcal{P}_{\text{full}}$, note that $C'(\xi)L(\xi) = u\xi^2(1 + \xi)$. According to corollary 4 we should compute all nonzero factors $x(\xi)$ of this polynomial with the property that $[x(\lambda) \quad \lambda^2] \neq 0$ for all λ. Among all possible factors $1, \xi, \xi^2$ and $1 + \xi$, only 1 and $1 + \xi$ qualify. We conclude that a parametrization of all controllers C is given by: $C(\xi) = u\xi^2$, $0 \neq u \in \mathbb{R}$ or $C(\xi) = u\xi^2(1 + \xi)$, $0 \neq u \in \mathbb{R}$.

Example 33 : Consider the full plant behavior $\mathcal{P}_{\text{full}}$ represented by

$$\begin{aligned} w_1 + \dot{w}_2 + \dot{c}_1 + c_2 &= 0 \\ w_2 + c_1 + c_2 &= 0 \\ \dot{c}_1 + c_1 + \dot{c}_2 + c_2 &= 0 \end{aligned}$$

We will parameterize all controllers $C(\frac{\mathrm{d}}{\mathrm{d}t})c = 0$ that stabilize $\mathcal{P}_{\text{full}}$ through c. We have

$$R_1 = \begin{bmatrix} 1 & \xi \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, R_2 = \begin{bmatrix} \xi & 1 \\ 1 & 1 \\ \xi+1 & \xi+1 \end{bmatrix}, \tilde{R}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & \xi+1 \end{bmatrix}, L = \begin{bmatrix} \xi & 1 \\ 1 & 1 \end{bmatrix}.$$

In $\mathcal{P}'_{\text{full}}$, represented by $R_1 w + \tilde{R}_2 c', = 0$, c' is observable from w. We first parameterize all controllers $C'(\frac{\mathrm{d}}{\mathrm{d}t})c' = 0$ that stabilize $\mathcal{P}'_{\text{full}}$. Performing the steps of corollary 5, we obtain $V_2 = [0 \quad 0 \quad 1]$, $V_2\tilde{R}_2 = TS$ with $T(\xi) = \xi + 1$ and $S = [0 \quad 1]$ Choose $C_0 = [1 \quad 0]$. The required parametrization is then $C'(\xi) = [d(\xi) \quad f(\xi)]$ with d an arbitrary Hurwitz polynomial, and f an arbitrary polynomial. We compute $C'(\xi)L(\xi) = [\xi d(\xi)+f(\xi) \quad d(\xi)+ f(\xi)]$. A parametrization for the original plant $\mathcal{P}_{\text{full}}$ is obtained by computing, for any choice of d and f, all nonzero common factors $x(\xi)$ of the polynomials $\xi d(\xi)+f(\xi)$ and $d(\xi)+f(\xi)$ with the property that $[x(\lambda) \quad d(\lambda) \quad f(\lambda)] \neq 0$ for all λ. Let d and f be given, d Hurwitz. We distinguish the following cases:

1. $x(\xi) = c$, constant, unequal to zero. These $x(\xi)$'s satisfy the requirements
2. $x(\xi)$ has at least one zero $\lambda \neq 1$. Then $\lambda d(\lambda) + f(\lambda) = 0$ and $d(\lambda) + f(\lambda) = 0$. If $d(\lambda) = 0$ then also $f(\lambda) = 0$, and this leads to $[x(\lambda) \quad d(\lambda) \quad f(\lambda)] = 0$, violating the rank condition. If $d(\lambda) \neq 0$ then a simple calculation shows that $\lambda = 1$, which contradicts the assumption that $\lambda \neq 1$. Thus this case does not yield required $x(\xi)$'s.
3. $x(\xi)$ has only $\lambda = 1$ as zero, in other words, $x(\xi) = c(\xi - 1)^k$ for some $c \neq 0$ and integer $k \geq 1$. In this case we distinguish further between the following cases:
   (a) $k = 1$. We have $d(1) + f(1) = 0$. Since d is Hurwitz, $d(1) \neq 0$, so we have $[x(1) \quad d(1) \quad f(1)] \neq 0$, and the rank condition holds. We conclude that $x(\xi) = c(\xi - 1)$, with $c \neq 0$, satisfies the requirements.
   (b) $k > 1$. In this case $\lambda = 1$ is also a common zero of the derivative polynomials $d(\xi) + \xi d'(\xi) + f'(\xi)$ and $d'(\xi) + f'(\xi)$. This implies $d(1) = 0$, which contradicts the fact that d is Hurwitz. We conclude that $x(\xi) = c(\xi - 1)^k$ for $k > 2$ does not satisfy the requirements.

Our conclusion is that a parametrization of all stabilizing controllers for $\mathcal{P}_{\text{full}}$ is given by: $C(\xi) = [\xi d(\xi) + f(\xi) \quad d(\xi) + f(\xi)]$, d Hurwitz polynomial, f arbitrary polynomial, or $C(\xi) = \frac{1}{\xi-1}[\xi d(\xi) + f(\xi) \quad d(\xi) + f(\xi)]$, d Hurwitz polynomial and f polynomial such that $d(1) + f(1) = 0$.

4.7 Summary

In this chapter we have considered the complementary part of chapter 3. We proceeded in the same way as there. Pole placement and stabilization problems were solved first. Then we provided the whole family of regularly implementing and stabilizing controllers for the partial interconnection case.

Chapter 5

Embedding Algorithms

Embedding, i.e expanding a nonsquare matrix to a square one, plays a significant role in systems and control theory. For instance, embedding a polynomial matrix $P(\xi)$ *unimodularly* allowed us to solve the pole placement problem in chapter 3 and *stable embedding* (a less restrictive version) yielded a solution for the stabilization problem.
In this chapter we will discuss existing and new algorithms[1] to solve the embedding problem. The scope of the chapter is as follows. In section 5.1 we define the problem and give some motivating examples, in section 5.2 we give an overview of existing algorithms to solve the embedding problem, and we give a couple of examples to show that the method based on the Smith form does not give acceptable answers. After that we turn to the algorithm of Beelen [4],[5] based on matrix pencils. In section 5.3 we introduce matrix pencils, in section 5.4 we relate the embedding problem for an arbitrary polynomial matrix to one for a pencil by using a state space representation. In section 5.5 we give the solution for the embedding problem for a pencil, section 5.6 describes the way to calculate that solution, in section 5.7 we describe the algorithm that gives a unimodular or stable embedding starting from a polynomial matrix.

5.1 Problem formulation

Let $P(\xi)$ be a nonsquare polynomial matrix. Embedding $P(\xi)$ into a square matrix means to enhance $P(\xi)$ with the necessary number of rows or columns to get a square matrix $W(\xi)$. This problem in itself is of course not very interesting, it becomes interesting if we add additional conditions to the resulting $W(\xi)$. In this chapter we will look specifically at two forms of this problem, the unimodular and the stable embedding problem, unimodular embedding problem defined as follows:
Given $P(\xi)$, the *unimodular (stable)* embedding problem consists in finding a (polynomial)

[1] As we know, an *algorithm* is a sequence of commands or actions to accomplish some task after a finite number of steps. The name was taken from the very famous Muslim mathematician and astronomer Abu Ja'far Muhammad ibn Musa Al-Khwarizmi (born nearby of Baghdad, ca. 780- ca. 850 AD). The words "algebra" and "guarism" (the Arabic sign which represents a number) were taken from him as well.

R.Z. Yoe: *Modelling and Control of Dynamical Systems: Numerical Implementation in a Behavioral Framework*, Studies in Computational Intelligence (SCI) **124**, 95–113 (2008)
www.springerlink.com

matrix $Q(\xi)$ such that the resulting stacked matrix

$$W(\xi) = \begin{bmatrix} P(\xi) \\ Q(\xi) \end{bmatrix} \tag{5.1}$$

is unimodular (Hurwitz).
Recall that a polynomial matrix is unimodular if it is square and its determinant is a nonzero constant, and Hurwitz if it is square and its determinant has no zeros in the closed right half plane.

Physically speaking, we can interpret this as follows. Let us consider a given electrical, mechanical, thermal,etc. (in general a *physical*) one -port with generalized impedance Z_1 (which can be represented by a polynomial matrix $P(\xi)$), and suppose we want to find a second generalized impedance Z_2 (represented by another polynomial matrix $Q(\xi)$), such that the equivalent generalized parallel impedance Z_{eq} (seen from this equivalent one -port) allows no activity (unimodular embedding problem), or allows only declining activity (stable embedding problem). Then, such an equivalent generalized impedance is represented by the enhanced matrix $[P(\xi); Q(\xi)]$. An additional requirement could be that the impedance Z_2 (the controller) has to be constructed in such a way that the *internal* structure of Z_1 has to be taken into account (see chapter 2)
Consider the following example. Let us assume we have the following SISO[2] electrical system consisting of one resistor and one inductor. Such a network is modelled by the equation

$$v_1 + L\frac{di_1}{dt} + R_1 i_1 = 0$$

where v_1 is the exciting voltage (in volts), i_1 is the instant current (output in Amperes), R_1 is the value of the resistor (measured in Ohms) and L is the inductance of the inductor (in Henrys). We can write the latter model in matrix form as follows:

$$\begin{bmatrix} 1 & L\frac{d}{dt} + R_1 \end{bmatrix} \begin{bmatrix} v_1 \\ i_1 \end{bmatrix} = 0, \text{ i.e., } P(\frac{d}{dt})w = 0$$

Unimodular embedding means to enhance $P(\xi)$ with the necessary number of rows in such a way that we get a square matrix $W(\xi)$ with non zero constant determinant. When this is the case, we have solved *the unimodular embedding problem* . By inspection, we might propose an embedding of $P(\xi)$ looking like $Q(\xi) = [1 \quad L\xi]$. Then, we would have the following result:

$$W(\xi) = \begin{bmatrix} 1 & L\xi + R_1 \\ 1 & L\xi \end{bmatrix},$$

[2]Single input, single output

which is unimodular: $\det W(\xi) = -R_1$.
According to the example given above, it would mean that we have to generate another network (in terms of the unique existing one) such that the parallel connection has only the zero solution.
If we have to force that $\det(W(\xi))$ has only stable roots, we refer to that as the *stable embedding problem* . The collection of rows determined will define a matrix called $Q(\xi)$, which will be *the embedding* of $P(\xi)$ or equivalently said, $P(\xi)$ will be embedded by $Q(\xi)$. In this case, a 1×2 matrix $Q(\xi)$ will do the work.
If we change the example above by adding a factor $\xi + 1$ on the left hand side:

$$(\xi + 1) \begin{bmatrix} 1 & L\xi + R_1 \end{bmatrix} \begin{bmatrix} v_1 \\ i_1 \end{bmatrix} = 0, \text{ i.e., } P(\xi)w = 0$$

then we can never solve the unimodular embedding problem, since the first row of the resulting matrix W would be zero for $\xi = -1$. However, we can solve the stable embedding problem, i.e. finding a $W(\xi)$ which is Hurwitz. The same Q would yield then $\det(W(\xi)) = -R_1(\xi+1)$, but we could also take $Q = [-\xi \;\; 1]$, since then $\det(W(\xi)) = L\xi^2+R_1\xi+1$,which has only roots in the left half plane.

5.2 Preliminaries

Since the behavioral point of view is a polynomial based approach, embedding polynomial matrices means that physically, we are connecting a given plant, represented by the polynomial matrix $P(\xi)$, to some suitable controller, represented by $C(\xi)$. Hence, every time we embed a polynomial matrix, we actually connect the corresponding plant to some controller in order to implement a desired behavior, for instance a stable one, or one in which the poles are at certain desired locations.

5.2.1 Historical overview

In spite of the theoretic research some people have done on the unimodular embedding problem just described above ([20], [2], [4]), it is amazing to find out that little attention has been paid to the numerical implementation of the solutions ([4], [95]).
In fact the embedding problem is readily solved using the Smith form: Write $P = U[D \quad 0]V$, with, which U and V unimodular of appropriate size and D diagonal. Note the problem is only solvable if $P(\lambda)$ has full row rank for all $\lambda \in \mathbb{C}$. In that case we can take $D = I$, and then $Q = [0 \quad I]V$ leads to

$$W = \begin{bmatrix} U & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} D & 0 \\ 0 & I \end{bmatrix} V,$$

so Q solves the unimodular embedding problem. Unfortunately the numerical implementation of this solution is not satisfying. If for instance we try to do this in MAPLE we get

wrong answers. Constructing the Smith form is based on elementary column and row operations. These transformations are numerically unstable as we can see from the following example.

Example 34 Let $P(\xi)$ be given as

$$P(\xi) = \begin{bmatrix} \alpha & \xi \\ \xi & \xi \end{bmatrix}$$

where $\alpha \approx 0$ but $\alpha \neq 0$. The corresponding Smith form can be obtained via the row transformation as it is indicated below

$$\begin{bmatrix} 1 & 0 \\ -\xi/\alpha & 1 \end{bmatrix} \begin{bmatrix} \alpha & \xi \\ \xi & \xi \end{bmatrix} = \begin{bmatrix} \alpha & \xi \\ 0 & -\xi^2/\alpha + \xi \end{bmatrix}$$

As we see, the entries of the transforming matrix diverge for small values of α. Consequently, the numerical errors will not converge in this case. This example is small and something could be done in order to deal with small values of α. Nevertheless, for bigger matrices, the accumulation of such effects renders the reduction to Smith form numerically unstable.

Next, we shall try to embed a bigger matrix by means of the Smith form, by using the command `smith` in MAPLE.

Example 35

$$P(\xi) = \begin{bmatrix} 1 + 11\,\xi & 2 + 9.5\,\xi & 3 + 3\,\xi \\ 2.5 + 1.4\,\xi & 1.7 + 3\,\xi & 7.6 + 2.7\,\xi \end{bmatrix}$$

The corresponding Smith form will be

$$S(\xi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} = U(\xi)P(\xi)V(\xi)$$

where

$$U(\xi) = \begin{bmatrix} -0.0536 & 0.4214 \\ -2.5 - 1.4\,\xi & 1.0 + 11.0\,\xi \end{bmatrix}$$

$$V(\xi) = \begin{bmatrix} 1.0 & 0.2597 + 0.06907\,\xi & 10.1 + 63.5\,\xi + 16.65\,\xi^2 \\ 0.0 & -0.3037 - 0.10579\,\xi & -0.1 - 74.6\,\xi - 25.5\,\xi^2 \\ 0.0 & -0.0245 + 0.08172\,\xi & -3.3 - 4.85\,\xi + 19.7\,\xi^2 \end{bmatrix}$$

Matrices U and V are unimodular but let us take a look at how unimodular they are numerically speaking. Obtaining determinants we see that

$$\det(U(\xi)) = 1.0 + 0.0000000001\,\xi$$

and

$$\det(V(\xi)) = 1.0 + 0.000000001\,\xi - 0.000000002\,\xi^2,$$

So $\det(U) = 1+O(10^{-10})\xi$ and the zeros of $\det(V)$ are $\lambda_{1,2} = \{-22360.42978, 22360.92978\}$. Partitioning $V^{-1} = [V_1; V_2]$ where V_1 is $m \times n$ and V_2 is $n-m \times n$, we complete the "missing" rows of $P(\xi)$ with V_2, i.e., $W(\xi) = [P; V_2]$ which is given by

$$W(\xi) = \begin{bmatrix} 1.0 + 11.0\,\xi & 2.0 + 9.5\,\xi & 3.0 + 3.0\,\xi \\ 2.5 + 1.4\,\xi & 1.7 + 3.0\,\xi & 7.6 + 2.7\,\xi \\ 0.0 & 0.0245 - 0.0817\,\xi & -0.3038 - 0.1058\,\xi \end{bmatrix}$$

Computing $\det(W(\xi))$ we obtain

$$\det(W(\xi)) = 1.0 - 2(10^{-9})\,\xi - 10^{-9}\,\xi^2 + 10^{-9}\,\xi^3$$

In fact $\det(W) = 0$ for

$$\lambda_{1,2} = 500.3337221 \pm 866.0247300j,\ \lambda_3 = -999.6674442$$

We see that the command `smith` in MAPLE does give the Smith form and $\det(W) = 1$ if we neglect $O(10^{-9})$ terms. Nevertheless, we examine now an example where the latter does not happen in line with the observations of Van Dooren [78].

Example 36 Next let us consider the following polynomial matrix of degree $d = 4$.

$$P(\xi) = \begin{bmatrix} 0.96 + 6700\xi + 14\xi^2 + 20\xi^3 + \xi^4 & 0.0 & 0.054 + 50\xi + 0.1\xi^2 & -0.033 - 50\xi \\ 0.0 & 0.96 + 6700\xi + 14\xi^2 + 20\xi^3 + \xi^4 & -0.033 - 50\xi & -0.038 - 150\xi - 0.1\xi^2 \end{bmatrix}$$

In this case

$$S(\xi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = U(\xi)P(\xi)V(\xi)$$

Embedding $P(\xi)$ as we did in example 35 produces the following determinant

$$\det(W(\xi)) = 1.0 - 0.01\,\xi^2 + 10\,\xi^3 - 0.1\,\xi^4 - 0.01\,\xi^5$$

which has stable and unstable roots

$\lambda_1 = 27.01512230, \lambda_{2,3} = 0.2320179368 \pm 0.4025966714j, \lambda_4 = -0.4631458693, \lambda_5 = -37.016011230.$

So, the Smith form does not give in general- a unimodular W.

The behavioral approach concerns modelling and control of physical systems and therefore it is important to have reliable numerical procedures for our algorithms. As far as we know there does not exist a single work which studies the numerical properties of the embedding problem. Even serious computing libraries[3] do not have a tool to solve the problem.

As a next try we turned to the algorithm explained in the work of Beelen[4]. We will explain the method in detail in the last part of this chapter. Unfortunately this method also turned out to be unreliable as we will show in chapter 6.

Fortunately in the end we succeeded in finding a simple, elegant algorithm that behaves very well numerically. We will describe it the final chapter 7. The current chapter will be concerned with the algorithm as described by Beelen. We think that understanding the reason why this algorithm does not yield satisfactory answers is important because there are algorithms for other polynomial problems that are currently solved by the same linearization procedure. These might suffer from similar numerical problems. The procedure is for instance applied in the calculation of row reduced matrices, in coprime factorization and in the calculation of Kronecker indices.

In the work of Beelen [4] only the unimodular embedding problem is treated. We will extend his results to the stable embedding problem in this chapter. For completeness we also discuss the unimodular embedding problem in detail.

The unimodular embedding problem has solutions in a small set: we require at the end that W is unimodular, and unimodular matrices form an algebraic strict subset of all square matrices. Hurwitz matrices on the other hand are not that special, they form an open set. Therefore we expect that the stable embedding problem might be more robust than the unimodular embedding problem.

5.2.2 Notation

For notational convenience we denote the class of non constant polynomial matrices $P(\xi)$, such that $P(\lambda)$ has full row rank for all $\lambda \in \mathbb{C}$, by $\mathcal{U}$, and those that have full row rank for all λ in the closed right half plane by $\mathcal{M}$. A *unimodular* matrix is a square matrix in $\mathcal{U}$, a *Hurwitz* matrix is a square matrix in $\mathcal{M}$.

In this book a special role is played by polynomial matrices of degree one. In line with usual terminology we call this a (matrix) *pencil* , and we will denote such polynomial matrices by $\xi E - A$.

[3] LAPACK, BLAS , SLICOT. Actually, the SLICOT library is built on the LAPACK and BLAS libraries, containing a collection of highly performant and numerically reliable basic linear algebra subroutines [72].

5.3 Pencils and Matrix Pencils

We can obtain an idea of the meaning of *pencil* if we recall the basic fact that through a given single point an infinite number of straight lines can pass. Geometrically speaking they form a "bundle" of lines passing through that point. A matrix polynomial of degree 1, $\xi E - A$, describes analogously a bundle of lines through the "point" (E, A). In fact in this way we can build a pencil of many mathematical structures and a pencil of matrices is not an exception. Matrix pencils arise in the study of linear continuous and discrete time invariant state space systems and descriptor systems. In the approach of Beelen, matrix pencils are used as linearizations of arbitrary matrix polynomials. Now we are dealing with a matrix polynomial of degree one, instead of handling a matrix polynomial of degree d.

5.3.1 Canonical forms of pencils

We will define canonical forms for square and nonsquare pencils. We recall that, to a square pencil $\xi I - A$, we can associate its corresponding Jordan canonical form. For a more general (not necessarily square) pencil $\xi E - A$ there exists the so called *Kronecker canonical form.* Kronecker showed that any pencil can be expressed as a canonical block diagonal matrix (see Gantmacher [25]). If we restrict both pencils to the case that $E = I$, this canonical form yields the Jordan canonical form. This canonical form is quite appreciated from a theoretical point of view because it displays the complete eigenstructure of the pencil, i.e., many properties of $\xi E - A$ can be derived from the block entries of this structure. From a numerical point of view, however, canonical forms have been shown to be impractical to implement [77]. A more suitable way to deal with this problem had to be found. An example of an equivalent form is the so called *generalized Schur form* [5]. This form is defined in the following section.

5.3.2 A little bit deeper into matrix pencils

It is also possible to define a generalized Schur form for matrix pencils. Let us consider the pencil $\xi E - A$ where A and E are arbitrary constant matrices of equal dimensions. We say that a pencil is *regular* if

$$\det(\xi E - A) \neq 0$$

The pencil is called *singular* if either the latter is not satisfied or if $\xi E - A$ is not square. A *strictly equivalent pencil* is obtained if we apply constant, invertible matrices U, V such that

$$U(\xi E - A)V = \xi E_1 - A_1$$

which implies the equivalence (denoted by "$\sim$") of two pencils:

$$\xi E - A \sim \xi E_1 - A_1$$

If U, V are unitary matrices, the pencils are said to be *unitarily equivalent*. Taking this into account, we can define the *generalized Schur form* as a unitarily equivalent pencil given by

$$\xi E - A \sim \left[\begin{array}{c|c|c|c} \xi E_\epsilon - A_\epsilon & * & * & * \\ \hline & \xi E_\infty - A_\infty & * & * \\ \hline & & \xi E_f - A_f & * \\ \hline & & & \xi E_\eta - A_\eta \end{array}\right]$$

where
a) $\xi E_f - A_f$ is a square regular pencil containing the finite elementary divisors of $\xi E - A$ (E_f is invertible).
b) $\xi E_\infty - A_\infty$ is a square regular pencil containing the infinite elementary divisors of $\xi E - A$ (E_∞ is nilpotent, A_∞ is invertible)
c) $\xi E_\eta - A_\eta$ and $\xi E_\epsilon - A_\epsilon$ are singular pencils containing the Kronecker row and column structure, respectively.

Without loss of generality, we will consider a simpler version of the structure shown above:

$$U(\xi E - A)V = \left[\begin{array}{c|c} \xi E_{\epsilon\infty} - A_{\epsilon\infty} & * \\ \hline 0 & \xi E_{f\eta} - A_{f\eta} \end{array}\right]$$

In this case the diagonal block $\xi E_{\epsilon\infty} - A_{\epsilon\infty}$ is in so called staircase form that completely reveals the structure elements of Kronecker canonical form. More precisely, the inner structure of the block entries looks as follows

$$\left[\begin{array}{c|c} \xi E_{\epsilon\infty} - A_{\epsilon\infty} & * \\ \hline 0 & \xi E_{f\eta} - A_{f\eta} \end{array}\right]$$

$$= \left[\begin{array}{c|c|c|c|c} -A_{11} & \xi E_{12} - A_{12} & \dots & & \xi E_{1\ell+1} - A_{1\ell+1} \\ \hline 0 & -A_{22} & \dots & & \xi E_{2\ell+1} - A_{2\ell+1} \\ \hline \vdots & \vdots & \ddots & \vdots & \vdots \\ \hline 0 & 0 & \dots & -A_{\ell\ell} & \xi E_{\ell\ell+1} - A_{\ell\ell+1} \\ \hline 0 & 0 & \dots & 0 & \xi E_{\ell+1\ell+1} - A_{\ell+1\ell+1} \end{array}\right]$$

where $\xi E_{\ell+1\ell+1} - A_{\ell+1\ell+1} := \xi E_{f\eta} - A_{f\eta}$ and

1) $E_{\ell+1\ell+1}$ has full row rank
2) All the A_{ii} have full row rank ν_i $(i = 1, \dots, \ell)$
3) The E_{i-1i} have full column rank μ_i, $(i = 2, \dots, \ell)$
4) The following block has full row rank and is in echelon form

$$\left[\frac{E_{\ell\ell+1}}{E_{\ell+1\ell+1}}\right]$$

We will be exploiting the above mentioned generalized Schur form of the pencil, $\xi E - A$, in both embedding problems that we consider. But first we will explain the way pencils come in.

5.4 The state space representation

Any behavior, defined by a kernel representation, can also be represented by an observable state space representation ([60, 64]). Let the behavior be given by $\mathfrak{B} = \{w \in \mathcal{L}_1^{\text{loc}}(\mathbb{R}, \mathbb{R}^n | P(\frac{d}{dt})w = 0\}$, where $P(\xi) = P_0 + P_1\xi + P_2\xi^2 + \ldots + P_d\xi^d$ is a matrix polynomial of size $m \times n$ (we consider solutions to the differential equations in distributional sense in the space of locally integrable functions). Let the truncation of P, $T(P)$ be defined by $T(P) = [-P_d\xi; -P_d\xi^2 - P_{d-1}\xi; \ldots; -P_d\xi^{d-1} - \ldots - P_2\xi]$.

Let $\mathfrak{B}_s = \{(x, w) \mid (E\frac{d}{dt} - A)x = 0, w = Cx\}$ where the $dm \times (d-1)m + n$ matrices A and E are defined by

$$A = \begin{bmatrix} I_m & & & \\ & \ddots & & \\ & & I_m & \\ & & & P_0 \end{bmatrix}, \quad E = \begin{bmatrix} 0 & & & -P_d \\ I_m & \ddots & & -P_{d-1} \\ & \ddots & \ddots & \vdots \\ & & I_m & -P_1 \end{bmatrix}$$

and $C = [0 \quad 0 \quad \ldots \quad 0 \quad I_n]$. $\mathfrak{B}_s$ is an observable state space representation for $\mathfrak{B}$, in fact $(x, w) \in \mathfrak{B}_s$ if and only if $x = (T(P)(\frac{d}{dt})w; w)$ and $P(\frac{d}{dt})w = 0$, as can be seen by premultiplying $\xi E - A$ with the unimodular matrix

$$U(\xi) = -\begin{bmatrix} I & 0 & \ldots & 0 \\ \xi I & I & \ldots & 0 \\ \vdots & & \ddots & \\ \xi^{d-1}I & \ldots & \ldots & I \end{bmatrix} :$$

$(x, w) \in \mathfrak{B}_s$ if and only if

$$\begin{bmatrix} I & -T(P)(\frac{d}{dt}) \\ 0 & P(\frac{d}{dt}) \end{bmatrix} x = 0$$

and $w = Cx$, so if and only if $x = [T(P)(\frac{d}{dt})w; w]$, with $w \in \mathfrak{B}$. Therefore $\mathfrak{B}_s$ is a latent variable representation of $\mathfrak{B}$, and since the equations (in the original formulation) are of degree one in x and of degree zero in w, it is a state space representation.
This also shows that $\mathfrak{B}$ is controllable (stabilizable) if and only if $\mathfrak{B}_s$ is. In fact, the rank of $\lambda E - A$ equals the rank of $P(\lambda) + (d-1)m$, so in particular we have

Corollary 8 *$P(\xi) \in \mathcal{U}(\mathcal{M})$ if and only if $\xi E - A \in \mathcal{U}(\mathcal{M})$.*

We will exploit this relation with the state representation to solve the problem. Suppose we find a $K(\xi)$ such that $[\xi E - A; K(\xi)]$ is unimodular (or Hurwitz). This amounts to adding an equation $K(\frac{d}{dt})x = 0$ to the state space representation $(E\frac{d}{dt} - A)x = 0$. Now consider the behavior $\mathfrak{B}_a$ determined by these two equations:

$$\mathfrak{B}_a = \{(x, w) \mid \begin{bmatrix} E\frac{d}{dt} - A \\ K(\frac{d}{dt}) \end{bmatrix} x = 0\}$$

Multiplying the equation by the unimodular matrix $\mathrm{diag}(U, I_{n-m})$ we see that we can also represent $\mathfrak{B}_a$ by

$$\begin{bmatrix} I & -T(P)(\frac{d}{dt}) \\ 0 & P(\frac{d}{dt}) \\ K_1(\frac{d}{dt}) & K_2(\frac{d}{dt}) \end{bmatrix} x = 0$$

where we split K conformally. Finally premultiplying by

$$\begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ -K_1 & 0 & I \end{bmatrix}$$

we find the equivalent representation

$$\begin{bmatrix} I & -T(P)(\frac{d}{dt}) \\ 0 & P(\frac{d}{dt}) \\ 0 & K_2(\frac{d}{dt}) + K_1 T(P)(\frac{d}{dt}) \end{bmatrix} x = 0.$$

Taking

$$Q(\xi) = K_2(\xi) + K_1(\xi)T(P)(\xi) \tag{5.2}$$

and noting that

$$\begin{bmatrix} I & -T(P)(\xi) \\ 0 & P(\xi) \\ 0 & Q(\xi) \end{bmatrix}$$

is still unimodular (Hurwitz), we see that in fact $[P(\xi); Q(\xi)]$ is unimodular (Hurwitz).

5.5 Embedding for a pencil

If the pencil $\xi E - A$ has a special form: upper block diagonal with full row rank constant and unimodular (or Hurwitz) matrices on the diagonal, then finding K is easy.

Lemma 9 *Let the pencil $\xi\widehat{E} - \widehat{A}$ be given by*

$$\left[\begin{array}{c|c|c|c|c} -A_{11} & \xi E_{12} - A_{12} & \dots & & \xi E_{1\ell+1} - A_{1\ell+1} \\ \hline 0 & -A_{22} & \dots & & \xi E_{2\ell+1} - A_{2\ell+1} \\ \hline \vdots & \vdots & \ddots & \vdots & \vdots \\ \hline 0 & 0 & \dots & -A_{\ell\ell} & \xi E_{\ell\ell+1} - A_{\ell\ell+1} \\ \hline 0 & 0 & \dots & 0 & \xi E_{\ell+1\ell+1} - A_{\ell+1\ell+1} \end{array}\right] \tag{5.3}$$

such that each A_{ii} has full row rank for $i = 1 \dots \ell$ and $\xi E_{\ell+1\ell+1} - A_{\ell+1\ell+1}$ is unimodular (Hurwitz). Let $\hat{K} = \operatorname{diag}(K_{11}(\xi), \dots, K_{\ell\ell}(\xi),\ 0)$, be such that each block $[A_{ii}; K_{ii}(\xi)]$ is unimodular (Hurwitz). Then $[\xi\widehat{E} - \widehat{A}; \hat{K}]$ is unimodular (Hurwitz)

Proof. The proof is straightforward. Using row permutations we bring the resulting pencil in an upper block triangular form with the blocks $[-A_{ii}; K_{ii}(\xi)]$ and $\xi E_{\ell+1\ell+1} - A_{\ell+1\ell+1}$ on the diagonal, proving that the determinant of the pencil is a nonzero constant (has its roots in the open left half plane).

The idea of the construction is to find orthogonal matrices M and N such that $M(\xi E - A)N$ has the structure displayed in (5.3). Then $\hat{K}(\xi)N^T$ is the polynomial matrix that we are looking for:

Lemma 10 *Let $\xi E - A$ be an arbitrary matrix pencil in $\mathcal{U}$ $(\mathcal{M})$, and let M, N be orthogonal, constant matrices such that $M(\xi E - A)N$ has the structure (5.3) of lemma 9. Then, there exists a polynomial matrix $K(\xi)$ such that $[\xi E - A; K(\xi)]$ is unimodular (Hurwitz).*

Proof. Choose $\hat{K}(\xi)$ such that $[M(\xi E - A)N; \hat{K}]$ is unimodular (Hurwitz). Since

$$\begin{bmatrix} \xi E - A \\ \hat{K}(\xi)N^T \end{bmatrix} = \begin{bmatrix} M^T & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} M(\xi E - A)N \\ \hat{K}(\xi)) \end{bmatrix} N^T,$$

it is also unimodular (Hurwitz). So we can take $K(\xi) = \hat{K}(\xi)N^T$.

So the remaining task is to find for an arbitrary pencil in $\mathcal{U}$ $(\mathcal{M})$ the matrices M and N.

5.6 Transforming the pencil

The next theorem proves that we can find M and N.

Theorem 16 *Let $\xi E - A$ be a pencil in $\mathcal{U}(\mathcal{M})$. Then there exist orthogonal matrices M and N such that $M(\xi E - A)N =$*

$$\left[\begin{array}{c|c|c|c|c} -A_{11} & \xi E_{12} - A_{12} & \dots & & \xi E_{1\ell+1} - A_{1\ell+1} \\ \hline 0 & -A_{22} & \dots & & \xi E_{2\ell+1} - A_{2\ell+1} \\ \hline \vdots & \ddots & \ddots & \vdots & \vdots \\ \hline 0 & \dots & 0 & -A_{\ell\ell} & \xi E_{\ell\ell+1} - A_{\ell\ell+1} \\ \hline 0 & & \dots & 0 & \xi E_{\ell+1\ell+1} - A_{\ell+1\ell+1} \end{array}\right]$$

where A_{ii} has full row rank for $i = 1 \dots \ell$, and the pencil $\xi E_{\ell+1\ell+1} - A_{\ell+1\ell+1}$ is either unimodular (Hurwitz) or void.

Remark 37 *By void we mean that it has size 0×0, so then the whole last block column is missing.*

Proof. The theorem is proved by induction on the size s of the pencil, the sum of the number of rows and columns.
For $s = 2$ the pencil is scalar, and being in $\mathcal{M}$ it has to be a nonzero constant, so $\ell = 1$, and the lower right pencil is missing (or $\ell = 0$, and the lower right pencil is constant).
Now suppose that $s > 2$. Let the pencil have m rows and n columns (because it is in $\mathcal{M}$ we have that $m \leq n$). If $m = n$ then the pencil is Hurwitz and it has the desired form already with $\ell = 0$, so assume that $m > n$.
Let N_1 be an orthogonal matrix such that $EN_1 = [0\ E_2]$, where E_2 has full column rank. Decompose AN_1 in the same way: $AN_1 = [A_1\ A_2]$.
If $A_1 = 0$, then $\xi E_2 - A_2$ has to be Hurwitz, so we are finished with $N_1 = N, M = I$, and $\ell = 0$.
If $A_1 \neq 0$, then let M_1 be an orthogonal matrix such that $M_1 A_1 = [A_{11};\ 0]$, with A_{11} having full row rank. Decompose the products of M_1 with the other blocks likewise: $M_1 A_2 =: [A_{12};\ A_{22}]$, $M_1 E_2 =: [E_{12};\ E_{22}]$.
The lower right pencil $\xi E_{22} - A_{22}$ is in $\mathcal{M}$ and has size smaller than s, so by induction there exist M_2, N_2 such that $M_2(\xi E_{22} - A_{22})N_2$ has the desired structure.
Let $M = (\mathrm{diag}(I, M_2))M_1$, $N = N_1(\mathrm{diag}(I, N_2))$, then

$$\begin{aligned} & M(\xi E - A)N \\ = & \begin{bmatrix} I & 0 \\ 0 & M_2 \end{bmatrix} \begin{bmatrix} A_{11} & \xi E_{12} - A_{12} \\ 0 & \xi E_{22} - A_{22} \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & N_2 \end{bmatrix} \\ = & \begin{bmatrix} A_{11} & \xi E_{12} - A_{12} N_2 \\ 0 & M_2(\xi E_{22} - A_{22})N_2 \end{bmatrix} \end{aligned}$$

which proves the theorem.

5.7 The algorithm

Summing up: starting with a polynomial matrix $P(\xi) \in \mathcal{U}(\mathcal{M})$ we construct the associated pencil $\xi E - A \in \mathcal{U}(\mathcal{M})$. We find M and N such that $M(\xi E - A)N$ has the structure of (5.3), and construct a $\hat{K}(\xi)$ which embeds this structured pencil into a unimodular (Hurwitz) pencil. $K = \hat{K}N^T$ then embeds the original $\xi E - A$. Decomposing $K = [K_1\ K_2]$, then yields $Q(\xi)$ as $K_1(\xi)T(P)(\xi) + K_2(\xi)$.
Note that the associated pencil and $T(P)$ are defined directly in terms of the original matrix $P(\xi)$, so the only elements that we really compute are the matrices $K_i(\xi)$. Starting from the associated pencil this is accomplished by QR decompositions alone.
In this section we will first discuss the orthogonal decompositions that we will use in the actual implementation of the algorithm, then we will concentrate on the transformation of the original pencil into staircase form, and finally we will describe the implementation of the complete algorithm.

5.7.1 QR Decompositions

In order to obtain the structure given by 5.3, we need to find a "good" class of row and column transformations (matrices M and N there). At this moment, and likely also in the near future, the most attractive method is still the *QR algorithm* [82]. Because of its characteristics (like convergence velocity, ease of implementation, etc) QR decompositions were chosen[4] [33], [54]. From a numerical point of view, the QR factorization is an attractive tool, used extensively [33], [27], [29]. Of course the ratio cost - profit depends on the problem at hand and on the way to compute it. It can be implemented by the use of Householder reflections, by Givens rotations or by the Gram-Schmidt method. In general, a QR factorization of $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ is a decomposition

$$A = Q^T R_A = Q^T \left[\frac{A_r}{0}\right] \tag{5.4}$$

where $Q \in \mathbb{R}^{m\times m}$ is orthogonal and $A_r \in \mathbb{R}^{r\times n}$ has full row rank. We call this structure a *row compression* of the matrix A. Since we want to construct a staircase form, we shall consider (5.4) written slightly different:

$$R_A = QA,$$

The QR factorizations yields more: A_r is not only full row rank but is in row echelon form. For numerical reasons it is advisable not to apply the decomposition to A, but first order the columns of A to their Euclidean norm. This is called *QR decomposition with column pivoting* [27].

[4] In fact, the QR algorithm belongs to a more general class known as *GR algorithms*, where "G" stands for generic or general. When G=Q (unitary or orthogonal transformation) the algorithm is called QR [82].

QR Decomposition with column pivoting

Given $A \in \mathbb{R}^{m\times n}$ it computes the following factorization

$$R_A = QAP, \quad R_A = \left[\begin{array}{c|c} R_{1,1} & R_{1,2} \\ \hline 0_{2,1} & 0_{2,2} \end{array}\right] \tag{5.5}$$

where $R_{1,1} \in \mathbb{R}^{\rho_A\times\rho_A}$ is upper triangular, $R_{1,2} \in \mathbb{R}^{\rho_A\times n-\rho_A}$, $0_{2,1} \in \mathbb{R}^{m-\rho_A\times\rho_A}$, $0_{2,2} \in \mathbb{R}^{m-\rho_A\times n-\rho_A}$, $\rho_A = rank(A)$, Q is orthogonal, and P is a permutation matrix.
Next, the corresponding pseudo code is given. It is Matlab syntax based. We remark that a line starting with "%" means a comment. The first pseudo code line means the way the algorithm is called in the real program, i.e., it is a program function which receives some data in the right hand side and produces a result, given in the left hand side. So, the meaning of the first line given below is that the function (the program) is called "$algorithm_{QR}$" and it needs a matrix A as data in order to compute R_A, P and Q. Lines of commands with no "%" indicates that something has to be computed. Finally, ";" means end of line with no need of showing what is computed.

Algorithm 7 `[Algorithm QR with column pivoting]`

```
(% indicates comment)
[R_A, Q, P] = algorithm_QR(A)
Q = I_m; P = I_n;
k=1;
while k ≤ min(m, n)
    col_j(A) := A_{k≤i≤m, k≤j≤n};
    % Determine ℓ (k ≤ ℓ ≤ n) such that
    ||col_ℓ(A)|| = max_{k≤j≤n} ||col_j(A)||;
        if ℓ − k ≠ 0 then
            Interchange col_k(A) and col_ℓ(A);
            According to the latter, update P;
            for i = m : −1 : k + 1
               % Zeroing entries A(i, k) as well as updating of Q is done here
               % Q is a Givens rotation
                A := P_{i−1,i} A;  Q := P_{i−1,i} Q;
            end
        end
k=k+1;
end
%Result:
R_A, Q, P
```

RQ Decomposition

An RQ factorization of an $m \times n$ matrix A is given by

$$A = \bar{R}_A \bar{Q}^T = \left[\begin{array}{c|c} 0 & A_c \end{array}\right] \bar{Q}^T$$

where $\bar{Q}$ is orthogonal $A_c \in \mathbb{R}^{m\times r}$ is in column echelon form, and has full column rank. Such a form is called *column compression.* As we did for the QR case we can write as well

$$\bar{R}_A = A\bar{Q}$$

In addition, as we did before, we can provide the latter with a row permutation matrix $\bar{P}$ as follows

RQ Decomposition with column pivoting

Given $A \in \mathbb{R}^{m\times n}$ it computes the following factorization

$$\bar{R}_A = \bar{P}A\bar{Q}, \quad \bar{R}_A = \left[\begin{array}{c|c} 0_{1,1} & R_{1,2} \\ \hline 0_{2,1} & R_{2,2} \end{array}\right] \tag{5.6}$$

where $0_{1,1} \in \mathbb{R}^{m-\rho_A\times n-\rho_A}$, $R_{1,2} \in \mathbb{R}^{m-\rho_A\times \rho_A}$, $0_{2,1} \in \mathbb{R}^{\rho_A\times n-\rho_A}$, $R_{2,2} \in \mathbb{R}^{\rho_A\times \rho_A}$ is upper echelon, $\rho_A = rank(A)$, $\bar{P}$ a permutation matrix and $\bar{Q}$ is an orthogonal transformation.

Algorithm 8 `[RQ decomposion with pivoting]`
`(% indicates comment)`
$[\bar{R}_A, \bar{Q}, \bar{P}] = algorithm_{RQ}(A)$
$\bar{A} = (\mathtt{rot}_{180}(A))^T;$
$[\widehat{A}, Q, P] = algorithm_{QR}(\bar{A})$
$\bar{R}_A = (\mathtt{rot}_{180}(\widehat{A}))^T;\ \bar{P} = \mathtt{rot}_{180}(P);\quad \bar{Q} = rot_{180}(Q);$
`% Result:`
$\bar{R}_A, \bar{Q}, \bar{P}$

Above, rot_{180} indicates we rotate 180 degrees clockwise the entries of a given matrix: $(\mathrm{rot}_{180}A)_{ij} = A_{m-i+1,n-j+1}$. Note that rot_{180} is its own inverse.

5.7.2 Staircase form of $\xi E - A$

The main structure of this algorithm is the implementation of the constructive proof of theorem 16. This algorithm consists of a nested collection of QR and RQ decompositions. Depending on the position of the transforming matrix (left or right multiplying our pencil $\xi E - A$) we need to apply either a QR or an RQ decomposition. Of course it is very difficult to specify all details here, but having the cited proof in mind plus the two basic algorithms described, the reader can get a good idea of the general implementation.

The starting point for each iteration step is a pencil $\xi E_j - A_j$ of size $m_j \times n_j$. We first obtain a column echelon form $[0 \mid E_{j,2}]$ for the matrix E_j of the pencil. To achieve this we have to find an RQ-decomposition of it, getting two matrices: a column transformation $\tilde{Q}_j$ and the corresponding row permutation, $\tilde{P}_j$. Since we have modified E, we have to do the same with A, getting $\tilde{P}_j A_j \tilde{Q}_j = [A_{j,1} \mid A_{j,2}]$, partitioning A_j conformally. From this partition we can immediately derive $\rho_j = \rho(E) = \rho(E_{j,2}) = \text{rank}(E_j)$. Define $\nu_j = n_j - \rho_j$. It is now the turn of the row compression of $A_{j,1}$. Let $\mu_j = \text{rank}(A_{j,1})$. As a result of such a process, we get two transformation matrices $\bar{Q}_j$ (row compression) and $\bar{P}_j$ (column permutation) for $A_{j,1}$. Let $\bar{Q}_j A_{j,1} \bar{P}_j = (A_{j,11}; 0)$. It yields the first block zeroing of the original pair (E_j, A_j) (see the proof of theorem 16).

$$\bar{Q}_j \tilde{P}_j (\xi E_j - A_j) \tilde{Q}_j \bar{P}_j = \begin{bmatrix} -A_{j,11} & \xi E_{j,21} - A_{j,21} \\ 0 & \xi E_{j,22} - A_{j,22} \end{bmatrix}$$

with $A_{j,11} \in \mathbb{R}^{\mu_j \times \nu_j}$.

Start for $j = 1$ with $\xi E_1 - A_1 = \xi E - A$, and take $\xi E_{j+1} - A_{j+1} = \xi E_{j,22} - A_{j,22}$. Define the accumulated sums $s_{j+1} = \mu_j + s(j)$, $s_1 = 0$, $t_{j+1} = \nu_j + t_j$, $t_1 = 0$. The size of the new structure $\xi E_{1,22} - A_{1,22}$ is then $m_{j+1} = m_j - t_j$ by $n_{j+1} = n_j - s_j$.
Define $M_j = \text{diag}(I_{s_j}, \bar{Q}_j \tilde{P}_j)$, $M = M_{jmax} \cdots M_1$, and $N_j = \text{diag}(I_{t_j}, \tilde{Q}_j \bar{P}_j)$, $N = N_{jmax} \cdots N_1$. Then $M(\xi E - A)N$ is in the generalized Schur form.
We remark that it was very helpful to build 3D matrices (called here sometimes *"cells"* or arrays) because we could construct the block entries of 5.3 computing each step of the algorithm within a "page" of a 3D matrix (*page* means the third dimension of our set of 3D arrays).

Algorithm 9 Block Quasi-triangular form of $\xi E - A$

`(% indicates comment)`
$[\xi \widehat{E} - \widehat{A}, M, N] = Staircase(\xi E - A)$;

`% Initializing variables:`
$j := 1$; $m_1 := m$; $n_1 := n; \rho_1 = m$; $A_{1,1,1} = A$; $E_{1,1,1} = E$;
`% Induction step for` $j \geq 1$
`while` $\rho_j \geq 1$
`%RQ decomposition of` $E_{1,1,j}$ `yielding its column echelon form`
$[Ep_{(1,2,j)}, M_j, N_j] =$ ***algorithm***$_{RQ}(E_{1,1,j})$;

$\rho_j = \text{rank}(Ep_{(1,2,j)}); \nu_j = n_j - \rho_j$
$Ep_{(1,1,j)} = 0_{m_j, \nu_j}$
`% We have a partition of` $M_j E_j N_j$ `as` $E_j = [0 \mid Ep_{(1,2,j)}]$
`% where` $0 \in \mathbb{R}^{m_j v \times n_j - \rho(j)}$, $Ep_{(1,2,j)} \in \mathbb{R}^{m_j \times \rho}$, $\rho = rank(E)$
$[Ap_{(1,1,j)}, Ap_{(1,2,j)}] := M_j A_{1,1,j} N_j$;

```
    if ν_j = 0
     break
    else
     % Row compression of A_p(1,1,j) to full rank μ_j while keeping E_(1,1,j) zero
     [A_c(1,1,j), Q_{c_j}, P_{c_j}] = algorithm_QR(A_p(1,1,j))
     μ_j = rank(A_c(1,1,j))
     A_c(2,1,j) = 0_{m_j − μ_j, ν_j}
     [A_c(1,2,j); A_c(2,2,j)] = Q_{c_j} A_p(1,2,j)
```

$$\left[\begin{array}{c|c} Ec_{(1,1,j)} & Ec_{(1,2,j)} \\ \hline Ec_{(2,1,j)} & Ec_{(2,2,j)} \end{array}\right] = [0 \quad | \quad Qc_j E_{p(1,2,j)}]$$

```
     % New small pencil to work with
     A_{j+1} := Ac_(2,2,j); E_{j+1} := Ec_(2,2,j)
     % M and N are the accumulated row and column
     % transformations, respectively
     ρ_{j+1} = ρ(j);  s_j := Σ_{i=1}^{j−1} ν_i;  t_j := Σ_{i=1}^{j−1} μ_i;
     M_j := diag(I_{s_j}, Q_{c_j} P_j) M_{j−1};
     N_j := N_{j−1} diag(I_{t_j}, N_j Q_{c_j});
     m_{j+1} := m_j − ν_j;  n_{j+1} := n_j − μ_j;
    end
     j := j + 1;  j_max := j;
end
Retrieve of required matrices from 3D-arrays;
% Result
ξÊ − Â
```

We recall that since our matrices are subindexed we needed to use cells (Matlab term to denote matrix storage of any dimension). This fact was an advantage because in all the steps of the algorithm, we had available all the intermidiate matrices to compute $\xi\widehat{E}-\widehat{A}$. A small example is shown below. There we have a subcell which stores part of $\widehat{A}$ (a submatrix of $\widehat{A}$) calculated at some step j of the program's running.

5.7.3 Algorithm: Embedding $P(\xi)$

At this moment every step needed to compute a Q that embeds P in a W of the required form has been covered, except the construction of K_F from lemma 9. Since the staircase algorithm (algorithm 9) finds A_{ii} in row echelon form, an obvious choice would be to take $K_F = \text{diag}K_i$ where $K_i \in \mathbb{R}^{\nu_i-\mu_i,\nu_i}$ has a single one in each row to make up for the 'missing rows' is A_{ii}:

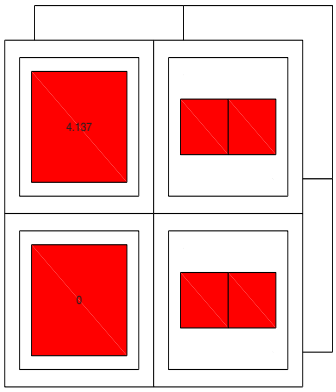

Figure 5.1: *3D storing of E_c and A_c*

Algorithm 10 `[Embedding a row echelon matrix]`
K=`Regular_embedding`(A, ν)
`%` $0 = \nu_0 < \nu_1 < \nu_2 < \ldots < \nu_m < \nu_{m+1} = n+1,\ A \in \mathbb{R}^{m,n}$
`% in upper row echelon form with` $A_{j\nu_j}$ `the first non zero entry in row` j
`% We make` K `by putting ones in the columns that are not covered by` ν
`For` $j = 1 : m+1$
$K_j = [0_{\nu_j - \nu_{j-1} - 1, \nu_{j-1}}, I_{\nu_j - \nu_{j-1} - 1}, 0_{\nu_j - \nu_{j-1} - 1, n+1-\nu_j}]$
`end`
$K = [K_1; \ldots; K_m]$

This may not be the wisest choice from a numerical point of view. We will come back to this in chapter 6.

Algorithm 11 `[Computation of` $Q(\xi)$`]`
`(% indicates comment)`
$[\xi E - A] = pencil(P(\xi))$;
$[\xi \widehat{E} - \widehat{A}, M, N] = Staircase(\xi E - A)$;
$K_F = \mathrm{diag}($`Regular_embedding`$(A_{jj}), \nu_{jj})$
`Compute` $Q(\xi)$ `with equation 5.2`
`Build the final embedding` W `given by equation 5.1`

We use an algorithm (not listed but implemented from section 5.4) called "pencil" to compute the associated pencil $\xi E - A$ from $P(\xi)$, by $[\xi E - A, R_i] =$`pencil`$(P(\xi))$.

5.8 Summary

Since the Behavioral Approach considers lumped dynamical systems as polynomial matrices (in one unknown). Such dynamical systems have to be controlled or stabilized. In this setting stabilization is equivalent to adding rows to a polynomial matrix s.t. the resulting square matrix is *Hurwitz*, while for pole placement we need to add rows in such a way that the resulting square matrix is unimodular, see chapter 3.

In this chapter we discussed two existing algorithms, one based on the Smith form, and one developed by Beelen [5]. We relaxed the conditions imposed in Beelen on the embedding process in order to provide a less restrictive and possibly more promising algorithm for the stable embedding problem. The process we followed was transferring the Hurwitz property as it is indicated here:

$$\left[\begin{array}{c} \xi E - A \\ K(\xi) \end{array}\right]_{\text{Hurwitz}} \rightarrow \left[\begin{array}{c} \xi \widehat{E} - \widehat{A} \\ \widehat{K}(\xi) \end{array}\right]_{\text{Hurwitz}} \rightarrow \left[\begin{array}{c} P(\xi) \\ Q(\xi) \end{array}\right]_{\text{Hurwitz}}$$

Hence:
• A stable polynomial matrix embedding is computed for a stabilazable one.

• This algorithm resembles the one described by [4], [5] for the unimodular embedding problem (although there, no further development or implementation, as far as we know, was done).

• Numerical problems associated with this kind of algorithms are treated in chapter 6.
• In Chapter 7 we will describe a new algorithm for embedding.

Chapter 6

Numerical Implementation

6.1 Introduction

In chapter 5 we described an algorithm to find an embedding for a given polynomial matrix. We started the chapter by showing that using the Smith form leads to unsatisfying answers. Therefore we turned to the algorithm developed by Van Dooren and Beelen [5]. In this chapter we will discuss the numerical properties of this algorithm. It turns out that the implementation that we suggested in chapter 5 does not work well. We start this chapter by giving a number of examples that show this. In the remaining part of the chapter we will investigate the properties of matrix pencils to show that the transformation from polynomial matrices to matrix pencils (see section 5.4) suffers from some inherent numerical problems.

6.2 Analysis of an example

Let us consider the following full row rank polynomial matrix ($d = 3, m = 2, n = 4$).

$$P(\xi) = \begin{bmatrix} \xi^2 - 1 & \xi^3 + 1 & \xi^3 + 11 & \xi^2 + 7 \\ \xi^3 + 2 & \xi^3 - 4 & 3\xi^3 + 3\xi + 1 & \xi^2 + \xi + 22 \\ \xi^3 + \xi + 1 & \xi^2 + \xi + 5 & \xi^3 + \xi + 9 & \xi^2 + 78 \end{bmatrix}$$

With corresponding linearization $\xi E - A$, denoted below as $\Pi(\xi)$ ($d = 1, m_p = 9, n_p = 10$).

R.Z. Yoe: *Modelling and Control of Dynamical Systems: Numerical Implementation in a Behavioral Framework*, Studies in Computational Intelligence (SCI) **124**, 115–134 (2008)
www.springerlink.com

$$\Pi(\xi) = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & -\xi & -\xi & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & -\xi & -\xi & -3\,\xi & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & -\xi & 0 & -\xi & 0 \\ \xi & 0 & 0 & -1 & 0 & 0 & -\xi & 0 & 0 & -\xi \\ 0 & \xi & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -\xi \\ 0 & 0 & \xi & 0 & 0 & -1 & 0 & -\xi & 0 & -\xi \\ 0 & 0 & 0 & \xi & 0 & 0 & 1 & -1 & -11 & -7 \\ 0 & 0 & 0 & 0 & \xi & 0 & -2 & 4 & -3\,\xi - 1 & -\xi - 22 \\ 0 & 0 & 0 & 0 & 0 & \xi & -\xi - 1 & -\xi - 5 & -\xi - 9 & -78 \end{bmatrix}$$

In order to transform the latter $\Pi(\xi)$ to the equivalent form $\widehat{\Pi} = \xi\widehat{E} - \widehat{A}$ we apply the following row/column orthogonal transformation matrices, M (which is 10×10 big) and N, which is 9×9 big, respectively:

$$M = \begin{bmatrix} 0.0123 & 0.0123 & 0.0123 & 0 & 0.0123 & -0.0000 & 0.0860 & 0.2704 & 0.9586 \\ 0.0248 & -0.0125 & 0.0248 & -0.0871 & -0.3110 & -0.9329 & -0.1372 & -0.0632 & 0.0337 \\ 0.1330 & 0.2404 & 0.7809 & 0.1108 & -0.0724 & 0.1218 & -0.5052 & -0.1473 & 0.0730 \\ -0.0979 & 0.1765 & 0.5153 & 0.0078 & 0.2061 & -0.1763 & 0.7835 & -0.0324 & -0.0714 \\ -0.1566 & 0.0999 & -0.1969 & 0.6238 & -0.0151 & -0.0195 & 0.0695 & -0.7028 & 0.1954 \\ -0.3565 & 0.1289 & -0.0270 & 0.4282 & 0.5938 & -0.2437 & -0.2628 & 0.4254 & -0.1008 \\ -0.1007 & 0.2328 & -0.1044 & -0.6068 & 0.5575 & -0.0794 & -0.1611 & -0.4429 & 0.1319 \\ -0.8109 & 0.3232 & -0.0008 & -0.1876 & -0.4247 & 0.1329 & -0.0278 & 0.0622 & -0.0033 \\ -0.3906 & -0.8517 & 0.2710 & -0.0648 & 0.1063 & 0.0011 & -0.0632 & -0.1590 & 0.0616 \end{bmatrix}$$

$$N = \begin{bmatrix} -0.4472 & 0.0811 & 0.1170 & 0.5549 & -0.1300 & -0.2666 & 0.1855 & -0.5392 & 0.2413 & 0.0192 \\ -0.4472 & 0.0431 & -0.0349 & -0.0004 & 0.2304 & -0.0653 & -0.5566 & 0.3892 & 0.5266 & -0.0316 \\ -0.4472 & 0.0811 & -0.8055 & -0.2487 & -0.1683 & 0.0336 & 0.1519 & -0.0463 & -0.1675 & -0.0003 \\ 0 & -0.0888 & -0.1555 & 0.0593 & 0.9143 & 0.1044 & 0.3137 & -0.1317 & 0.0400 & 0.0188 \\ -0.4472 & -0.2612 & 0.4183 & -0.4933 & -0.0041 & 0.4339 & -0.0434 & -0.3416 & -0.0659 & 0.0472 \\ 0.0000 & -0.9510 & -0.1487 & 0.1885 & -0.0939 & -0.1353 & -0.0158 & 0.1018 & -0.0006 & -0.0183 \\ 0.0000 & 0.0254 & -0.0932 & 0.4656 & -0.1465 & 0.8174 & 0.0966 & 0.1747 & 0.1201 & -0.1734 \\ 0.0000 & 0.0254 & -0.0932 & 0.2356 & 0.1714 & 0.0295 & -0.6296 & -0.3019 & -0.5428 & -0.3502 \\ -0.0000 & -0.0127 & 0.0669 & -0.2012 & -0.0307 & -0.1369 & 0.2397 & 0.0150 & 0.1891 & -0.9175 \\ -0.4472 & 0.0558 & 0.3051 & 0.1874 & 0.0721 & -0.1355 & 0.2626 & 0.5379 & -0.5345 & -0.0345 \end{bmatrix}$$

Being thus the equivalent pencil $\widehat{\Pi}(\xi) = \xi\widehat{E} - \widehat{A}$ the one provided below. Because of the size of $\widehat{\Pi}(\xi)$ we had to partition it as $\widehat{\Pi}(\xi) = [\widehat{\Pi}_1(\xi) \;\; \widehat{\Pi}_2(\xi)]$

$$\widehat{\Pi}_1(\xi) = \begin{bmatrix} 36 & -\xi - 4.50 & -0.0690\,\xi - 25 & -0.320\,\xi - 15 & 0.00170\,\xi - 6 & -0.550\,\xi + 11 & 0.00420\,\xi - 21 & -0.0560\,\xi - 43 \\ 0 & -1 & 1.10\,\xi + 0.0270 & 0.670\,\xi - 0.370 & 0.210\,\xi - 0.100 & -0.180\,\xi - 0.160 & -0.180\,\xi + 0.390 & 0.500\,\xi - 0.0560 \\ 0 & 0 & 1.40 & -0.350\,\xi - 1 & -0.400\,\xi - 0.110 & -0.970\,\xi - 1 & -0.140\,\xi + 1.70 & -0.0340\,\xi + 0.740 \\ 0 & 0 & 0 & 2.40 & 0.940\,\xi + 0.0820 & -0.350\,\xi + 1.60 & -0.250\,\xi - 1.20 & -0.190\,\xi + 0.630 \\ 0 & 0 & 0 & 0 & -1.50 & -1.40\,\xi + 1.10 & 0.680\,\xi + 2 & -0.0900\,\xi + 1.20 \\ 0 & 0 & 0 & 0 & 0 & -1.70 & -1.40\,\xi - 0.720 & -\xi - 0.650 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2.30 & 1.20\,\xi + 1.10 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1.30 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\widehat{\Pi}_2(\xi)\begin{bmatrix} 0.21\,\xi+43.0 & 2.2\,\xi+13.0 \\ -1.2\,\xi+0.52 & -0.50\,\xi-1.0 \\ -0.11\,\xi+0.55 & 1.6\,\xi-4.5 \\ -0.14\,\xi-1.5 & 0.77\,\xi+7.3 \\ 0.48\,\xi+2.0 & -1.7\,\xi+2.8 \\ 0.48\,\xi-0.97 & \xi-3.8 \\ 0.15\,\xi+1.1 & -0.72\,\xi-0.28 \\ -0.79\,\xi+0.37 & 0.21\,\xi-0.25 \\ 1.6 & -3.4\,\xi+0.0069 \end{bmatrix}$$

The associated matrix $Q(\xi)$ for this case is given below.

$$Q(\xi) = \begin{bmatrix} 0.1734 & 0.3502 & 0.1975 & 0.0346 \end{bmatrix} +$$

$$\xi \begin{bmatrix} 0.0202 & -0.0189 & 0.1233 & 0.0219 \end{bmatrix} +$$

$$\xi^2 \begin{bmatrix} 0.0197 & -0.0369 & 0.1233 & 0.0822 \end{bmatrix} +$$

Notice that this $Q(\xi)$ is *not the desired embedding* of $P(\xi)$ because $\det(W(\xi)) = (\xi - 2.216)(\xi+19.36)(\xi+2.124)(\xi+1.549\pm3.996j)(\xi+1.114\pm0.9672j)(\xi-0.5177\pm2.158j)(\xi-1.616\pm1.156j)$ is neither a zero constant nor Hurwitz.
The question addressed in this chapter is concerned with this: why does not Q have the theoretically predicted properties? The first explanation might be that the orthogonal transformations M and N are not really orthogonal: computing $M^{-1}\widehat{\Pi}(\xi)N^{-1}$ does not yield the original pencil $\Pi(\xi)$. This is true in fact but the deviations are of the same order as the machine precision, while the roots of $\det(W(\xi))$ are in normal range, so there has to be another difficulty in the procedure. In section 6.3 we will describe the geometry of the pencil space, in section 6.5 we calculate condition numbers for polynomial matrices and pencils.

6.3 The geometry of the orbit of a pencil

Elmroth ([21], and references therein) author gives a nice theoretical/practical point of view to study the structure of matrix pencils. We give a short overview of his results here. For any pencil define its orbit as the set of all pencils that are strictly equivalent to it in the sense that we defined in chapter 5

$$o(\xi E - A) = \{M(\xi E - A)N : \det(M), \det(N) \neq 0\}$$

In this way any matrix pencil $\xi E - A$ with real or complex entries defines a manifold of strictly equivalent pencils in the $2m_p n_p$ dimensional space. Hence, it is possible to say that

an orbit of matrix pencils is a set of pencils with the same Kronecker canonical form. In fact, if we find that for some pencil $\xi E - A$, $m_p \neq n_p$, then for almost all (E, A) it will have the same Kronecker structure, depending only on its size [21]. In this *generic* case the pencil has full rank $\forall \lambda \in \mathbb{C}$. In contrast, when the (non-square) pencil at hand does not have full rank for all $\lambda \in \mathbb{C}$, it has to be *non generic.*

Since the dimension of the orbit of $\xi E - A$ is equal to the dimension of the tangent space to the orbit at that point $(\xi E - A)$, it is possible to say that the tangent space is the range space of the following $2m_p n_p \times n_p^2 + m_p^2$ matrix T ($\otimes$ denotes (right) Kronecker product):

$$T = \begin{bmatrix} -A^T \otimes I_{m_p} & -I_{n_p} \otimes -A \\ -E^T \otimes I_{m_p} & -I_{n_p} \otimes -E \end{bmatrix}$$

We can define the normal space $\mathrm{nor}(\xi E - A)$, as the space perpendicular to its tangent space $\tan(\xi E - A)$. The dimension of the normal space is also known as the codimension of the orbit, $\mathrm{cod}(\xi E - A)$ which can be computed as the *number of zero singular values of* T. Finally, with all this, it is possible to compute for a given pencil a lower bound on the distance to the closest non generic pencil $\xi(E + \delta E) - (A + \delta A)$ by means of

$d = 1$	Size of $P(\xi)$	Size of $\xi E - A$	cod$(\xi E - A)$	$\|(\delta E, \delta A)\|$	$\det(W(\xi))$
Ex_T	(3,6)	(3,6)	0	0.3333	Ok
Ex_0	(2,3)	(2,3)	0	0.2998	Ok
Ex_1	(5,14)	(5,14)	0	0.4256	Ok
Ex_2	(7,9)	(7,9)	$0(\approx 1)$	0.0095	Not Ok
d=2					
Ex_1	(5,24)	(10,29)	0	0.1584	Not Ok
Ex_2	(4,6)	(8,10)	0(many ≈ 0)	8.243 $\times 10^{-5}$	Not Ok
d=3					
Ex_1	(3,4)	(9,10)	0 (many ≈ 0)	0.0014	Not Ok
Ex_2	(3,5)	(9,11)	0 (many ≈ 0)	2.2923 $\times 10^{-4}$	Not Ok
Ex_3	(4,6)	(12,14)	0 (many ≈ 0)	1.2286 $\times 10^{-4}$	Not Ok
d=4					
Ex_1	(4,5)	(16,17)	0 (many ≈ 0)	4.827 $\times 10^{-5}$	Not Ok
d=6					
Ex_K	(3,5)	(18,20)	66	7.2310 $\times 10^{-18}$	Not Ok
d=7					
Ex_M	(2,5)	(14,17)	350(many ≈ 0)	3.2097 $\times 10^{-18}$	Not Ok

Table 6.1: Geometric parameters.

$$||(\delta E, \delta A)|| \geq \frac{\sigma_{min}(T)}{\sqrt{m_p + n_p}} = \frac{\sigma_{min}(T)}{\sqrt{n - m}}$$

where the norm is the induced Euclidean matrix norm.
Now, we explain the content of table 6.1. The first column includes the set of examples for different degrees d. Next to it, we give the sizes of the corresponding polynomial matrices $P(\xi)$ under study as well as the size of their associated pencils (notice the way the size increases). Column 4 gives the number of zero singular values of T, or equivalently the codimension of the orbit of $\xi E - A$. If $\text{cod}(\xi E - A) = 0$, the dimension of the corresponding complementary space is $2m_p n_p$. In this case, $\tan \xi E - A$ spans the whole $2m_p n_p$ space. The lower bound specified above is in fact a corollary of a more general lower bound for the distance to a nongeneric pencil of codimension r, which involves the r smallest singular values of T. So if many singular values of T are small, the pencil could be close to a very nongeneric pencil. Column 6 specifies this number (the table counts the number of singular values smaller[1] than 10^{-8}). This number is zero for for the examples with $d = 1$ but not for $d > 1$. The latter implies that numerically the $2m_p n_p$ pencil will not be spanned and we run into computational problems.
Column 5 gives the calculated lower bound to the nearest non generic pencil. Since that pencil will have a different Kronecker structure, inaccuracies in the calculations of order bigger than this lower bound might lead to a wrong generalized Schur form, and hence to a wrong $\hat{K}, K$ and $Q(\xi)$.
The last column shows whether the outcome of the algorithm is indeed unimodular.

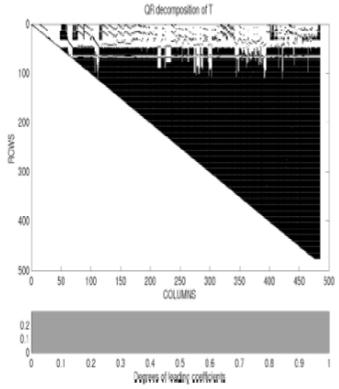

Figure 6.1: *Corruption of controllability.*Ex_1,$d = 7$

Finally look at figure 6.1 It gives an idea of how nearly singular T is by applying a QR

[1]We take this size since calculating distances involves taking square roots

factorization. Let us take Ex_1,$d = 7$. The black squares represent non zero entries and blank squares are zero entries. We observe that white zones are "invading" the black one. Many rows are practically blank. That represents that T is very close to lose rank, which is equivalent to say that $\xi E - A$ has practically lost controllability and is close to be rank deficient. This QR factorization was applied to all the examples. All of them shown the latter pattern.
Concluding we can say that for polynomial matrices of high degree, or with significantly more columns than rows, the linearized pencil is very close to a nongeneric pencil, in fact closer than the machine precision, which could explain the failure of the proposed algorithm.

6.4 Matrix pencils as mathematical relations

It is known [10] that a pencil can be defined as a relation. A matrix pencil $\xi E - A$,$E, A \in \mathbb{R}^{m_p \times n_p}$ defines the relation $\Xi \subset \mathbb{R}^{n_p} \times \mathbb{R}^{n_p}$

$$\Xi = \{(x, y) \in \mathbb{R}^{n_p} \times \mathbb{R}^{n_p} | Ey = Ax\} = \texttt{null}([A \quad -E])$$

for $\Xi \in \mathbb{R}^{2n_p \times 2n_p - mp}$. The latter has the pencil as a special case for $y = \xi x$. This relation has an advantage. It considers the joined effect of A and E in a single matrix (independent of ξ), $\texttt{null}([A - E])$. By constructing these relations for the twelve standard examples, and applying a QR factorization on each of them, it is possible to see how Ξ loses rank for big values of d. It is not surprising (after having seen the material of the latter sections) that Ξ also shows that the pencil (constructed from those two matrices) almost loses rank. More precisely, let us pay attention to an example to illustrate this. We took $Ex_1, d = 3$. In figure 6.2 we can observe the way that Ξ loses rank. We can see blanks. This fact

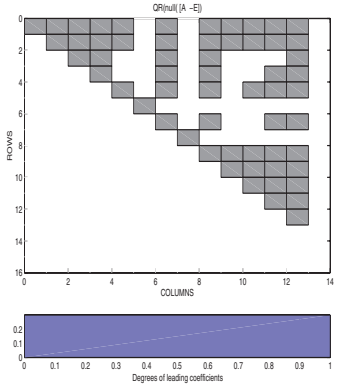

Figure 6.2: *QR(null([A -E])),*$Ex_1, d = 3$.

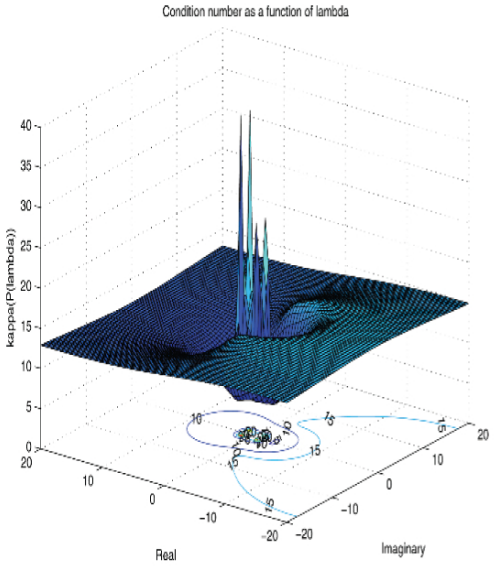

Figure 6.3: $\kappa(P(\lambda)), Ex_1, d = 3$.

expresses the same that we saw in the previous section: $\xi E - A$ is close to be degenerate.

6.5 Conditioning of the pencil

Another clue to understanding the failure of the algorithm is the condition function. Define for a polynomial matrix $P(\lambda)$ the condition function κ as the quotient of the largest and smallest singular value for each value of $\lambda \in \mathbb{C}$: $\kappa(P(\lambda)) = \sigma_{\max}(P(\lambda))/\sigma_{\min}(P(\lambda))$, where $\sigma_{\max}$ and $\sigma_{\min}$ denote the largest resp. smallest singular value of a matrix. Applying M, N to the pencil will lead to rounding errors in the calculation of $\xi\widehat{E} - \widehat{A}$ which will have the largest consequences for the $\lambda's$ with a high value of the condition function.
First we show some figures exhibiting the condition function $\kappa(P(\lambda))$ for the starting polynomial matrix $P(\lambda)$, and the one for the corresponding pencil, $\kappa(\Pi(\lambda)) = \sigma_{max}(\lambda E - A)/\sigma_{min}(\lambda E - A)$.

Example 38 *Let us consider the following polynomial matrix.*

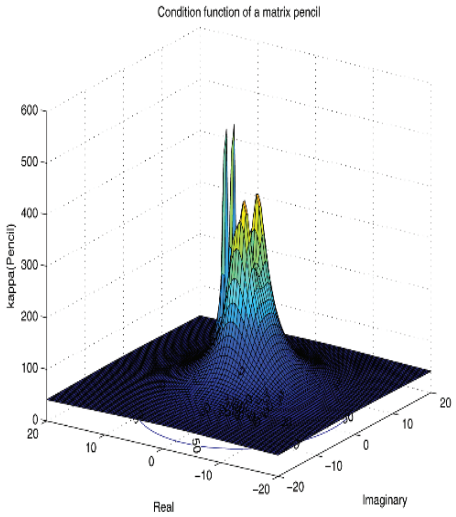

Figure 6.4: $\kappa(\Pi(\lambda)), Ex_1, d = 3$.

$$P = \begin{bmatrix} -1+\xi^2 & 1+\xi^3 & 11+\xi^3 & 7+\xi^2 \\ 2+\xi^3 & -4+\xi^3 & 1+3\,\xi+3\,\xi^3 & 22+\xi+\xi^2 \\ 1+\xi+\xi^3 & 5+\xi+\xi^2 & 9+\xi+\xi^3 & 78+\xi^2 \end{bmatrix}$$

The corresponding $\kappa(P(\lambda))$ is shown in figure 6.3. The figure reveals its deviating character. As usual the exponent $log(\kappa(H)))$ (for H any constant matrix) of the condition number indicates the number of significant decimal places that the computer can loose to roundoff errors. IEEE standard double precision numbers have about 16 decimal digits of accuracy, so if a matrix has a condition number of 10^{10}, you expect only six digits to be accurate in the answer. If the condition number is much greater than $1/\sqrt{\epsilon}$, caution is advised for subsequent computations. For IEEE arithmetic, the machine precision, ϵ, is about 2.2×10^{-16}, and $1/\sqrt{\epsilon} = 6.7 \times 10^8$. Although some of the pictures may look inoffensive in this sense, $\kappa(P(\lambda))$ certainly reveal the inherent difficult nature of handling a physical system represented by $P(\xi)$. Worse is handling an associated system $\Pi(\xi)$ (not numerically equivalent at all) with $\kappa(\Pi(\lambda))$ (see figure 6.4) because then we have a *conditioning gain function* $p(\lambda) = \kappa(\Pi(\lambda))/\kappa(P(\lambda))$. See also figure 6.5.

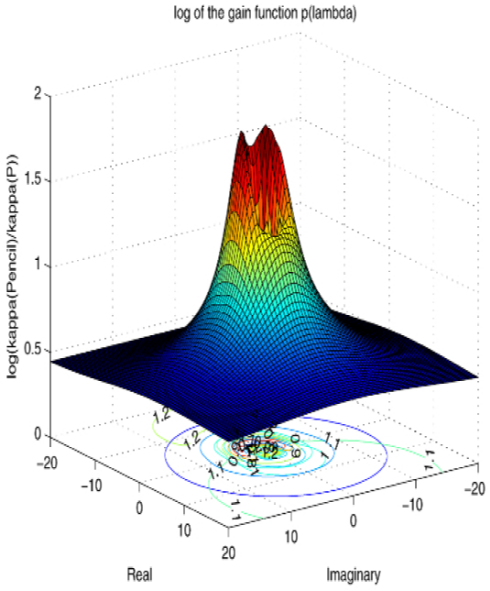

Figure 6.5: $\mathrm{p}(\lambda)$),$Ex_1, d = 3$.

In table 6.2 we show for all examples the maximum of the condition functions for both the polynomial matrix and the pencil. It is interesting to notice that the condition function for the pencil resembles that for the original polynomial matrix but that in general the condition function of the pencil is a factor (depending of λ) bigger than the one of the original P, and especially the peaks for the pencil are much more pronounced than those for the matrix. In all examples the condition number (at each λ) for the corresponding pencil is worse than the one for the original matrix.

Degree of P	Size of P	Size of Π	$\max(\kappa(P(\lambda)))$	$\max(\kappa(\Pi(\lambda)))$	$\max(\log(\kappa(\Pi(\lambda))/\kappa(P(\lambda))))$
Ex_T	(3,6)	(3,6)	1	1	0
Ex_0	(2,3)	(2,3)	9.6	9.6	0
Ex_1	(5,14)	(5,14)	12.6	12.6	0
Ex_2	(7,9)	(7,9)	116	116	0
d=2					
Ex_1	(5,24)	(10,29)	10	55	0.74
Ex_2	(4,6)	(8,10)	33	120	0.56
d=3					
Ex_1	(3,4)	(9,10)	40	900	1.35
Ex_2	(3,5)	(9,11)	30	220	0.86
Ex_3	(4,6)	(12,14)	55	1400	1.4
d=4					
Ex_1	(4,5)	(16,17)	4.5	2500	2.74
d=6					
Ex_K	(3,5)	(18,20)	10^6	10^8	2
d=7					
Ex_M	(2,5)	(14,17)	10^6	10^8	2.5

Table 6.2: Condition Numbers.

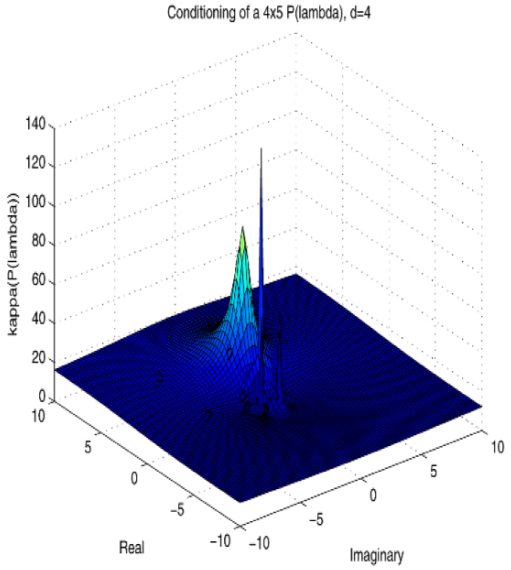

Figure 6.6: $\kappa(P(\lambda)), Ex_1, d = 4$.

In figures 6.6, 6.7 and 6.8 we show the graphs of $\kappa(P(\xi)), \kappa(\Pi(\xi))$ and $\log(\kappa(P(\xi))/\kappa(\Pi(\xi)))$ for the following example:

$$P = \begin{bmatrix} 0.59 & 4.70 & 4.80 & 9.60 & 6.10 \\ 0.89 & 9.10 & 6.00 & 6.00 & 7.00 \\ 2.70 & 6.00 & 1.60 & 0.29 & 0.92 \\ 4.10 & 3.30 & 8.30 & 8.10 & 4.20 \end{bmatrix} +$$

$$+\xi \begin{bmatrix} 3.80 & 4.50 & 7.70 & 6.40 & 4.90 \\ 1.70 & 9.60 & 4.40 & 2.50 & 4.10 \\ 8.30 & 1.50 & 6.20 & 3.50 & 4.60 \\ 8.40 & 8.70 & 9.50 & 1.90 & 6.10 \end{bmatrix} + \xi^2 \begin{bmatrix} 0.7100 & 6.20 & 4.60 & 4.00 & 3.90 \\ 3.10 & 2.50 & 5.40 & 3.10 & 5.00 \\ 6.10 & 5.90 & 9.40 & 4.10 & 7.20 \\ 1.80 & 5.10 & 3.40 & 2.90 & 3.10 \end{bmatrix} +$$

$$+\xi^3 \begin{bmatrix} 1.10 & 6.60 & 7.10 & 9.00 & 1.70 \\ 4.40 & 7.20 & 7.80 & 4.50 & 3.90 \\ 4.70 & 2.80 & 9.90 & 8.00 & 5.20 \\ 0.1500 & 2.60 & 4.70 & 8.30 & 7.20 \end{bmatrix} + \xi^4 \begin{bmatrix} 5.70 & 4.40 & 7.60 & 6.00 & 7.00 \\ 4.60 & 3.70 & 9.50 & 8.20 & 5.20 \\ 4.50 & 3.00 & 5.60 & 9.80 & 9.30 \\ 0.8800 & 8.50 & 0.1400 & 2.20 & 7.10 \end{bmatrix} +$$

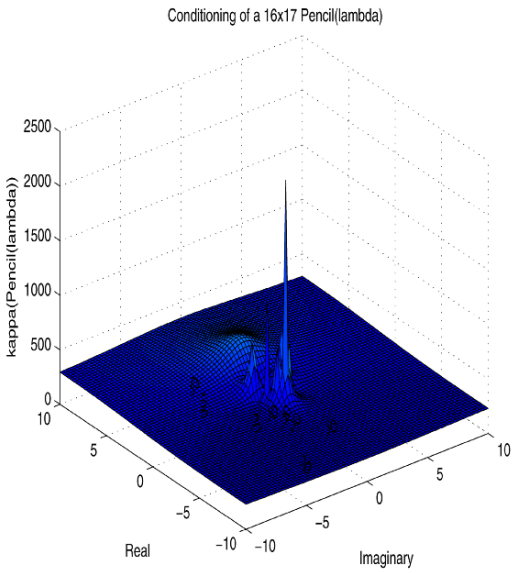

Figure 6.7: $\kappa(\Pi(\lambda)), Ex_1, d = 4$.

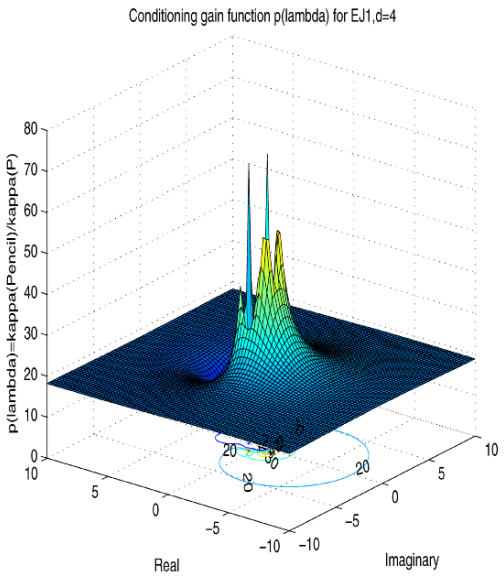

Figure 6.8: $\mathtt{p}(\lambda)), Ex_1, d = 4$.

6.6 Modelling polynomially and assessing numerically

Let us consider the following 4th order mechanical system illustrated in figure 6.9 [41]. Their parameters are: m_1=1, m_2=1; k_1=5, k_2=33, k_3=21; b_1=10, b_2=5, where $m_i, i = 1, 2$ are the masses (in kg), $k_j, j = 1, 2, 3$ are the spring stiffness constants (in N/m) and $b_q, q = 1, 2$ are the damper viscosity parameters (in $N/(m/s)$). The polynomial description of the plant is

$$P(\xi) = \begin{bmatrix} 960 & 0 & 54 & -33 \\ 0 & 960 & -33 & -38 \end{bmatrix} +$$

$$+\xi \begin{bmatrix} 670 & 0 & 5 & -5 \\ 0 & 670 & -5 & -15 \end{bmatrix} + \xi^2 \begin{bmatrix} 140 & 0 & 1 & 0 \\ 0 & 140 & 0 & -1 \end{bmatrix} +$$

$$+\xi^3 \begin{bmatrix} 20 & 0 & 0 & 0 \\ 0 & 20 & 0 & 0 \end{bmatrix} + \xi^4 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$P(\lambda)$ has full row rank for all $\lambda \in \mathbb{C}$, so $P(\xi)$ represents a *controllable* system [58]. Numerically speaking, it is known that a way to measure how much difficulty a system offers to be controlled is by means of its *condition number*. For each $\lambda \in \mathbb{C}$ this number is defined by

$$\kappa(P(\lambda)) = \frac{\sigma_{max}\left(P(\lambda)\right)}{\sigma_{min}\left(P(\lambda)\right)} \tag{6.1}$$

Obviously, getting the condition number as indicated above would be difficult and useless to describe the inner nature of $P(\xi)$. Nevertheless, we shall get helpful resources if we sweep the complex plane for a lot of values of $\lambda = a + bj \in \mathbb{C}$ (ideally for all of them) and we obtain $\kappa(P(\lambda))$ as we proceeded before. Then, we shall be able of finding out the

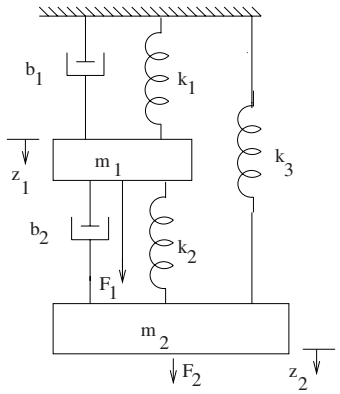

Figure 6.9: *Mechanical structure* $P(\xi)$.

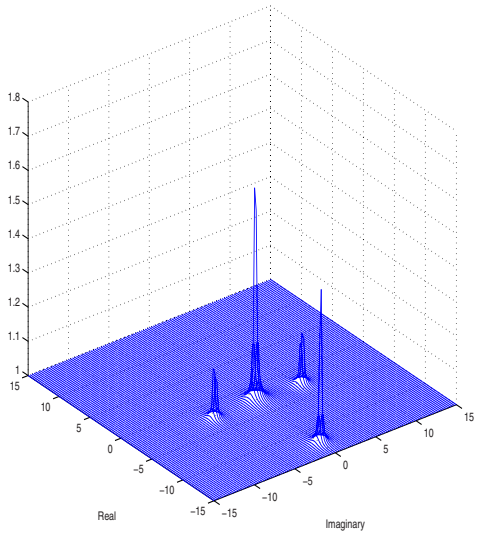

Figure 6.10: $\kappa(P(\lambda))$ *(scaled) of the mechanical structure*

condition number at *each* λ (i.e., at each frequency ω given in rad/s). The latter will reveal again an interesting pattern shown in figure 6.10.

We noticed that, although $P(\lambda)$ has full row rank *for all* $\lambda \in \mathbb{C}$, such a property is a little bit "damaged" (but not destroyed!) at four points of the complex plane. It is there, at each point where those values of λ, $\widehat{\lambda}$ produce a "high" condition number. They behave almost as eigenvalues because it is there when $P(\lambda)$ is closest to lose rank numerically. These numerical *"quasi eigenvalues"* are nothing but the poles of the mechanical structure. They are placed at $\widehat{\lambda}_1 = -12.40$, $\widehat{\lambda}_{2,3} = -2.72 \pm 5.35j$, $\widehat{\lambda}_4 = -2.16$ where $\kappa(\widehat{\lambda}_1) = 13.58$, $\kappa(\widehat{\lambda}_{2,3}) = 2.17$ and $\kappa(\widehat{\lambda}_4) = 3.10$, resp. As well, we realize that at each value of $\widehat{\lambda}$ there is a corresponding $\widehat{\omega}$ at which the system is more difficult to control than for other value of ω. These $\widehat{\lambda}$ behave as *"quasi uncontrollable"* modes of the mechanical plant, i.e., the system will be more reluctant to be controlled at those frequencies $\widehat{\omega}$. Moreover, in [95] and [96] we remarked that linearizing $P(\xi)$ via $\xi E - A$ the uncontrollability and instability of $W(\xi)$ were activated at the same time (the embedding problems are not numerically solved) because $\xi E - A$ computationally almost loses rank. The latter was explained in detail in [96] and from above, we have a clearer idea about why this happens. This situation is linked to the fact that as higher the peaks produced at the poles $\widehat{\lambda}$ as higher the plane $\kappa(P(\lambda))$ is (with respect to the original $\kappa(P(\lambda)) = 1$, i.e., the whole surface $\kappa(P(\lambda))$ is lifted

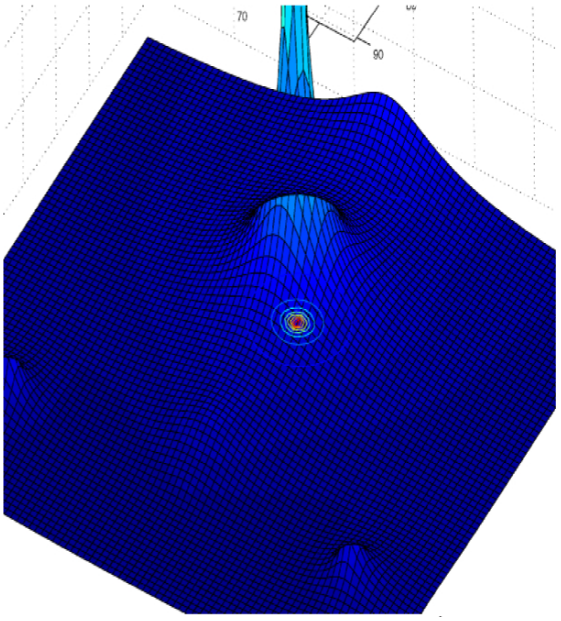

Figure 6.11: *Zooming from below at the pole* $\widehat{\lambda_1}$ *in* $\kappa(P(\lambda))$.

up by means of the peaks). In addition, this lifting up action creates a level curves effect on $\mathbb{C}$, appearing roundish contours on $\mathbb{C}$ which defines new stability/instability regions created by numerical/physical effect on $\xi E - A$ (see figure 6.11 where the concentric circles represent the new stability/instability area). This means that although the poles of the system are stable, linearizing its degree d polynomial version adds unstable (*regions of*) new poles in $\xi E - A$. All of this damage seriously the "embedding" $W(\xi)$. We can have an idea of that by zooming (from below) on $\widehat{\lambda_1}$ (figures 6.11, 6.12). In figure 6.13 we show the function $p(\lambda)$ which has the same shape as $\kappa(\Pi(\lambda))$.
For instance $P_{g_1} = [sI \;\; I]$, $P_{g_2} = [I \;\; sI \;\; s^2I]$, etc. are examples of that. Naturally, these polynomial matrices are perfectly conditioned,i.e.,$\kappa(\lambda) = 1$ (they are perfectly controllable for all frequencies ω) because its condition number is always one, something that can not occur to non generic structures. In addition, we notice that for almost all frequencies ω (for almost all $\lambda \in \mathbb{C}$) the conditioning of $P(\lambda)$ is perfect ($\kappa(P(\lambda)) = 1$). But there are some frequencies $\widehat{\omega}$ at which controlling the plant will be more difficult ($\kappa(P(\widehat{\lambda}) > 1)$). When we consider $\xi E - A$ and $P(\xi)$ equivalents, the latter effects are much more stressed. In the latter sections we have described some numerical issues which show the inherent difficulty to deal with the linearization $\xi E - A$. Nevertheless, we need to remember that linearizing $P(\xi)$ via $\xi E - A$ is the key step in this way of constructing an embedding $Q(\xi)$ for $P(\xi)$. After having completed this stage and once with some $Q(\xi)$ at hand

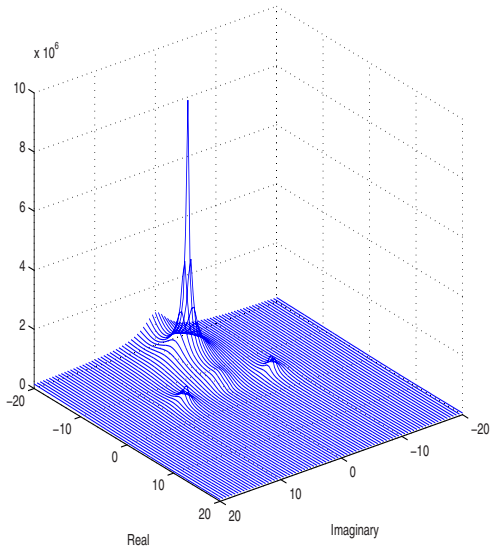

Figure 6.12: $\kappa(\lambda E - A)$ *associated to the mechanical model*

(which will be numerically "contaminated") we still have to verify whether the embedded matrix $W(\xi) = [P(\xi); Q(\xi)]$ is unimodular (which naturally will not be the case), i.e., the determinant of $W(\xi)$ will need to be a non zero constant. Besides, computing the determinant of a polynomial matrix offers some difficulties. This fact complicates the numerical solution of the embedding problem even more. Some problems linked to this fact are briefly described below.

6.7 Computing the determinant of a polynomial matrix

It is known that the degree q of $det(W(\xi))$ is given by the number of finite zeros of $W(\xi)$, $\nu(z_f(W))$ in terms of the following expression [99]:

$$q = \nu(z_f(W)) = \nu(p_\infty(W)) - \nu(z_\infty(W))$$

where $\nu(p_\infty(W))$ and $\nu(z_\infty(W))$ denote algebraic multiplicity of the pole and zero of $W(\xi)$ at infinity, respectively. Naturally, since $W(\xi)$ is theoretically unimodular, $q = 0$ and $\nu(p_\infty(W))$=$\nu(z_\infty(W))$. However, recalling our first example considered in the appendix

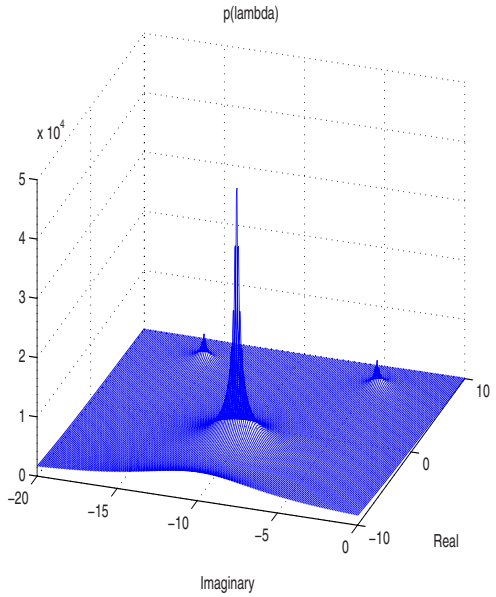

Figure 6.13: *Conditioning gain* $p(\lambda)$.

($Ex_1,\ d = 1$), we saw that $q = 2$. If we compute the same example with SCILAB [69] (Leverrier's command based *detr*), the determinant will be given as

$$\det(W(\xi)) = (-0.966\epsilon)\xi^2 + (-31.4088\epsilon)\xi - 6.9719888$$

where ϵ is the machine precision. Nevertheless, the command *determ* (the one which is implemented via FFT) produces $det(W(\xi)) = -6.9719888$. On the other hand, for this example, the palindromic form $W^{\sharp}(\xi)$ of $W(\xi)$ reveals the corresponding infinite structure of this embedded result. Such an infinite structure says that we have two poles $\lambda_{p\infty}(W) = \{\lambda_1^1, \lambda_1^1\}$ plus one infinite zero of degree zero, $\lambda_\infty(W) = \{\lambda_1^0\}$. The finite structure will be given by the roots of the determinant (quite variable between some MATLAB releases and SCILAB). The Polynomial Toolbox, for instance ([59]) rounds off numbers up to two significative digits. That produces two finite roots (which of course would not be present in a unimodular matrix!): $\lambda_f(W) = \{-84.0205, 80.9522\}$. We can get all the zeros $\lambda(W)$ by means of the MATLAB command "polyeig". The behavior in SCILAB is quite alike to this one. In fact there, this determinant is thrown by a two commands: one is Leverrier's based algorithm [71] and the second one available considers an FFT (Fast Fourier Transform) based instruction. Although it does not seem to exist a stability analysis in the literature for the former, what does exist is an analysis and comparison of calculating these kind of determinants using the FFT and other forms (see [70] and references therein). However,

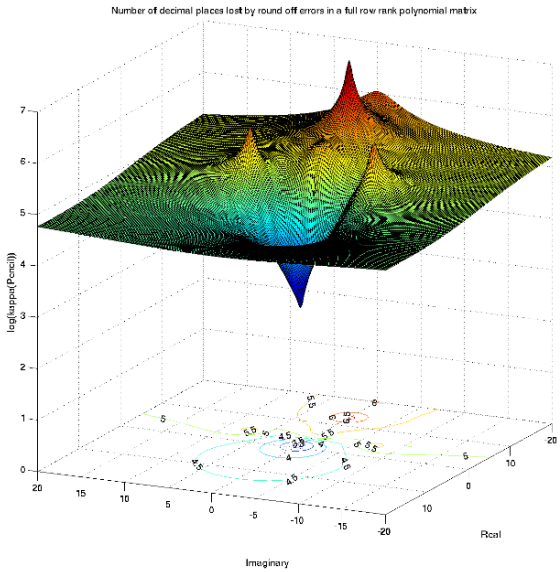

Figure 6.14: *Number of digits lost due to round off errors in* $\kappa(\Pi(\lambda))$

even using the latter form to evaluate the determinants of our embeddings they did not improve a lot, which is attributed to the consideration of the numerical (finite) value of the multiple $\lambda = \infty$ during the embedding process among other reasons sketched below. Computing zeros of a polynomial matrix via its determinant is not very stable as we verified here (and as it is known in our days [99], [70]) because calculating a determinant is already difficult. In addition, we have to compute the roots of such determinant, which is not easy either. Additionally, determining the rank of matrices implies arising of numerical instability in natural way. Besides, the conditioning (sensitivity to small perturbations) of linearizations has to be taken into account because different linearizations for a given polynomial matrix can have different condition numbers. Finally, we remark that all the algorithms involved in [5], [95], [96] were programed in MATLAB and SCILAB.

6.8 Concluding remarks

And then shall we linearize in order to embed? The answer to this question is now clear. According to [95], [96], and the content of this chapter, we rather could say what we should not do in order to linearize (and hence to embed) $P(\xi)$. Any linearization has a set of numerical drawbacks. This comes as a result of the physical/numerical characteristics

a given plant $P(\xi)$ has, because the numerical problems exhibited by $\xi E - A$ are nothing but an amplified effect of the the dynamics displayed by $P(\xi)$ reflected computationally in its numerical model $P(\xi)$. Up to this extend, it seems we should find a linearization able enough to avoid this or we could find a different way of constructing an embedding for $P(\xi)$. Up to now, embedding numerically $P(\xi)$ generate unstable eigenvalues of the controlled system $W(\xi)$, defining then new rounded stability/instability areas in $\mathbb{C}$ (as a result of the level curves projected on $\mathbb{C}$ which can be observed in the 3D surfaces $\kappa(P(\lambda)), \kappa(\lambda E - A)$). We can not get rid of this yet. Moreover, several linearizations were proposed while we investigated the one discussed here with no success.
After collecting the observations we have, we can make the following remarks:

a. The linearization as proposed by Beelen and Van Dooren does not lead to a numerically reliable algorithm.

b. The geometry of the space of pencils seems to be weird, in the sense that non-generic pencils seem to be scattered all over the space, such that every pencil, certainly in higher dimension is awfully close to a non-generic one.

c. A suitable linearization method has to guarantee at least that some of the geometric and numeric problems that we encountered are circumvented. At this moment we do not see such a linearization.

d. An additional argument for this is the fact that the ratio of the condition functions of $P(\xi)$ and the associated pencil has large values for all examples considered.

e. Most of the arguments do apply in general for linearizations, they are restricted to application to the embedding problem.

We realize that other authors reported difficulties with the linearization before, for instance Byers, He and Mehrmann [14], Iwata and Shimizu [32] and Pervouchine [56]. Concluding, we can say that the linearization proposed in [4], causes an ill conditioned solution for the embedding problems.
In this aspect it is interesting to realize that the polynomial toolbox in MATLAB does not use linearization in the calculation of the null space of a polynomial matrix, in fact the main application described in the thesis of Beelen. As we said before, our algorithms were implemented in MATLAB. In order to have a wider perspective, our embedding algorithms were programmed in SCILAB as well [69]. This package does contain some computer commands developed in [4]. We noticed that the algorithms in SCILAB performed slightly better than our own programs in MATLAB. We attribute the latter to the use of FORTRAN based numerically optimized subroutines (taken from BLAS [45] and LAPACK [44]).

Chapter 7

A new algorithm for embedding problems

In this chapter we describe the algorithm for embedding problems that we promised in chapter 5. Let us first quickly rephrase the problem:
Let $P(\xi)$ be a polynomial matrix with m rows and n columns, where $m < n$. The unimodular embedding problem is to find a polynomial matrix $Q(\xi)$ with $n - m$ rows and n columns such that the stacked matrix

$$\begin{bmatrix} P(\xi) \\ Q(\xi) \end{bmatrix}$$

is unimodular. Recall that such a $Q(\xi)$ exists if and only if the Smith form of $P(\xi)$ is equal to $[I_m\ 0]$, where I_m denotes the $m \times m$ identity matrix. Below we will describe an algorithm to compute a required $Q(\xi)$ [98].

7.1 The algorithm

If P has full row rank then there exists a unimodular matrix U of size n such that $PU = [P^*\ 0]$, where P^* is a square nonsingular matrix. If $P(\lambda)$ has full row rank for every $\lambda \in \mathbb{C}$, the same holds for P^*, so then P^* is unimodular:

Lemma 11 *Let $P \in \mathbb{R}[\xi]^{m\times n}$ have full row rank for all $\lambda \in \mathbb{C}$, then there exists a unimodular U such that $PU = [P^*\ 0]$, with P^* unimodular.*

If $U^{-1} = [V_1; V_2]$ with V_2 of size $(m - n) \times n$, then $Q = V_2$ will solve the problem:

Theorem 17 *Let $P \in \mathbb{R}[\xi]^{m\times n}$ have full row rank for all $\lambda \in \mathbb{C}$, and let $V = [V_1;\ V_2]$ be such that $PV^{-1} = [P^*; 0]$ as in lemma 11, then $[P; V_2]$ is unimodular.*

Proof. Clearly P^* is unimodular and V is unimodular, so

$$\left[\begin{array}{c|c} P^* & 0 \\ 0 & I \end{array}\right] V = \begin{bmatrix} P \\ V_2 \end{bmatrix}$$

R.Z. Yoe: *Modelling and Control of Dynamical Systems: Numerical Implementation in a Behavioral Framework*, Studies in Computational Intelligence (SCI) **124**, 135–140 (2008)
www.springerlink.com

is unimodular.

Note that this actually works for any embedding problem for which the resulting polynomial matrix can be post multiplied by a unimodular matrix without losing the desired property. This holds in particular for the stable embedding problem: if W is Hurwitz, so is WV for any unimodular V.
This leads to the following algorithms for the embedding problems considered in this book:

step 1. (Column compression) . Find a $n \times n$ unimodular polynomial matrix $U(\xi)$ such that $P(\xi)U(\xi) = [P^*(\xi)\ 0]$ with $P^*(\xi)$ full column rank.

step 2. Check the number of columns p of $P^*(\xi)$. If $p < m$ then a required $Q(\xi)$ does not exist (in that case $\text{rank}(P(\lambda))$ is at most $p < m$ for any complex value of λ).

step 3. If $p = m$, then check whether $P^*(\xi)$ is unimodular. If not, then again a required $Q(\xi)$ does not exist (in that case there exists a value for λ such that $\text{rank}(P(\lambda)) < m$).

step 4. If $P^*(\xi)$ is unimodular, then a required $Q(\xi)$ does exist. We compute one as follows. Solve the polynomial equation $Q(\xi)U(\xi) = [0\ I_{n-m}]$. Then

$$\left[\begin{array}{c} P(\xi) \\ Q(\xi) \end{array}\right] U(\xi) = \left[\begin{array}{c|c} P^*(\xi) & 0 \\ 0 & I_{n-m} \end{array}\right]$$

Since the matrix on the right in this equation is unimodular, the same holds for the matrix on the left. Thus $Q(\xi)$ does the job.

Now let $P(\xi)$ be a polynomial matrix with m rows and n columns, where $m < n$. The goal is now find a polynomial matrix $Q(\xi)$ with $n - m$ rows and n columns such that the stacked matrix

$$\left[\begin{array}{c} P(\xi) \\ Q(\xi) \end{array}\right]$$

is Hurwitz. Obviously such $Q(\xi)$ exists if and only if $P(\lambda)$ has full row rank m for every complex number λ with $\text{Re}(\lambda) \geq 0$. Below we describe the algorithm to compute a required $Q(\xi)$.

step 1. (Column compression). Find a $n \times n$ unimodular polynomial matrix $U(\xi)$ such that $P(\xi)U(\xi) = [P^*(\xi)\ 0]$ with $P^*(\xi)$ full column rank.

step 2. Check the number of columns p of $P^*(\xi)$. If $p < m$ then a required $Q(\xi)$ does not exist (in that case $\text{rank}(P(\lambda))$ is at most $p < m$ for any complex value of λ).

step 3. If $p = m$, then check whether $P^*(\xi)$ is Hurwitz. If not, then again a required $Q(\xi)$ does not exist (in that case there exists a value for λ with $\text{Re}(\lambda) \geq 0$ such that $\text{rank}(P(\lambda)) < m$).

step 4. If $P^*(\xi)$ is Hurwitz, then a required $Q(\xi)$ does exist. We compute one as follows. Solve the polynomial equation $Q(\xi)U(\xi) = [0\ I_{n-m}]$. Then

$$\begin{bmatrix} P(\xi) \\ Q(\xi) \end{bmatrix} U(\xi) \left[\begin{array}{c|c} P^*(\xi) & 0 \\ 0 & I_{n-m} \end{array}\right]$$

Since the matrix on the right in this equation is Hurwitz, the same holds for the matrix on the left. Thus $Q(\xi)$ does the job.

Notice that the embedding of $[P^*; 0]$ obtained in lemma 11 does not need necessarily to be in terms of the identity matrix i.e. $[0; I]$. If we rather take Q as $Q = \Omega V_2(\xi)$ (with Ω non singular) the embedding of $[P^*; 0]$ is $[0; \Omega]$. Moreover, we may take $Q = \Omega(\xi)V_2(\xi)$ in order to obtain a more general embedding $[0; \Omega(\xi)]$ for $[P^*; 0]$. In (numerical) practice this is more difficult (as we shall see below where we provide the MATLAB based pseudocode to implement this) because then we would have to solve a polynomial matrix equation of the form $QU = \Gamma$ where Γ is polynomial rather than constant (see listings of chapter 3 and [59]).

Algorithm 12

```
% Column compression of P(ξ)
% (we compute a unimodular U(ξ) s.t.)
P(ξ)U(ξ) = [P*(ξ)  0]
% P*(ξ) has full column rank (fcr)
[m*, n*]=size(P*(ξ))
if n* == m* and det(P(ξ))=k≠0,k ∈ ℝ or
real(roots(det(P*)))<0
     Define Γ := [0  Ω],  Ω ∈ ℝ^((n−m)×(n−m))
     Determine Q(ξ) from Q(ξ)U(ξ) = Γ(ξ)
else
     % There does not exist such a Q(ξ)
     error('P(ξ) can not embedded')
end
```

7.2 Inside the algorithm

To find the embedding we used column compressing in the previous section. At first sight this seems a circle: column compression can be achieved by calculating a Minimal Right Annihilator $R(\xi)$ for a polynomial matrix $P(\xi)$ and embedding $R(\xi)$ that into a unimodular matrix.
Therefore we use not simply column compression, but column reduction: instead of finding a minimal right annihilator we find a unimodular $U(\xi)$ such that

$$P(\xi)U(\xi) = [P^*(\xi)\ 0],$$

where the columns of $P^*(\xi)$ form a minimal polynomial basis (in the sense of Forney [23]) of the column space of $P(\xi)$.
This is achieved by the algorithm `colred.m` in the polynomial toolbox [59]. Colred uses either the factor extraction method of Callier, or the iterative algorithm of Beelen, van der Hurk and Praagman [7].

7.3 Numerical computation

The computer implementation is based in the explanation given below. It resembles the way we apply the Smith form to this problem, although, as it is known, this popular matrix form is not reliable numerically speaking (see chapter 6).
Let $P(\xi)$ be an m by n full row rank polynomial matrix. Let us say that $m < n$. Naturally, a row compression of $P(\xi)$ will not be useful because of the property of full row rank matrix $P(\xi)$ has (it would not compress anything). On the other hand, if we transpose $P(\xi)$ we could row compress $P^T(\xi)$ applying a unimodular transformation $U(\xi)$ as $U(\xi)P^T(\xi) = [P(\xi)^*; 0]$. An obvious choice to enhance $[P(\xi)^*; 0]$ is $[0; I]$. In this way we get an embedding for $[P(\xi)^*; 0]$. Solve $U(\xi)Q(\xi) = [0; I]$ then leads to the desired embedding Q for P. This equation is solved in MATLAB by using the standard command `axb`.
Note that $U(\xi)$ is known (it was used to rowcompress $P^T(\xi)$ and we have only one matrix unknown, the embedding $Q(\xi)$). Of course, we can start with a full row rank matrix $P(\xi)$ $(m > n)$ without considering its transpose, $P^T(\xi)$ and applying column compression.

7.4 Examples

Next, we show some full row rank polynomial matrices $P(\xi)$ embedded by another full row rank matrix $Q(\xi)$. The resultant matrix $W(\xi)$ is either unimodular or Hurwitz as we shall see. The constant value of $\det(W(\xi))$ is also given.

Example 39 *Let us consider the following polynomial matrix.*

$$P = \begin{bmatrix} -1+\xi^2 & 1+\xi^3 & 11+\xi^3 & 7+\xi^2 \\ 2+\xi^3 & -4+\xi^3 & 1+3\,\xi+3\,\xi^3 & 22+\xi+\xi^2 \\ 1+\xi+\xi^3 & 5+\xi+\xi^2 & 9+\xi+\xi^3 & 78+\xi^2 \end{bmatrix}$$

The matrix $Q(\xi)$ which embeds $P(\xi)$ into a unimodular one $W(\xi)$ is given by

$$Q = \begin{bmatrix} 1.0000 & -0.1177 & 0.0000 & 1.0448 \end{bmatrix} +$$

$$+\xi \begin{bmatrix} -0.0524 & -0.0974 & 0.3198 & 0.3007 \end{bmatrix} +$$

$$+\xi^2 \left[\begin{array}{cccc} -0.1414 & 0.1514 & 0.3661 & -0.0566 \end{array} \right] +$$

$$+\xi^3 \left[\begin{array}{cccc} 0.1593 & -0.0380 & 0.2992 & 0.0000 \end{array} \right] +$$

In this case $\det(W(\xi)) = -3.9 \times 10^3$.

Example 40 *Now we consider a polynomial matrix $P(\xi)$ which is stabilizable (full row rank for all $\lambda \in \overline{\mathbb{C}^+}$ and hence the embedded W is Hurwitz.*

$$P = \left[\begin{array}{ccccc} 2+3\xi+\xi^2 & 0 & 0 & -1 & 1 \\ 0 & 2+3\xi+\xi^2 & 0 & 4-\xi-\xi^2 & 8+\xi-2\xi^2 \\ 0 & 0 & 1+\xi & 2-\xi & 4-2\xi \end{array} \right]$$

The corresponding embedding $Q(\xi)$ is as follows.

$$Q = \left[\begin{array}{ccccc} -3.4\xi & 0.29\xi & 1 & -0.29\xi & -0.57\xi \\ -1.2\xi - 0.53\xi^2 & 0.044\xi^2 & 0.16\xi & 1 - 0.067\xi - 0.044\xi^2 & -0.089\xi^2 \end{array} \right]$$

where $\det(W(\xi)) = (\xi+1)(\xi+2)$.

Example 41 *In order to finish with the examples section we give an example with degree $d = 5$.*

$$P(\xi) = \left[\begin{array}{ccc} -1.6656 & -0.0376 & 2.1832 \\ -0.8323 & 0.8580 & 0.6900 \end{array} \right] +$$

$$+\xi \left[\begin{array}{ccc} 0.1253 & 0.3273 & -0.1364 \\ 0.2944 & 1.2540 & 0.8156 \end{array} \right] +$$

$$+\xi^2 \left[\begin{array}{ccc} 0.2877 & 0.1746 & 0.1139 \\ -1.3362 & -1.5937 & 0.7119 \end{array} \right] +$$

$$+\xi^3 \left[\begin{array}{ccc} -1.1465 & -0.1867 & 1.0668 \\ 0.7143 & -1.4410 & 1.2902 \end{array} \right] +$$

$$+\xi^4 \left[\begin{array}{ccc} 1.1909 & 0.7258 & 0.0593 \\ 1.6236 & 0.5711 & 0.6686 \end{array} \right]$$

$$+\xi^5 \left[\begin{array}{ccc} 1.1892 & -0.5883 & -0.0956 \\ -0.6918 & -0.3999 & 1.1908 \end{array} \right]$$

This matrix is embedded by $Q(\xi)$. Such matrix is exhibited below.

$$Q(\xi) = \left[\begin{array}{cccc} 0.2758 & 0.2544 & 0.5636 \end{array} \right] +$$

$$\xi \begin{bmatrix} -0.6097 & -0.9607 & 0.4723 \end{bmatrix} +$$

$$\xi^2 \begin{bmatrix} 0.4089 & -0.5812 & 0.4048 \end{bmatrix} +$$

$$\xi^3 \begin{bmatrix} 0.0859 & 0.0589 & 0.3940 \end{bmatrix} +$$

$$\xi^4 \begin{bmatrix} -0.7086 & 0.0212 & 0.5608 \end{bmatrix}$$

Here, $\det(W(\xi)) = -1.5$.

7.5 Summary

In this chapter we offered a solution to the unimodular/stable embedding problems in a relative simple way. We provided some working examples which verify the theory. From those examples (and from many others we work with) we observed that the degree d_q of the embedding matrix $Q(\xi)$ is *almost always* equal to the degree d of $P(\xi)$. Finding bounds for d_q still needs to be done.

Conclusions and further research

In this book the behavioral point of view is considered in a polynomial matrix perspective. As such, the tools we need to construct, to develop all the theoretical machinery are: right/left annihilators, row/column compressions, solutions of polynomial matrix equations of the type $AX = B$ and $XA = B$ as well as unimodular or stable embeddings. As we saw, actual numerical implementation of the latter operations represent *additional problems*, which may in fact lead to *new* solutions as illustrated by the embedding problem solved in chapter 7. During many years, the paradigmatic equivalent pencil was used without knowing that this representation defines an ill posed problem (if we consider rectangular polynomial matrices where $m \times n,\ m < n$). The numerical experiments in fact showed that the difficulties that arise reflect the internal nature of the system represented by a polynomial matrix. The alternative theory we have developed in this book is numerically implementable in a computer.

At the end of this project we could say that numerical implementation of existing theory produced problems which eventually led to an enrichment of the crude theoretical part, closing in a sense a virtuous circle.

The set of MATLAB based commands developed in this work will be joined to the older version of the Behavioral Toolbox [19], [17].

On the other hand, since the family of all implementing and stabilizing controllers have been found, a natural step further would be to robustify our designs with respect to some requirement. For instance, we may find those sets of all controllers for a perturbed plant P_Δ in such a way the perturbed system $P + P_\Delta$ preserves the full row rank property. We may find stability radii to provide robust margins in the frequency domain (λ). Once in this direction, we may optimize our designs with respect to some performance. Optimizing is linked to derivatives, in this case, we may work with the derivatives of the eigenvalues with respect to some parameter, or as we did in [96] obtaining derivatives of $P, \lambda E - A$ with respect to λ might define an optimization measure.

Bibliography

[1] L. Aavsland, P. Bjorstad, *The generalized eigenvalue problem in shipdesign and offshore industry- A comparision of traditional methods with Lanczos process*, Proceedings Pite Havsbad, 1982. Lecture Notes in Mathematics (Matrix Pencils). Springer Verlag 1982.

[2] P.J. Antsaklis, Some relations satisfied by prime polynomial matrices and their role in linear multivariable system theory, *IEEE Trans. Automat. Contr.*, Vol. AC-24, pp. 611-616. Aug. 1979.

[3] Z. Bai, J. Demmel, A. McKenney, On computing condition numbers for the nonsymetric eigenproblem, *ACM Transactions on Mathematical Software.* Vol. 19, No. 2, June 1993, 202-223.

[4] Th.G.J. Beelen, *New algorithms for computing the Kronecker canonical form of a Pencil with applications to systems and control theory*, PhD Thesis, Eindhoven Institute of Technology, 1985.

[5] Th.G.J. Beelen, P. Van Dooren, A pencil approach for embedding a polynomial matrix into a unimodular matrix, *SIAM J. Matrix Anal. Appl.,* Vol. 9, No.1, Janaury 1988.

[6] Th.G.J. Beelen, P. Van Dooren, An improved algorithm for the computation of Kronecker's canonical form of a singular pencil, *Linear Algebra and its Applications*, 105 (1988) 9-65.

[7] Th.G.J. Beelen, G.J.H.H. van den Hurk, C. Praagman, A new method for computing a column reduced polynomial matrix, *Systems & Control Letters*, Vol. 10 (1988) 217-224.

[8] M.N. Belur, *Control in a Behavioral Context*, PhD Thesis, University of Groningen, The Netherlands, 2003.

[9] M.N. Belur, H.L. Trentelman, Stabilization, pole placement and regular implementability, *IEEE Transactions on Automatic Control*, Vol. 47, nr. 5, pp. 735-744, 2002.

[10] P. Benner, R. Byers, An arithmetic for rectangular matrix pencils, In A. Beghi, L. Finesso, G. Picci, editors, *Mathematical Theory for Networks and Systems*, pages 573-576, Il Poligrafo, Padova, Italia, 1998.

[11] M. Bisiacco, M. Valcher, A note on the direct sum decompositions of two-dimensional behaviors, *IEEE Trans. Circuits Systems*, Vol. 48, nr. 4, pp. 490-494, 2001.

[12] G. Bountry et al., The generalized Eigenvalue problem for Non-Square Pencils Using a Minimal Perturbation Approach, *SIAM Journal On Matrix Analysis and Applications*, May 2003.

[13] R.W. Brockett, *Control theory and analytical mechanics.*, The 1976 Ames Research Center (NASA) Conference on Geometric Control Theory (Moffett Field, Calif., 1976), pp. 1-48. Lie Groups: History, Frontiers and Appl., Vol. VII, Math Sci Press, Brookline, Mass., 1977.

[14] R. Byers, C. He, V. Mehrmann, Where is the nearest non-regular pencil?, *Linear Algebra and its Applications*, 285 (1998) 81-105.

[15] C. Chen, *Linear Systems Theory and Design*, HRW Series in Electrical and Computer Engineering. M.E. Van Valkenburg Series Editor, NY, 1980.

[16] M. Chiu, Inverse eigenvalue problems, *SIAM REV.*, Vol. 40, No.1, pp. 1-39, March 1998.

[17] T. Cotroneo, *Algorithms in behavioral systems theory*, Ph.D. thesis, Groningen, 2001.

[18] C.A. Desoer, E.S. Kuh *Basic Circuit Theory*, International Student Edition. McGraw-Hill Book Company, Tokyo, Japan, 1969.

[19] J. van Dijk, T. Cotroneo, Modelling with the Behavioral Approach, *Mathematical and Computer Modelling of Dynamical Systems*, Taylor and Francis, Volume 8, Number 4, December 2002, pp. 361-376.

[20] R. Eising, Polynomial matrices and feedback, *IEEE Trans. Automat. Control*, AC-30, (1985), pp. 1022-1025.

[21] E. Elmroth, *Matrix Computations: Factorizing in Parallel and Surfing the Kronecker Structure Hierarchies*, PhD Thesis, 1995. Department of Computing Science. Umea University, Sweden.

[22] E. Fermi, *Thermodynamics*, Prentice Hall, Inc. New York, N. Y., 1937.

[23] G.D. Forney, Minimal bases of rational vector spaces with applications to multivariable linear systems, *SIAM J. Control*, Vol. 13, pp 439-520, 1975.

[24] J.M. Gracia, I. de Hoyos, F. E. Velasco, Safety Neighbourhoods for the Kronecker Canonical Form, *http://www.vc.ehu.es/campus/centros/farmacia/deptos-f/depme/gracia1.htm*

[25] F. R. Gantmacher, *The theory of matrices*,Vol. I, Vol. II, Chelsea Publishing Co. New York, N. Y., 1971.

[26] J. Giedymin, Geometrical and physical conventionalism of Henri Poincaré in epistemological formulation, *Stud. Hist. Philos. Sci.*, 22, no. 1, 1-22, 1991.

[27] G.H. Golub, C.F. Van Loan, *Matrix Computations*, Oxford, 1983.

[28] R. Hermann, On the accessibility problem in control theory, *Internat. Sympos. Nonlinear Differential Equations and Nonlinear Mechanics*, pp. 325-332, Academic Press, New York, 1963.

[29] N. Higham, *Accuracy and stability of numerical algorithms*, SIAM, Philadelphia, 1996.

[30] N. Higham, F. Tisseur, More on pseudospectra for polynomial eigenvalue problems and applications in control theory, *Linear Algebra and its Applications*, 351-352 (2002) 435-453.

[31] N. Higham, D. S. Mackey, F. Tisseur, The conditioning of linearizations of matrix polynomials, *Numerical Analysis Report 465*, Manchester Centre for Computational Mathematics, April 2005.

[32] S. Iwata, R. Shimizu, Combinatorial Analysis of Generic Matrix Pencils, Mathematical Engineering Technical Reports, METR 2004-18, March 2004. Dept. of Math. Informatics. Graduate School of Information Science and Technology. The University of Tokyo, Japan.

[33] Alan Jennings, J.J McKeown, *Matrix computations*, John Wiley and Sons, West Sussex, 1992.

[34] B. Kagstrom, A. Ruhe editors. *Matrix Pencils*, Lecture Notes in Mathematics. Proceedings of a Conference Held at Pite Havsbad, Sweden, March 22-24, 1982. Springer - Verlag, Umea, Sweden.

[35] T. Kailath, *Linear systems*, Prentice Hall Information and Systems Science Series, Thomas Kailath editor, 1980.

[36] R.E. Kalman, J.E. Bertram, *A unified approach to the theory of sampling systems.* J. Franklin Inst. 267 1959 405-436.

[37] D. Karnopp, The energetic structure of multibody dynamic systems, *J. Franklin Inst.*, 306 (1978), no. 2, 165-181.

[38] E. King - Wah Chu, Perturbation of eigenvalues for matrix polynomials via the Bauer-Fike Theorem, *SIAM J. Matrix Anal. Appl.*, Vol. 25, No. 2, pp. 551-573, 2003.

[39] G. Kron, Numerical Solution of Ordinary and Partial Differential Equations by Means of Equivalent Circuits, *J. Appl. Phys.*, 16, 172, 1945.

[40] M. Kuijper, Why do stabilizing controllers stabilize?, *Automatica*, Vol. 31, pp. 621-625, 1995.

[41] G. Lago, L.M. Benningfield, *Teoria de sistemas y circuitos*, Limusa, 1984.

[42] R. Lozano, B. Brogliato, O. Egeland, B. Maschke, *Dissipative systems analysis and control*, London, Springer - Verlag, 2000.

[43] P. Lancaster, P. Psarrakos, A note on weak and strong linearizations of regular matrix polynomials. *Numerical Analysis Report*, Manchester Center for Computational Mathematics, Manchester, England, June 2005.

[44] *Lapack, User's guide*, SIAM, 1992.

[45] C.L. Lawson et al, Basic Linear Algebra Subprograms for Fortran Usage, *ACM Transactions on Mathematical Software*, Vol. 5, No 3., September 1979, Pages 308-323.

[46] D.S. Mackey, N. Mackey, C. Mehl, V. Mehrmann, Vector spaces of linearizations for matrix polynomials. Technical Report, DFG Research Center, Technische Universität, Berlin, Germany. April 2005.

[47] D.S. Mackey, N. Mackey, C. Mehl, V. Mehrmann, Palindromic polynomial eigenvalue problems: Good vibrations from good linearizations. Technical report, DFG Rsearch Center MATHEON, Mathematics for key technologies in Berlin, TU Berlin, Str. des 17 Juni 136, D-10623 Berlin, Germany.

[48] B. Maschke, Geometrical formulation of bond graph dynamics with application to mechanisms. *Current topics in bond graph related research.* J. Franklin Inst. 328 (1991), no. 5-6, 723-740.

[49] *MATLAB/SIMULINK site: http://www.mathworks.com*

[50] *MODELICA site: http://www.modelica.org/*

[51] C. B. Moler, G. W. Stewart, An algorithm for generalized matrix eigenvalue problems, *SIAM J. Numer. Anal.*, Vol. 10, No. 2, April 1973.

[52] K. Ogata, *Ingenieria de Control Moderna*, Prentice Hall, 1980.

[53] R. Ortega, A. Loria, P.J. Nicklasson, H.S. Ramírez, *Passivity - based control of Euler - Lagrange systems*, Springer - Verlag, London, 1998.

[54] R.V. Patel, A.J. Laub, P. Van Dooren, *Numerical Linear Algebra Techniques for Systems and Control*, Edited by IEEE Press, NY, 1994.

[55] A.S. Perelson, H.M. Paynter, Discussion: The properties of bond graph junction structure matrices, *Trans. ASME Ser. G. J. Dynamic Systems, Measurement and Control*, 95 (1973), 362-367 by J. R. Ort and H. R. Martens.

[56] D. Pervouchine, Hierarchy of closures of matrix pencils, *J. Lie Theory*, Vol. 14, pp. 1-37, 2004.

[57] S.S. Petrova, Heaviside and the development of the symbolic calculus, *Arch. Hist. Exact Sci.* 37, no. 1, 1-23, 1987.

[58] J.W. Polderman, J.C. Willems, *Introduction to Mathematical Systems Theory: a Behavioral Approach*, Springer-Verlag, Berlin, 1997.

[59] *PolyX Ltd. The Polynomial Toolbox for MATLAB version 2.5.8. Prague, Chech Republic, 2003. See www.polyx.cz.*

[60] C. Praagman, *Inputs, outputs and states in the representation of time series* in A. Bensoussans, J.L. Lions (editors), Analysis and Optimization of systems, pp. 1069-1078, LNCIS, Vol. 11, 1988.

[61] C. Praagman, H.L. Trentelman, Ricardo Zavala Yoé, On the Parametrization of all Regularly Implementing Controllers, *Proceedings of the 16th IFAC World Congress on Automatic Control*, Prague, Czech Republic, July 4-8, 2005.

[62] C. Praagman, H.L. Trentelman, Ricardo Zavala Yoé, On the Parametrization of all Regularly Implementing and Stabilizing Controllers, *SIAM Journal on Control and Optimization*, Vol. 45, pp. 2035-2053, 2007.

[63] C. Praagman, H.L. Trentelman, Ricardo Zavala Yoé, Column Compression and Embedding of Polynomial Matrices, *IEEE Transactions on Automatic Control*. (Manuscript submitted in 2008).

[64] P. Rapisarda, J.C. Willems, State maps for linear systems, *SIAM J. Control Optim.*, Vol. 35, pp. 1053-1091, 1997.

[65] P. Rocha, J. Wood, Trajectory control and interconnection of 1D and nD systems, *SIAM Journal of Control and Optimization*, Vol 40, pp. 107-134, 2001.

[66] A.J. van der Schaft, Achievable behavior of general systems, *Systems and Control Letters*, Vol. 49, pp. 141-149, 2003.

[67] A.J. van der Schaft, *L2-gain and passivity techniques in nonlinear control*, Lecture notes in control and information sciences, ISSN 0170-8643; 218, Springer - Verlag, London, 1996.

[68] Issai Schur, On the characteristic roots of a linear substitution with an application to the theory of integral equations (in German). *Mathematische Annalen*, 66 (1909), pp. 488-510.

[69] *SCILAB site: http://scilabsoft.inria.fr*

[70] M. Sebek, S. Pejchova, D. Henrion, H. Kwakernaak, Numerical Methods for Zeros and Determinant of Polynomial Matrix, *Proceedings of the IEEE Mediterranean Symposium on New Directions in Control and Automation*, pp. 488-491, Chania, Crete, Greece, June 1996.

[71] Shui-Hung Hou, A simple proof of the Leverrier-Faddev characteristic polynomial algorithm, *SIAM. Rev.*, Vol. 40, No. 3, pp. 706-709, September 1998.

[72] *SLICOT site: http://www.win.tue.nl/wgs/slicot.html*

[73] F. Tisseur, Backward error and condition for polynomial eigenvalue problems, *Linear Algebra and its Applications*, 309 (2000) 339-361.

[74] F. Tisseur, K. Meerbrengen, The Quadratic Eigenvalue Problem, *SIAM Review*, Vol. 43, No. 2, pp. 236-286, 2001

[75] H.L. Trentelman, R. Zavala Yoé, C. Praagman, Polynomial embedding algorithms for controllers in a behavioral framework, *IEEE Transactions on Automatic Control*, Vol. 52, pp. 2182-2187, 2007.

[76] H.L. Trentelman, R. Zavala Yoé, C. Praagman, Polynomial embedding algorithms for stabilization and pole placement, *17th International Symposium on Mathematical Theory of Networks and Systems, (MTNS 2006)).*

[77] P. Van Dooren, The computation of Kronecker canonical form of a singular pencil, *Linear Algebra Applications*, 27 (1979), pp. 103-141.

[78] P. Van Dooren, The Basics of Developing Numerical Algorithms, *IEEE Control Systems Magazine*, (2004), pp. 18-27.

[79] P. Van Dooren, Michel Verhaegen, On the use of unitary state space transformations, *Contemporary Math., AMS Contemporary Mathematics Series*, vol. 47, pp. 447-463, 1985.

[80] M. Vidyasagar, *Input-output analysis of large-scale interconnected systems: decomposition, well-posedness and stabiblity*, Lecture Notes in control and information sciences, 29, Berlin, Springer-Verlag, 1981.

[81] M. Vidyasagar, *Control System Synthesis: A Factorization Approach*, The MIT Press, Cambridge, Massachusetts, 1985.

[82] David S. Watkins, QR-like Algorithms for Eigenvalue Problems, *Lectures in Applied Mathematics, v. 32, The Mathematics of Numerical Analysis, Ed. J. Renegar, M. Shub, S. Smale, American Mathematical Society*, 1996.

[83] P.E. Wellstead, *Physical System Modelling*, Academic Press, New York, New York, 1979.

[84] N. Wiener, *Cybernetics, or control and communication in the animal and the machine*, 2nd. ed. The M.I.T Press, Cambridge, Mass., John Wiley and Sons., Inc., New York - London, 1961.

[85] J.C. Willems, *The Analysis of Feedback Systems*, The MIT Press, Cambridge, Massachusetts, USA, 1971.

[86] J.C. Willems, System theoretic models for the analysis of physical systems, *Ricerche de Automatica*, Vol. 10 , No. 2, December 1979.

[87] J.C. Willems, Paradigms and Puzzles in the Theory of Dynamical Systems, *IEEE Transactions on Automatic Control*, Vol. 36, pp. 259-294, 1991.

[88] J.C. Willems, On interconnections, control and feedback, *IEEE Transactions on Automatic Control*, Vol. 42, pp. 326-339, 1997.

[89] J.C. Willems, Open dynamical systems, *Doc. Math. J. DMV*, Extra volume ICM 1998 - 111, pp. 697-706, 1998.

[90] J.C. Willems, M.N. Belur, A. Agung Julius, H.L. Trentelman, The canonical controller and its regularity, *Proceedings of the IEEE Conference on Decision and Control*, 2003.

[91] J.C. Willems, H.L. Trentelman, Synthesis of dissipative systems using quadratic differential forms - part I, *IEEE Transactions on Automatic Control*, Vol. 47, nr. 1, pp. 53-69, 2002.

[92] T. Wright, L. Trefethen, *Pseudospectra of rectangular matrices*, IMA Journal of Numerical Analysis (2002) 22, 501-519.

[93] D.C. Youla, H.A. Jabr, J.J. Bongiorno, Modern Wiener-Hopf design of optimal controllers, Part 2: The multivariable case, *IEEE Trans. Aut. Control*, Vol. 21, pp. 319-338, 1976.

[94] Ricardo Zavala Yoé, Fuzzy Control of Second Order Vectorial Systems: L2 Stability, *Proceedings of the 4th ECC97 (European Control Conference)*, Brussels, Belgium, July 1-4, 1997.

[95] Ricardo Zavala Yoé, C. Praagman, H.L. Trentelman, The Stable Embedding Problem, *Proceedings of the 16th IFAC World Congress on Automatic Control*, Prague, Czech Republic, July 4-8 , 2005.

[96] Ricardo Zavala Yoé, C. Praagman, H.L. Trentelman, Embedding Polynomial Matrices: A Practical Perspective, *Proceedings of the joint IEEE Conference On Decision and Control (CDC) - European Control Conference (ECC)*, Sevilla, Spain, December 12-15, 2005.

[97] Ricardo Zavala Yoé, C. Praagman, H.L. Trentelman, Inherently Numerical and Physical Issues of Polynomial Modelling and Control, *1st IFAC Workshop on Applications of Large Scale Industrial Systems*, Helsinki - Stockholm (ALSI) 2006 (submitted).

[98] Ricardo Zavala Yoé, C. Praagman, H.L. Trentelman, An Embedding Algorithm for Polynomial Matrices Tested Numerically, *17th International Symposium on Mathematical Theory of Networks and Systems, (MTNS 2006).*

[99] J.C. Zúñiga, D. Henrion, Numerical stability of block Toeplitz algorithms in polynomial matrix computations, LAAS-CNRS Research Report No. 04506, October 2004.

Summary

Professionals involved in systems and control theory and control engineering formulate equations which describe a physical system for all time. This fact is referred to as *modelling* . The better the model (the equations obtained) the better the knowledge we get from the system under study. Using these model we can modify the original system if we are not satisfied with its performance (as most of the times happens). This process is called *controlling* a system. Since the model of a system is given in terms of differential or difference equations, designing a *controller* means finding a new set of equations which "combined" in some way with the model of the given system produces a desired performance. Such a specification is established by the designer (a systems and control theorist or a control engineer).

There has been a paradigmatic point of view during the controller design stage: the cause - effect point of view (referred to as input - output approach). Although at first sight this approach might look logical, this is not always the case. It is well known that there exist systems for which this way of modelling is not suitable. We describe some of these in the book. They illustrate the necessity of having a wider point of view. Roughly speaking, such a wider point of view for modelling and control systems is offered by the so called Behavioral Approach for systems and control. Here, we consider that our systems are representation - free structures and as such we are not restricted to any other paradigm, rather we give our point of view as an alternative. We consider a system as an entity that is described by its trajectories (solutions to differential or difference equations) over time. In this way we don't need to be model structure dependent. This is an issue in our field because we can consider traditional results in our area as particular cases of our more general approach. Although this Behavioral Approach is well established, the main contribution done in this book is - roughly speaking - providing a description of *all controllers* that satisfy a certain performance. This is called a *parametrization* of all controllers.

On the other hand, since the Behavioral Approach is intended to be a tool for modelling and control of physical systems, a natural and key issue is the *numerical implementation* of the obtained models and controls from this viewpoint. Particularly, controllers are implemented via *algorithms* which end up with actual computational implementations simulating the *controlled system* (the modified system which satisfies our new requirements). This is remarkable because we might not have the real, actual physical structure at hand and one way to simulate it is by means of programs which run in a digital computer.

In this book we study a well known technique to design polynomial control algorithms

(neither implemented in the computer before nor reported about its consequences). As a matter of fact, we show that this technique does not work correctly in actual numerical implementations. In contrast, we offer improved options to do this. This way is simpler and easier to implement in the computer. We tested this result in many examples to support our claim. We consider this fact as key during this work. The algorithms were implemented in MATLAB and SCILAB. MAPLE was also useful.

Index

algorithms, 95
annihilator, 37
autonomous behavior, 27, 29

behavioral approach, 1
behavioral equations, 20

closed system, 3
column compression, 136
condition function, 121
condition number, 122
conditioning gain function, 122
conditioning of the pencil, 121
control as interconnection, 30
controllability, 24
controllable part, 29
controller behavior, 32

detectability, 23
dynamical system, 4, 19

elimination, 21
embedding, 95, 135
    stable, 95
    unimodular, 95
embedding algorithm, 135
equivalent representations, 23

full controlled behavior, 32
full interconnection, 35

generalized Schur form, 110
generic pencil, 118

hidden behavior, 33
Hurwitz, 96
hybrid representation, 22

image representation, 25
implementability, 35, 36, 59
input cardinality, 29
input/output approach, 4
input/output partition, 28
inputs, 27
interconnection, 30

kernel representation, 21
Kronecker canonical form, 101

latent variable representation, 22
latent variables, 21
linear differential systems, 19
linearization, 115

manifest behavior, 21
manifest control behavior, 32
manifest plant behavior, 31
manifest variables, 21
mathematical model, 2
matrix pencils, 101

non generic pencil, 118
numerical implementation, 115

observability, 23
open system, 4
orbit of a pencil, 117
output cardinality, 29
outputs, 27

paradigms, 2
parametrization, 35, 59
    regularly implementing controllers, 49, 71, 77
    stabilizing controllers, 53, 83

partial interconnection, 32, 59
pencil, 100, 115
pole placement, 36, 45, 59

QR decomposition, 107

regular implementability, 35, 37
representation, 20
row compression, 22
RQ decomposition, 108

stabilizability, 24
stabilization, 36, 45, 59
stable embedding problem, 95
state space representation, 103

tearing, 13, 17
thermodynamics, 2

unimodular, 22, 96, 103, 135

zooming, 13