Christou, D. (2011). ERES Methodology and Approximate Algebraic Computations. (Unpublished Doctoral thesis, City University London)


CITY UNIVERSITY LONDON

City Research Online


# ERES Methodology and Approximate Algebraic Computations

by

DIMITRIOS CHRISTOU

Thesis submitted for the award of
the degree of Doctor of Philosophy (PhD)
in Mathematical Methods and Systems

Systems and Control Centre
School of Engineering and Mathematical Sciences
City University
London EC1V 0HB

October 2011

# Contents

# List of Tables

# List of Figures

*I dedicate this thesis to my sister's son,*
*my little nephew Christos.*

# Acknowledgements

I would like to express my sincerest gratitude to my supervisor, Professor Nicos Karcanias, who supported me throughout my thesis with his friendliness, patience and careful guidance, and shared with me his invaluable knowledge and expertise in the area of Approximate Algebraic Computations. I feel very fortunate to have had the opportunity to be one of his students.

I would also like to show my gratitude to Dr. Marilena Mitrouli, who gave me the first directions to my research, and helped me to stay focused on it. It was her encouragement, persistence, understanding and kindness that I decided to apply for doctoral training, after completing my master's degree under her supervision.

I would like to thank Dr. Stavros Fatouros and Dr. Dimitrios Triantafyllou for sharing with me their knowledge and their personal work on specific topics of my thesis, and above all for the friendship they offered me, and our excellent cooperation we had over the previous time.

I also thank Prof. George Halikias and Prof. Eric Rogers for their valuable suggestions and remarks which helped me to improve the thesis.

I would also like to thank my closest friends for helping me get through the difficult times, and for all the emotional support and caring they provided.

Lastly, and most importantly, I wish to thank my parents, who raised me, loved me, and supported me, and a special thanks goes to my beloved sister, Vicky Christou, for her love and encouragement to complete my doctoral studies.

*Dimitrios Ch. Christou*
*London, October 2011*

# Declaration

The University Librarian of the City University may allow this thesis to be copied in whole or in part without any reference to the author. This permission covers only single copies, made for study purposes, subject to normal conditions of acknowledgements.

# Abstract

The area of approximate algebraic computations is a fast growing area in modern computer algebra which has attracted many researchers in recent years. Amongst the various algebraic computations, the computation of the Greatest Common Divisor (GCD) and the Least Common Multiple (LCM) of a set of polynomials are challenging problems that arise from several applications in applied mathematics and engineering. Several methods have been proposed for the computation of the GCD of polynomials using tools and notions either from linear algebra or linear systems theory. Amongst these, a matrix-based method which relies on the properties of the GCD as an invariant of the original set of polynomials under elementary row transformations and shifting elements in the rows of a matrix, shows interesting properties in relation to the problem of the GCD of sets of many polynomials. These transformations are referred to as Extended-Row-Equivalence and Shifting (ERES) operations and their iterative application to a basis matrix, which is formed directly from the coefficients of the given polynomials, formulates the ERES method for the computation of the GCD of polynomials and establishes the basic principles of the ERES methodology.

The main objective of the present thesis concerns the improvement of the ERES methodology and its use for the efficient computation of the GCD and LCM of sets of several univariate polynomials with parameter uncertainty, as well as the extension of its application to other related algebraic problems.

New theoretical and numerical properties of the ERES method are defined in this thesis by introducing the matrix representation of the Shifting operation, which is used to change the position of the elements in the rows of a matrix. This important theoretical result opens the way for a new algebraic representation of the GCD of a set polynomials, the remainder, and the quotient of Euclid's division for two polynomials based on ERES operations. The principles of the ERES methodology provide the means to develop numerical algorithms for the GCD and LCM of polynomials that inherently have the potential to efficiently work with sets of several polynomials with inexactly known coefficients. The present new implementation of the ERES method, referred to as the "Hybrid ERES Algorithm", is based on the effective combination of symbolic-numeric arithmetic (hybrid arithmetic) and shows interesting computational properties concerning the approximate GCD and LCM problems. The evaluation of the quality, or "strength", of an approximate GCD is equivalent to an evaluation of a distance problem in a projective space and it is thus reduced to an optimisation problem. An efficient implementation of an algorithm computing the strength bounds is introduced here by exploiting some of the special aspects of the respective distance problem. Furthermore, a new ERES-based method has been developed for the approximate LCM which involves a least-squares minimisation process, applied to a matrix which is formed from the remainders of Euclid's division by ERES operations. The residual from the least-squares process characterises the quality of the obtained approximate LCM. Finally, the developed framework of the ERES methodology is also applied to the representation of continued fractions to improve the stability criterion for linear systems based on the Routh-Hurwitz test.

«Δύο αριθμῶν ἀνίσων ἐκκειμένων, ἀνθυφαιρομένου δέ ἀεὶ τοῦ ἐλάσσονος ἀπὸ τοῦ μείζονος, ἐὰν ὁ λειπόμενος μηδέποτε καταμετρῇ τὸν πρὸ ἑαυτοῦ, ἕως οὗ λειφθῇ μονάς, οἱ ἐξ ἀρχῆς πρῶτοι πρὸς ἀλλήλους ἔσονται.»

*Two unequal numbers being set out, and the less being continually subtracted in turn from the greater, if the number which is left never measures the one before it until an unit is left, the original numbers will be prime to one another.*

*Antenaresis (Euclidean Algorithm)*

*Euclid's Book of Elements*
*Book VII, Proposition I*

# Chapter 1

# Introduction

Algebraic and geometric invariants are instrumental in describing system properties and characterising the solvability of Control Theory problems [38, 65, 82]. These invariants are defined on rational, polynomial matrices and matrix pencils under different transformation groups (coordinate, compensation, feedback type) and their computation relies on algebraic algorithms, whereas symbolic tools are used for their implementation.

The different type of invariants and system properties defined on a family of linear system models may be classified to those which are *generic* and those which are *nongeneric* [46, 82]. Notions such as multivariable zeros of nonsquare systems and decoupling zeros [38] are nongeneric, whereas for square systems, the notion of zeros is generic. Notions such as minimal indices (of various types), are always defined, but they have certain generic values. In dealing with engineering system models, on the one hand the uncertainty about the true value of the parameters, and on the other hand the rounding off computational errors, makes the computation of nongeneric values of invariants a difficult task.

Engineering models are not exact and they are always characterised by parameter uncertainty. This introduces some considerable problems with any framework based on exact symbolic tools, given that the underlined models are always characterised by parameter uncertainty. The central challenge is the transformation of algebraic notions to an appropriate analytic setup within which meaningful *approximate* solutions to exact algebraic problems may be sought. This motivates the need for transforming the algebraic problems into equivalent linear algebra problems and then develop approximate algebraic computations, which are appropriate for the case of computations on models characterised by parameter uncertainty.

Computing, or evaluating nongeneric types, or values of invariants and thus associated system properties on models with numerical inaccuracies is crucial for applications. For such cases, symbolic tools fail, since they *almost always* lead to a generic solution, which does not represent the *approximate presence* of

the value property on the set of models under consideration. The formulation of a methodology for robust computation of nongeneric algebraic invariants, or nongeneric values of generic ones, has as prerequisites:

a) The development of a numerical linear algebra characterisation of the invariants, which may allow the measurement of degree of presence of the property on every point of the parameter set.

b) The development of special numerical tools, which avoid the introduction of additional errors.

c) The formulation of appropriate criteria which allow the termination of algorithms at certain steps and the definition of meaningful approximate solutions to the algebraic computation problem.

It is clear that the formulation of the algebraic problem as an equivalent numerical linear algebra problem, is essential in transforming concepts of an algebraic nature to equivalent concepts of an analytic character and thus set up the right framework for approximations. This property is referred to as *numerical reducibility* (NR) of the algebraic computation and it depends on the nature of the particular invariant.

Given that any set of engineering data has a given numerical accuracy it is clear that there is no point in trying to compute with greater accuracy than that of the original data and thus, an approximate solution has to be sought at some stage, before the procedure converges to some meaningless generic value. In fact, engineering computations are defined not on a single model of a system $S$, but on a ball of system models $\Sigma(S_0, r(\varepsilon))$, where $S_0$ is a nominal system and $r(\varepsilon)$ is some radius defined by the data error order $\varepsilon$. The result of computations has thus to be representative for the family $\Sigma(S_0, r(\varepsilon))$ and not just the particular element of this family. From this viewpoint, symbolic computations carried out on an element of the $\Sigma(S_0, r(\varepsilon))$ family may lead to results, which do reveal the desired properties of the family. Numerical computations have to stop, when we reach the original data accuracy and an approximate solution to the computational task has to be given.

### ▶ Nongeneric Computations

Numerical computations dealing with the derivation of an approximate value of a property, function, which is nongeneric on a given model set, will be called *nongeneric computations* (NG). If the value of a function always exists on every element of the model set and depends continuously on the model parameters, then the computations leading to the determination of such values will be called *normal numerical* (NN). Computational procedures aiming at defining the generic value of a property, function on a given model set (if such values exist), will be

called *generic* (GC). For instance, on a set of polynomials with coefficients taking values from a certain parameter set, the greatest common divisor (GCD) has in general the trivial value, equal to 1, and the existence of a nontrivial GCD is a nongeneric computation. On the contrary, the existence of the least common multiple (LCM) is considered generic, since it always exists, given by the product of all the polynomials of the set. Numerical procedures that aim to produce an approximate nontrivial value by exploring the numerical properties of the parameter set are typical examples of NG computations. The computation of nongeneric invariants is linked to nongeneric computations [46].

A number of important invariants for linear systems rely on the notion of the greatest common divisor of several polynomials. The link between control theory and the GCD problem is very strong; in fact, the GCD is instrumental in defining system notions such as zeros, decoupling zeros, zeros at infinity, notions of minimality of system representations and others. On the other hand, systems and control methods provide concepts and tools which enable the development of new computational procedures for GCD. An integral part of the derivation of the procedures for nongeneric computations is the relaxation of certain algebraic definitions, and their embedding in an analytical setup. Appropriate tools have to be devised to indicate degree of presence, or distance from strong possession of a certain property. In such cases, for example the computation of the GCD or the zeros of nonsquare systems, the attention has to be focused on the appropriate termination of the computational algorithm that will allow the estimation of the approximate solutions. The accuracy of the original data determines the threshold, where an algorithm has to terminate and give an approximate solution and where it has to continue.

A major challenge for the control theoretic applications of the GCD is that frequently we have to deal with a very large number of polynomials with inexactly known coefficients coming from real-time applications. It is this requirement that makes the pairwise type approaches for GCD [11, 12, 53, 61, 84] not suitable for such applications. The GCD related work described in this research goes back to the attempt to introduce the notion of *almost zeros* of a set of polynomials [43] and study the properties of such zeros from the feedback viewpoint. This work was subsequently developed to a methodology for computing the approximate GCD of polynomials using numerical linear algebra methods, such as the ERES [57] and Matrix Pencil methods [45]. The results in this area of computations are important in the development of meaningful solutions to algebraic system theory problems for models characterised by parameter uncertainty and they are linked to a large range of related problems, such as almost non-coprimeness and solutions of polynomial Diophantine equations, and approximate factorisation of rational transfer function models.

The definition of the *approximate GCD* can be considered as a distance problem in a projective space. The distance framework given for the approximate GCD [22, 42] provides the means for computing optimal solutions, as well as evaluating the strength of ad-hoc approximations derived from different algorithms.

Various methods and algorithms for the computation of the approximate GCD of polynomials have been proposed so far. Many of them rely on Euclid's algorithm, which is the oldest well-known method for computing the GCD of integer numbers [11, 12, 61, 69, 70]. Other newer methods make use of subresultant matrices [5, 17, 20, 22, 73], perform optimizations and quadratic programming [16, 17, 52], use Padé approximations and approximations of polynomial zeros [63] or matrix pencils [45, 49]. Such algorithms usually consist of several different algebraic procedures with a specific task. These algebraic procedures have to be organised properly either in sequential or in iterative way in order to produce the best possible results for the problem that the algorithm is designed to solve. However, the implementation of such algorithms in an appropriate programming environment is not trivial and requires careful selection of data structures and arithmetic system.

### ▶ Hybrid Computations

In conventional computer algebra, the usual aim is to perform algebraic computation exactly using rational number arithmetic and the introduction of algebraic and transcendental numbers. But many problems coming from areas like computer vision, robotics, computational biology, physics *etc*, are described with inexact numbers ("empirical" numbers) as the input parameters or coefficients. In this context, the usual exact algorithms of computer algebra may not be easily applicable, or may be inefficient. Recent years have witnessed the emergence of new research combining symbolic and numeric computations and leading to new kinds of algorithms, involving algebraic computations with approximate numeric arithmetic, such as floating-point number arithmetic. This combination gives a different perspective in the way to implement an algorithm and introduces the notion of *hybrid computations*.

*Hybridity* refers in its most basic sense to mixture; a mixture of different ways, components, methods *etc*, which can produce the same or similar results. The basic idea of making something "hybrid" is to improve on its characteristics and therefore make it work better. In our case, we focus on the mixture of symbolic arithmetic and numeric arithmetic, which will be referred to as *hybrid arithmetic*.

In a hybrid arithmetic system both exact symbolic and numeric finite precision arithmetic operations can be carried out simultaneously. *Symbolic computations* refer to arithmetic operations either with arbitrary variables or fractions of integers to represent the numerical input data. The symbolic computations

which involve only numerical data in rational format, are also referred to as *rational computations* and they are always performed in almost infinite accuracy, depending on the symbolic kernel of the programming environment. On the other hand, *numerical computations* refer to arithmetic operations with numbers in floating-point format (decimal numbers). However, the accuracy of the performed numerical computations is limited to a specific number of decimal digits which gives rise to numerical rounding errors that often cause serious complications and must be avoided [81]. Therefore, the different algebraic procedures, which form an algorithm, can be implemented independently either using symbolic computations or numerical computations. Such kind of implementation will be referred to as *hybrid implementation* and hence, the algorithm that is implemented by using symbolic-numeric computations, i.e. hybrid computations, will be called a *hybrid algorithm*.

The hybridisation of an algorithm (i.e. the hybrid implementation of an algorithm) is possible in software programming environments with symbolic-numeric arithmetic capabilities such as Maple, Mathematica, Matlab and others which involve an efficient combination of symbolic (rational) and numerical (floating-point) operations. However, the effective combination of symbolic and numerical operations depends on the nature of an algebraic method and the proper handling of the input data either as rational or floating-point numbers.

Using hybrid computations is basically a trade-off between accuracy and processing time. Symbolic processing always produce exact results and thus it is often used to improve on the conditioning of the input data, or to handle a numerically ill-conditioned subproblem. However, symbolic computations can be very demanding in respect of computational time and data storage, especially in case of large amounts of data. On the other hand, numerical processing is faster and, generally, consumes less computer memory, but the accuracy of the results may not be satisfactory. Therefore, numerical computations are preferable in accelerating certain parts of an algorithm, or in computing approximate outputs. These remarks can be considered as rough guidelines in designing and implementing a hybrid algorithm, but it is not clear how to develop *the* hybrid algorithm which is capable of giving the best possible results to the problem that is expected to handle. In practise, an effective hybridisation must lead to an algorithm which is fast and accurate (depending on the accuracy of the input data). Therefore, the amount of initial data, the structure of algebraic procedures (sequential or iterative), and the desired level of accuracy are very important factors to be taken into account for the development of an effective hybrid algorithm.

Concerning GCD algorithms and hybrid implementation, not all of them are suitable to be implemented in a symbolic-numeric computational environment. As mentioned above, the symbolic manipulation of data guarantees an error-free

solution, but exact symbolic operations are very expensive regarding computational time and computer memory. This problem is more obvious in matrix-based methods, especially in resultant based methods when the processed matrix is large and dense. On the other hand, numerical finite precision operations are fast and more preferable when an approximate solution is sought. Yet again, the accumulation of numerical errors, especially in iterative methods, can be awfully disastrous.

Amongst the various methods for the computation of an approximate GCD, the ERES [40, 57] is a matrix-based method that can handle sets of several polynomials and allows the development of an effective hybrid algorithm. The ERES is based on the invariance of the GCD under elementary row transformations and involves some basic algebraic procedures, such as Gaussian elimination with partial pivoting and Singular Value Decomposition, which can be implemented separately by using either exact symbolic operations or numerical floating-point operations combined in an optimal setup. The simple structure, the iterative nature and the ability to manipulate large amounts of data are advantages that put the ERES method at the centre of our study for the approximate GCD problem.

### ▶ Objectives

The main objectives of this thesis are:

1. To use the basic principles of the ERES method [40, 57] for defining approximate solutions to the GCD problem by developing the hybrid implementation of this method.

2. To use the recently developed framework for defining the *approximate notions* for the GCD as a distance problem in a projective space [42] to develop an optimization algorithm for evaluating the strength of different ad-hoc approximations derived from different algorithms.

3. To improve the context of the ERES methodology and extend its use to other related problems such as the computation of the approximate LCM of sets of polynomials, the evaluation of stability of linear systems and the representation of continued fractions.

The fundamental problems, which relate to the main objectives, are the algebraic representation of the GCD of several polynomials in terms of ERES operations and the investigation of the related problems, such as the matrix representation of the Shifting operation and the matrix representation of the remainder of Euclid's division algorithm.

Chapter 2 provides an overview to methods of algebraic computations especially developed to deal with the approximate GCD and approximate LCM problems

and serves as a motivator for the algebraic computational problems involving rational expressions, which are considered in the thesis.

Chapter 3 provides a theoretical presentation of the ERES method. This involves the theoretical issues of the processes involved, such as the selection of an appropriate basis matrix and the application of elementary row transformations, and shifting. The Shifting operation, applied to a matrix, is a key element in the algebraic representation of the whole ERES method. The presented theoretical algebraic procedure of representing the Shifting operation as a matrix product relies on the rank properties of the processed matrix and binds together the iterative steps of the method. This allows the formulation of a new matrix representation for the GCD of a set of several polynomials. The developed framework of the Shifting operation is also used to obtain an algebraic expression for the remainder of the division of two polynomials which evidently establishes a new procedure of polynomial division by using ERES operations.

In chapter 4, the numerical implementation of the ERES method in a symbolic-numeric programming environment is presented. The new ERES algorithm, referred to as Hybrid ERES algorithm, combines in an optimal setup the symbolical application of rows transformations and shifting, and the numerical computation of an appropriate termination criterion, which can provide the required approximate solutions. The termination criterion of the algorithm relies on the partial singular value decomposition method [75, 76]. A new variation of this method is specially developed for the Hybrid ERES algorithm, resulting in a dramatical improvement of its computational performance in the case of large sets of polynomials. The concept behind this method is that, in general, the ERES algorithm terminates when a matrix with rank equal to 1 is obtained. Thus, only the unique singular value and its right singular vector are necessary to be computed. The numerical behaviour of the Hybrid ERES algorithm is also discussed.

Chapter 5 starts with an overview of the fundamentals of the approximate GCD evaluation framework [21, 22, 42]. For sets of polynomials for a given number of elements and with fixed the two maximal degrees, a point in the projective space is defined, based on the coefficients of the polynomials in the set. The family of all sets, which have a GCD with a given degree, is defined by the properties of the generalised resultant and it is shown to be a special variety of the projective space referred to as the $d$-GCD variety. The factorisation of the resultant has allowed the definition of any $d$-degree approximate GCD as a subvariety of the $d$-GCD variety. Thus, the *strength* of the approximation, provided by the result of a given numerical method, may be completed as the evaluation of the distance of the given point (set of polynomials) from its subvariety of the $d$-gcd variety. This distance is worked out as the solution of a simple optimisation problem. The definition of the best $d$-degree approximate solution is equivalent to a

computation of the distance of the given set from the $d$-GCD variety. This distance is computed by minimising the Frobenius norm of the resultant characterising the dynamic perturbations. The numerical properties of this minimization problem are considered here from a different point of view. Such optimization problems are often non-convex and a reliable solution is not guaranteed. Alternatively, useful information can be obtained by computing some tight bounds for the strength. An algorithm for computing the *strength bounds* is presented in this chapter. Its main characteristic is that it exploits the properties of resultant matrices in order to produce meaningful results without using optimisation routines and allows the computation of an *average strength*. These bounds work as indicators, which characterise the quality of a given approximate GCD. The combination of the Hybrid ERES algorithm and the algorithm of strength bounds suggests a complete procedure for the computation and evaluation of an approximate GCD of a set of several polynomials.

The main objective in chapter 6 is to investigate the problem of defining a numerical procedure for the computation of the LCM of a set of several polynomials avoiding root finding and GCD computation. The developed methodologies depend on the proper transformation of the LCM computations to real matrix computations and thus also introduce a notion of *approximate* LCM when working on data with numerical inaccuracies. It is the aim of this chapter to give an alternative new way to compute the LCM of a set of several polynomials based on the ERES method. Two approaches are discussed. The first approach aims at the reduction of the computation of the LCM to an equivalent problem where the computation of GCD is an important part [47], and the second refers to the direct use of Euclid's division algorithm by ERES operations where there is no need to compute the GCD and the LCM is finally computed by solving an appropriate least-squares problem. The developed algorithms, which are based on these two methods for the computation of the LCM of several polynomials, are implemented in a symbolic-numeric computational environment and their numerical complexity and performance is analysed.

In chapter 7 the developed framework of the ERES methodology is considered for a new approach to evaluate the stability of a linear time-invariant system from its characteristic polynomial (Routh-Hurwitz test). This new approach is based on the ERES methodology in order to form a matrix-based criterion, according to a related continued fraction representation and the Routh-Hurwitz theorem [24]. Furthermore, the problem of finding the minimum distance of a stable polynomial from instability as well as the minimum norm stabilisation is considered. The developed algorithm provides simpler expressions of the terms in the Routh-Hurwitz stability test and simplifies the conditions of the addressed minimisation problem.

Finally, chapter 8 summarises the achievements and describes issues related to future research. Furthermore, the algorithms, which are presented in this thesis, are implemented in the software computational environment of Maple by using Maple's programming code and they are listed in the Appendix A.

## 1.1 Notation

In the following, $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$ and $\mathbb{C}$ denote the sets (fields) of natural, integer, rational, real and complex numbers, respectively. The imaginary unit in the set of complex numbers is denoted by `i`. $\mathbb{R}[s]$ denotes the ring of polynomials in one variable over $\mathbb{R}$. Capital letters denote matrices and small underlined letters denote vectors. Capital letters followed by a variable $s \in \mathbb{R}$ denote rational functions of $s$ or polynomial matrices. Small letters followed by a variable $s \in \mathbb{R}$ denote real polynomials of $s$. The following list includes the basic notations that are used in the document.

| | |
|---|---|
| $A \in \mathbb{R}^{\mu \times \nu}$ | Matrix $A$ with elements from $\mathbb{R}$ arranged in $\mu$ rows and $\nu$ columns ($\mu, \nu \in \mathbb{N}$ and $\mu, \nu \geq 2$). |
| $\underline{v} \in \mathbb{R}^{\mu}$ | Column vector with $\mu \geq 2$ elements from $\mathbb{R}$. |
| $A^t$ | Transpose matrix of $A$. |
| $\underline{v}^t$ | Transpose vector of $\underline{v}$. |
| $\rho(A)$ or $rank(A)$ | The rank of a matrix $A$. |
| $\det(A)$ | The determinant of a square matrix $A$, $(\mu = \nu)$. |
| $tr(A)$ | The trace of a square matrix $A$, $(\mu = \nu) : tr(A) = \sum_{1=1}^{\nu} a_{ii}$ |
| $p(s) \in \mathbb{R}[s]$ | A polynomial in one variable $s$ and coefficients in $\mathbb{R}$. |
| $\deg\{p(s)\}$ | The degree of a polynomial $p(s)$. |
| $\|\underline{v}\|_2$ | The Euclidean norm of $\underline{v} : \|\underline{v}\|_2 = \sqrt{\sum_{i=1}^{\mu} \|v_i\|^2}$ |
| $\|A\|_2$ | The Euclidean norm of $A : \|A\|_2 = \sqrt{\text{max eigenvalue of } A^t A}$ |
| $\|A\|_F$ | The Frobenius norm of $A : \|A\|_F = \sqrt{\sum_{j=1}^{\mu} \sum_{i=1}^{\nu} \|a_{ij}\|^2}$ |

| | |
|---|---|
| $\|A\|_\infty$ | The infinity norm of $A$ : $\|A\|_\infty = \max_{1\leq i\leq \mu} \sum_{j=1}^{\nu} \|a_{ij}\|$ |
| $\dim\{\ \}$ | The dimension of a vector space. |
| $O(k)$ | The highest term in the value equal to $O(k)$ is of order $k$. |
| $\triangleq$ | Mathematical operator which denotes equality by definition. |
| $:=$ | Mathematical operator which denotes equality by input (particularly used in algorithms). |
| $\approx$ | Mathematical operator which denotes approximate equality. |
| $\mathbf{u}$ | The machine's precision (hardware precision). For a t-digit arithmetic system $\mathbf{u} \approx 0.5 \cdot 10^{1-t}$. |

We are mainly concerned here with sets of $m$ polynomials ($m \in \mathbb{N},\ m \geq 2$) in one variable (univariate polynomials) and coefficients in $\mathbb{R}$, denoted by

$$\mathcal{P}_{m,n} = \Big\{ p_i(s) \in \mathbb{R}[s],\ i = 1, 2, \ldots, m \text{ with } n = \max_i \left(\deg\{p_i(s)\} \geq 1\right) \Big\} \quad (1.1)$$

Whenever we want to denote the number of elements and the maximal degree of a polynomial set we shall use the notation (1.1). Otherwise the set of polynomials will be abbreviated as $\mathcal{P}$. In the special case where we want to denote that the given set of polynomials $\mathcal{P}_{m,n}$ has at least one monic polynomial with maximum degree equal to $n$, we shall use the notation $\mathcal{P}_{h+1,n}$ , (i.e. $m = h+1$). The greatest common divisor and the least common multiple of the set $\mathcal{P}_{m,n}$ will be denoted as $\gcd\{\mathcal{P}\}$ and $\mathrm{lcm}\{\mathcal{P}\}$, respectively.

# Chapter 2

# Principles and methods of algebraic computations

## 2.1 Introduction

The area of *approximate algebraic computations* is a fast growing area which has attracted the interest of many researchers in recent years. Two well known problems of algebraic computations are the computation of the Greatest Common Divisor (GCD) and the computation of the Least Common Multiple (LCM) of sets of polynomials. Both of them have widespread applications in several branches of control theory, matrix theory or network theory.

A number of important invariants for linear systems rely on the notion of GCD of many polynomials and, in fact, the GCD is instrumental in defining system notions such as zeros, decoupling zeros, zeros at infinity, notions of minimality of system representations *etc.* On the other hand, systems and control methods provide concepts and tools which enable the development of new computational procedures for the GCD. The GCD and LCM problems are naturally interlinked [46], but they are of different nature. From the applications in control theory viewpoint, the GCD is linked with the characterisation of zeros of representation whereas LCM is connected with the derivation of minimal representations of rational models. The existence of a common divisor or a common factor of polynomials is a property that holds for specific sets and it is not true generically. For randomly selected polynomials, the existence of a nontrivial GCD is a nongeneric property [60], but the corresponding LCM always exists. Therefore, extra care is needed in the development of efficient numerical algorithms calculating correctly the required GCD and LCM.

In the present chapter, we present basic concepts and tools from numerical linear algebra and linear systems theory which set the theoretical background for the study and development of GCD and LCM methods. A number of existing methods, developed for the numerical computation of the GCD or LCM of real

univariate polynomials in a finite precision arithmetic system, are summarised and the fundamental problems related to the present research are considered.

## 2.2 Fundamental concepts and definitions

The most basic concept in our study is the *polynomial*. In simple terms, a polynomial is an algebraic expression of finite length constructed from variables and constants (also known as coefficients), using only the operations of addition, subtraction, multiplication, and non-negative integer exponents. A polynomial of the form:

$$a(s) = a_n\, s^n + a_{n-1}\, s^{n-1} + \ldots + a_1\, s + a_0 \tag{2.1}$$

with $n \in \mathbb{N}$ and $a_0, a_1, \ldots, a_n \in \mathbb{F}$, is a *polynomial in one variable (univariate) with coefficients in* $\mathbb{F}$, where $\mathbb{F}$ can be one of the common fields $\mathbb{R}$, $\mathbb{Z}$ or $\mathbb{Q}$. The maximum exponent $n$ for which $a_n \neq 0$ is called the *degree* of the polynomial and is denoted by $\deg\{a(s)\}$. If $n = 0$, then $a(s)$ is a *constant* polynomial. If $a_n = 1$ the polynomial $a(s)$ is called *monic*. The set of all univariate polynomials with coefficients in $\mathbb{F}$ together with the two basic operations of addition and multiplication forms the *polynomial ring* $\mathbb{F}[s]$.

In algebra of polynomials, one major property is *divisibility* among polynomials. If $a(s)$ and $b(s)$ are polynomials in $\mathbb{F}[s]$, it is said that $a(s)$ divides $b(s)$ or $a(s)$ is a divisor of $b(s)$ and we write $a(s)|b(s)$, if there exists a polynomial $q(s)$ in $\mathbb{F}[s]$ such that:

$$a(s) \cdot q(s) = b(s) \tag{2.2}$$

Every element in $\mathbb{F}$ that zeros a polynomial is called a *root* (or *zero*). It is easy to show that every root gives rise to a linear divisor, i.e. if $a(s) \in \mathbb{F}[s]$ and $c \in \mathbb{F}$ such that $a(c) = 0$, then the polynomial $q(s) = s - c$ divides $a(s)$.

Those polynomials which cannot be factorised into the product of two non constant polynomials are called *prime polynomials*, or *irreducible polynomials*. However, any polynomial may be factorised into the product of a constant by a product of irreducible polynomials.

The *greatest common divisor* (GCD) of $a(s)$ and $b(s)$ is a monic polynomial $g(s) \triangleq \gcd\{a, b\}$ of highest degree such that $g(s)$ is a divisor of $a(s)$ and of $b(s)$, whilst the *least common multiple* (LCM) of $a(s)$ and $b(s)$ is a polynomial $l(s) \triangleq \mathrm{lcm}\{a, b\}$ of lowest degree such that both $a(s)$ and $b(s)$ divide $l(s)$. The next equation describes the association between GCD and LCM of two polynomials:

$$a(s) \cdot b(s) = g(s) \cdot l(s) \tag{2.3}$$

If $\gcd\{a, b\} = 1$, the polynomials $a(s)$ and $b(s)$ are *relative prime* (or *coprime*).

Consequently, if $a(s)$ and $b(s)$ are coprime, then

$$\text{lcm}\{a, b\} = a(s) \cdot b(s)$$

**Theorem 2.1** ([35]). *If $\mathbb{F}$ is a field and $a(s)$ and $b(s)$ are polynomials in $\mathbb{F}[s]$ with $b(s) \neq 0$, then there exist unique polynomials $q(s), r(s) \in \mathbb{F}[s]$ with*

$$\deg\{r(s)\} < \deg\{q(s)\} < \deg\{a(s)\}$$

*such that*

$$a(s) = b(s) \cdot q(s) + r(s) \tag{2.4}$$

The above theorem refers to *Euclidean division*, which is also known as *polynomial long division*, and shows that the ring $\mathbb{F}[s]$ is a Euclidean domain [35]. The polynomial $q(s)$ is called the *quotient* and $r(s)$ is the *remainder* of the division, whilst $a(s)$ is the *dividend* and $b(s)$ is the *divisor*. Euclid's division algorithm (or *Euclidean algorithm*) is an effective iterative procedure for computing the GCD of a pair of polynomials $\{a(s), b(s)\}$, based on the identity (2.4).

**The Euclidean algorithm**

1. Set $i := 1$. Let $a^{(1)}(s) := a(s)$ and $b^{(1)}(s) := b(s)$.
2. Use the identity (2.4) and find polynomials $q^{(i)}(s)$, $r^{(i)}(s)$ with $\deg\{r^{(i)}(s)\} < \deg\{b^{(i)}\}$ such that $a^{(i)}(s) = b^{(i)}(s) \cdot q^{(i)}(s) + r^{(i)}(s)$.
3. If $r^{(i)}(s) = 0$ then stop; $b^{(i)}(s)$ is a greatest common divisor.
4. If $r^{(i)}(s) \neq 0$ then replace $a^{(i)}(s)$ by $b^{(i)}(s)$ and $b^{(i)}(s)$ by $r^{(i)}(s)$. Set $i := i + 1$ and go to step 2.

When the GCD is known, the LCM can be determined from the identity (2.3).

*REMARK* 2.1. In the following we assume that $\mathbb{F} := \mathbb{R}$ and a polynomial of the form (2.1) with coefficients in $\mathbb{R}$ will be referred to as a *real polynomial*.

**► Representation of polynomials**

A real polynomial $a(s)$ may also be represented in vector form as:

$$a(s) = [a_0, a_1, \ldots, a_{n-1}, a_n] \cdot \underline{e}_n(s) \tag{2.5}$$

where $\underline{a} = [a_0, a_1, \ldots, a_{n-1}, a_n]^t \in \mathbb{R}^{n+1}$ is a *vector representative* of the polynomial $a(s)$ and $\underline{e}_n(s) = [1, s, \ldots, s^{n-1}, s^n]^t$. Equivalently, $a(s)$ can be represented as:

$$a(s) = [a_n, a_{n-1}, \ldots, a_1, a_0] \cdot \underline{e}'_n(s) \tag{2.6}$$

where $\underline{a} = [a_n, a_{n-1}, \ldots, a_1, a_0]^t \in \mathbb{R}^{n+1}$ and $\underline{e}'_n(s) = [s^n, s^{n-1}, \ldots, s, 1]^t$.
However, in most GCD methods the representation of a real polynomial relies on square *Toeplitz* matrices or *companion* matrices which provide the means to formulate a representation in matrix terms of the standard factorization of the GCD of a set of polynomials [22, 59].

**Toeplitz matrix.** The Toeplitz matrix of order $n$ associated to the polynomial $a(s)$ of degree $n$ is the $(n+1) \times (n+1)$ matrix of the form:

$$T_a = \begin{bmatrix} a_0 & 0 & 0 & \ldots & 0 \\ a_1 & a_0 & 0 & \ldots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & \ldots & a_0 & 0 \\ a_n & a_{n-1} & \ldots & a_1 & a_0 \end{bmatrix}$$

**Companion matrix.** Suppose $a(s)$ is a monic polynomial (i.e. $a_n = 1$). The companion matrix associated to the monic polynomial $a(s)$ of degree $n$ is the $n \times n$ matrix of the form:

$$C_a = \begin{bmatrix} 0 & 0 & \ldots & 0 & -a_0 \\ 1 & 0 & \ldots & 0 & -a_1 \\ 0 & 1 & \ldots & 0 & -a_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & -a_{n-1} \end{bmatrix}$$

In some cases, the transpose of $C_a$ can be also considered as companion matrix of $a(s)$.

### ▶ Representation of sets of polynomials

We consider now a set of several real polynomials of the form:

$$\mathcal{P}_{m,n} = \left\{ p_i(s) \in \mathbb{R}[s],\ i = 1, 2, \ldots, m \text{ with } n = \max_i \left(\deg\{p_i(s)\}\right) \geq 1 \right\} \quad (2.7)$$

where $p_i(s) = a_{i,0} + a_{i,1}\, s + \ldots + a_{i,n-1}\, s^{n-1} + a_{i,n}\, s^n$, and $a_{i,n} \neq 0$. Each polynomial $p_i(s)$ has a vector representative of the form:

$$\underline{p}_i = [a_{i,0}, a_{i,1}, \ldots, a_{i,n-1}, a_{i,n}]^t \in \mathbb{R}^{n+1}\ , \quad i = 1, 2, \ldots, m$$

and therefore the set $\mathcal{P}_{m,n}$ may be associated with a vector set:

$$\overline{\mathcal{P}}_{m,n} = \left\{ \underline{p}_i \in \mathbb{R}^{n+1},\ i = 1, 2, \ldots, m \right\}$$

**Definition 2.1.** A *basis* (or *base*) of a vector set $\mathcal{V}$ is a set $\mathcal{B} \subseteq \mathcal{V}$ of linearly independent vectors that, in a linear combination, can represent every vector of $\mathcal{V}$.

In general, a vector set may have several different bases and there are several different algebraic methods which determine various types of bases for a given set of vectors (or a vector space) [18, 35]. Finding an appropriate basis for the vector set $\overline{\mathcal{P}}_{m,n}$ is an important issue that affects the performance of a GCD or LCM computational method.

The vector set $\overline{\mathcal{P}}_{m,n}$ has a direct matrix representation of the form:

$$P_m = \left[\underline{p}_1, \underline{p}_2, \ldots, \underline{p}_m\right]^t = \begin{bmatrix} a_{1,0} & \ldots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,0} & \ldots & a_{m,n} \end{bmatrix} \in \mathbb{R}^{m\times(n+1)}$$

A polynomial vector $\underline{p}(s) = [p_1(s), p_1(s), \ldots, p_m(s)]^t$ may always be associated with the set $\mathcal{P}_{m,n}$ and this vector can be written in the form:

$$\underline{p}(s) = P_m \cdot \underline{e}_n(s)$$

For the set $\mathcal{P}_{m,n}$ the polynomial vector $\underline{p}(s)$ is a vector representative and the matrix $P_m$ will be called the *direct basis matrix* of the polynomial set $\mathcal{P}_{m,n}$, which is formed directly from the coefficients of the polynomials without transformations. (Throughout this thesis, $P_m$ will simply be referred to as "basis matrix".)

The formulation of an appropriate matrix for the representation of a given a set of polynomials $\mathcal{P}_{m,n}$ is crucial for the development of an efficient matrix-based method for computing the GCD or LCM of the set. A broad class of GCD methods relies on matrices with special structure. Sylvester and Bézout matrices are the most common types of matrices which are used in several GCD methods, where procedures for the computation of the rank and nullity of these matrices are essential parts.

**Definition 2.2.** Let $A$ be an $m \times n$ real matrix.

i) The subspace spanned by the row vectors of $A$ is called the *row space* of $A$. The subspace spanned by the column vectors of $A$ is called the *column space* of $A$.

ii) The *rank* of $A$, denoted by $\rho(A)$, is the dimension of the column space of $A$. The matrix is said to have *full rank*, if $\rho(A) = \min\{m, n\}$. Otherwise it is *rank deficient*. A square matrix $A_n \in \mathbb{R}^{n\times n}$ is *nonsingular*, if $\rho(A_n) = n$. Otherwise it is *singular*.

iii) The space $\mathcal{N}_r(A) = \{\underline{v} \in \mathbb{R}^n : A\,\underline{v} = 0\}$ is called the *right nullspace* of $A$. The dimension of $\mathcal{N}_r(A)$ is called the *nullity* of $A$ and is denoted by $n(A)$. It

holds $\rho(A) + n(A) = n$. Similarly, the space $\mathcal{N}_l(A) = \{\underline{u} \in \mathbb{R}^m : \underline{u}^t A = 0\}$ is called the *left nullspace* of $A$.

For the following definitions let us consider two polynomials $a, b \in \mathbb{R}[s]$ such that:

$$\begin{aligned} a(s) &= a_n s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0 , \quad \deg\{a(s)\} = n \\ b(s) &= b_k s^k + b_{k-1} s^{k-1} + \ldots + b_1 s + b_0 , \quad \deg\{b(s)\} = k \end{aligned}$$

**Bézout matrix.** We assume that $n = k$. The Bézout matrix (or Bézoutian) of order $n$ associated to $a(s)$ and $b(s)$ is a $n \times n$ matrix obtained as follows:

$$B_n(a, b) = [c_{i,j}]_{i,j=1,2,\ldots,n}$$

where each element $c_{i,j}$ is given by

$$c_{i,j} = \sum_{t=0}^{\min\{i,n+1-j\}} \Big( a_{j+t-1}\, b_{i-t} - a_{i-t}\, b_{j+t-1} \Big) , \quad \text{for } i, j = 1, 2, \ldots, n$$

The Bézout matrix of order $n$ has the next basic properties:

- $B_n(a, b)$ is symmetric as a matrix,
- $B_n(a, b) = -B_n(b, a)$,
- $B_n(a, a) = 0$,
- $B_n(a, b)$ has full rank if and only if $a(s)$ and $b(s)$ are coprime.

**Sylvester matrix.** The Sylvester matrix associated to $a(s)$ and $b(s)$ is the $(n + k) \times (n + k)$ matrix obtained as follows:

$$S(a,b) = \left[\begin{array}{cccccccc} a_n & a_{n-1} & \ldots & a_1 & a_0 & 0 & \ldots & 0 \\ 0 & a_n & a_{n-1} & \ldots & a_1 & a_0 & \ldots & 0 \\ \vdots & \ddots & \ddots & & & & \ddots & \vdots \\ 0 & \ldots & 0 & a_n & a_{n-1} & \ldots & a_1 & a_0 \\ \hline b_k & b_{k-1} & \ldots & b_1 & b_0 & 0 & \ldots & 0 \\ 0 & b_k & b_{k-1} & \ldots & b_1 & b_0 & \ldots & 0 \\ \vdots & \ddots & \ddots & & & & \ddots & \vdots \\ 0 & \ldots & 0 & b_k & b_{k-1} & \ldots & b_1 & b_0 \end{array}\right] \begin{array}{l} \Big\} \text{ k lines} \\ \\ \\ \\ \\ \Big\} \text{ n lines} \\ \\ \\ \end{array}$$

An important property of the Sylvester matrix $S(a, b)$ is that its rank $\rho\big(S(a, b)\big)$, which in simple terms is the number of linear independent columns, determines

the degree of the GCD of $a(s)$ and $b(s)$, such that:

$$\deg\{\gcd\{a,b\}\} = n + k - \rho\big(S(a,b)\big) \tag{2.8}$$

The determinant of $S(a,b)$ is called the *resultant* of $a(s)$ and $b(s)$ and, hence, a Sylvester matrix is also referred to as a *resultant matrix*. An extended form of the Sylvester matrix for sets of many polynomials is presented in Chapter 5.

### 2.2.1 Algebraic tools for numerical computations

**► Eigenvalues, Characteristic polynomial, and Matrix Pencils**

Let $A \in \mathbb{R}^{n\times n}$ and $I$ the $n \times n$ identity matrix. Then the polynomial

$$p_n(\lambda) = \det(\lambda\, I - A)\ ,\ \lambda \in \mathbb{C}$$

is called the *characteristic polynomial* of $A$. The zeros of the characteristic polynomial are called the *eigenvalues* of $A$. Equivalently, $\lambda$ is an eigenvalue of $A$ if and only if there exists a vector $\underline{v} \in \mathbb{R}^n$ such that $A\,\underline{v} = \lambda\,\underline{v}$. The vector $\underline{v}$ is called a *right eigenvector* and similarly the vector $\underline{u} \in \mathbb{R}^n$ for which $\underline{u}^t\, A = \underline{u}^t\, \lambda$ is called a *left eigenvector*. It holds $\underline{u}^t\, \underline{v} = 1$.

Let $A, B \in \mathbb{R}^{n\times n}$, then a linear *matrix pencil* is the matrix defined as

$$T(s) = s\, A - B$$

for $s \in \mathbb{R}$ (or $s \in \mathbb{C}$). Matrix pencils play an important role in numerical linear algebra. A frequent problem that arises in several algebraic computational methods relates to the computation of the eigenvalues of a matrix pencil. We call *eigenvalues of a matrix pencil* $T(s)$ all numbers $s$ for which the determinant $\det(s\, A - B) = 0$. The problem of finding the eigenvalues of a pencil is known as the *generalized eigenvalue problem* [27] and has numerous applications. A special GCD method, presented in [45, 59], is based on matrix pencil theory.

**► Singular value decomposition**

The singular value decomposition (SVD) is a special factorisation method for matrices and it is one of the most important methods in numerical linear algebra with a wide range of applications. The development of the theory of the SVD began in the $19^{th}$ century, but its use became widespread after 1965 when G. H. Golub, W. Kahan, and C. Reinsch showed us how to compute the SVD in an efficient and numerically stable way [26]. The determination of the rank of a matrix (particularly the numerical rank and nearness to rank deficiency), the computation of orthonormal bases for the row and column space of a matrix,

the computation of the inverse or pseudo-inverse of a matrix, and solving linear least-squares problems, or linear systems are some of the problems that the SVD can handle very effectively even when numerical inaccuracies in the data are present [18, 27]. A number of significant properties of the SVD are summarised below.

**Definition 2.3.** i) An $n \times n$ matrix $A$ is said to be *invertible*, if there exists a $n \times n$ matrix $B$ such that $A\,B = B\,A = I_n$, where $I_n$ denotes the $n \times n$ identity matrix. The matrix $B$ is called the *inverse* of $A$ and it is denoted by $A^{-1}$.

ii) An $n \times n$ matrix is said to be *orthogonal*, if $A\,A^t = A^t\,A = I_n$, where $A^t$ denotes the $n \times n$ transpose of $A$. In this case $A^{-1} = A^t$.

**Theorem 2.2** ([18, 27])**.** *Let $A$ be a real $m \times n$ matrix ($A \in \mathbb{R}^{m\times n}$). Then, there always exist orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ such that*

$$U^t\,A\,V = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} = \Sigma$$

*where $\Sigma_1 \in \mathbb{R}^{r\times r}$ is a nonsingular diagonal matrix. The diagonal entries of $\Sigma \in \mathbb{R}^{m\times n}$ are all non-negative and can be arranged in nonincreasing order. The number $r$ of non-zero diagonal entries of $\Sigma$ equals the rank of $A$.*

The decomposition $A = U\,\Sigma\,V^t$ is known as the *singular value decomposition* of $A$. The diagonal entries of $\Sigma$ are called the *singular values* of $A$ and are denoted by $\sigma_i$, $i = 1, 2, \ldots, r$. The columns of $U$ are called *left singular vectors* and those of $V$ are called *right singular vectors*.

The above theorem implies that if $r = \rho(A)$, then there are exactly $r$ positive singular values. These are actually the positive square roots of the nonzero eigenvalues of the matrix $A^tA$ (or $A\,A^t$) [27]. If $r < \min\{m, n\}$, the remaining singular values are zero. Note that the singular values of a matrix are uniquely determined, but the singular vectors are not unique.

**Corollary 2.1.** *A matrix $A \in \mathbb{R}^{n\times n}$ is nonsingular if and only if all its singular values are different from zero.*

An efficient numerical algorithm for the computation of the SVD, which is today a standard algorithm for computing singular values and singular vectors, is known as the *Golub-Kahan-Reinsch algorithm* (GKR-SVD) [26, 27]. The algorithm involves numerical stable procedures such as matrix bidiagonalisation and implicit QR factorisation [18]. A variant, which is more efficient in certain cases, was proposed by Chan in [13] and a method for partial singular value decomposition was presented in [76].

The SVD has become an effective tool in handling various important problems arising in a wide variety of application areas, such as control theory, signal and image processing, network theory, pattern recognition, and robotics. Particularly in control theory, the problems requiring the use of SVD include controllability and observability, realisation of state-space models, balancing, robust feedback stabilization, model reduction and several others related problems. Furthermore, the SVD is the most effective tool in solving least-squares and generalized least-squares problems [25, 28].

### ▶ Compound matrices

Compound matrices [56] are useful algebraic tools that are used in certain GCD methods. The following are necessary to describe the notion of compound matrices [46, 59].

a) $Q_{p,n}$ denotes the set of strictly increasing sequences of $p$ integers $(1 \leq p \leq n)$ chosen from $1, \ldots, n$. The number of the sequences which belong to $Q_{p,n}$ is $\binom{n}{p}$. If $\alpha, \beta \in Q_{p,n}$ we say that $\alpha$ precedes $\beta$ $(\alpha < \beta)$, if there exists an integer $t$ $(1 \leq t \leq p)$ for which $\alpha_1 = \beta_1, \ldots, a_{t-1} = \beta_{t-1}$, $\alpha_t < \beta_t$, where $\alpha_i, \beta_i$ denote the elements of $\alpha$ and $\beta$. This describes the lexicographic ordering of the elements of $Q_{p,n}$. The set of sequences $Q_{p,n}$ will be assumed with its sequences lexicographically ordered and the elements of the ordered set $Q_{p,n}$ will be denoted by $\omega$.

b) Suppose $A = [a_{i,j}] \in \mathbb{R}^{m \times n}$, let $k, p$ be positive integers satisfying $1 \leq k \leq m$, $1 \leq p \leq n$ and let $\omega = (i_1, \ldots, i_k) \in Q_{k,m}$ and $\tilde{\omega} = (j_1, \ldots, j_p) \in Q_{p,n}$. Then, $A[\omega|\tilde{\omega}] \in \mathbb{R}^{k \times p}$ denotes the submatrix of $A$ which contains the rows $i_1, \ldots, i_k$ and the columns $j_1, \ldots, j_p$.

c) Let $A \in \mathbb{R}^{m \times n}$ and $1 \leq p \leq \min\{m, n\}$, then the $p^{th}$ compound matrix of $A$ is the $\binom{m}{p} \times \binom{n}{p}$ [1] matrix whose entries are $c_{i,j} = \det\{A[\omega_{i-1}|\tilde{\omega}_{j-1}]\}$, where $\omega_{i-1} \in Q_{p,m}$, $\tilde{\omega}_{j-1} \in Q_{p,n}$ for $1 \leq i \leq \binom{m}{p}$ and $1 \leq j \leq \binom{n}{p}$. This matrix will be denoted by $C_p(A)$.

### ▶ Minors of matrices

Let $A$ be an $m \times n$ matrix and $p$ an integer with $0 < p \leq \min\{m, n\}$. A $p \times p$ *minor* of $A$ is the determinant of a $p \times p$ matrix obtained from $A$ by deleting $m - p$ rows and $n - p$ columns. Since there are $\binom{m}{p}$ [1] ways to choose $p$ rows from $m$ rows, and there are $\binom{n}{p}$ [1] ways to choose $p$ columns from $n$ columns, there are a total of $\binom{m}{p} \cdot \binom{n}{p}$ minors of size $p \times p$.

[1] $\binom{k}{p} = \frac{k(k-1)(k-2)\ldots(k-p+1)}{p(p-1)(p-2)\ldots 1}$ for $k = m$ or $n$.

The (i,j) minor (usually denoted by $M_{ij}$) of an $n \times n$ square matrix $A$ is defined as the determinant of the $(n-1) \times (n-1)$ matrix formed by removing from $A$ its $i^{th}$ row and $j^{th}$ column. An (i,j) minor $M_{ij}$ is also called the minor of the element $a_{ij}$ of matrix $A$.

### 2.2.2 Basic concepts from Linear Systems

We summarise here the fundamentals of linear systems which are essential for describing the work related to GCD and LCM methods that have been developed by using concepts from systems theory [45, 48]. Basic definitions and tools are introduced, related to important properties of linear systems, such as system poles and zeros, controllability, observability, and stability.

A linear system may be represented in terms of first order differential equations as

$$S(A,B,C,D) \; : \quad \begin{array}{rcl} \dot{x}(t) & = & A\,x(t) + B\,u(t) \\ y(t) & = & C\,x(t) + D\,u(t) \end{array} \tag{2.9}$$

where the variable $t$ represents time, $x(t)$ is the *state vector*, $u(t)$ is the *input vector* and $y(t)$ is the *output vector*. The matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$, $C \in \mathbb{R}^{m\times n}$, and $D \in \mathbb{R}^{m\times p}$ are the *state*, *input*, *output*, and *feedforward* matrices, respectively. In system matrix form, we can represent the system by:

$$P(s) = \begin{bmatrix} s\,I - A & -B \\ -C & -D \end{bmatrix} \tag{2.10}$$

or by the *transfer function* model:

$$G(s) = C\,(s\,I - A)^{-1}B + D \tag{2.11}$$

which is an $m \times p$ polynomial matrix. The 4-tuple $(A,B,C,D)$ is said to be a *realisation* of $G(s)$. The matrix $P(s)$ is a matrix pencil entirely characterising the state-space model and it is known as the *Rosenbrock System Matrix Pencil* [65].

For the state-space model $S(A,B,C,D)$ of which is excited by an initial condition $x(0) = x_0$ and a control input $u(t)$, the corresponding solutions for the state and output trajectories $x(t),\ y(t)$ are given by [1] :

$$x(t) \;=\; e^{At}x_0 + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau \tag{2.12}$$

$$y(t) \;=\; Ce^{At}x_0 + \int_0^t Ce^{A(t-\tau)}Bu(\tau)d\tau + Du(t) \tag{2.13}$$

Taking Laplace transforms of (2.9), the following frequency domain representations

of the solutions are obtained:

$$x(s) = (sI - A)^{-1}x_0 + (sI - A)^{-1}Bu(s) \tag{2.14}$$
$$y(s) = C(sI - A)^{-1}x_0 + \big(C(sI - A)^{-1}B + D\big)\,u(s) \tag{2.15}$$

### ▶ Controllability, Observability

Let us consider a system $S(A, B, C, D)$ described by the equation (2.9). We say that the system is *controllable* if given any initial state $x(t_0) = x_0$, there exists a finite time $t_1 > t_0$ and a control $u(t)$ defined on $t_0 \leq t \leq t_1$, such that $x(t_1) = 0$. Therefore, controllability refers to the ability of a system to transfer the state from $x_0$ to the zero state in finite time.

**Theorem 2.3** ([1])**.** *The pair $(A, B)$ is controllable, if and only if*

$$rank\Big(\big[B, AB, A^2B, \ldots, A^{n-1}B\big]\Big) = n$$

The system (2.9) is said to be *observable*, if for any state $x(t_0) = x_0$ and given control vector $u(t)$ knowledge of $y(t)$ on $t_0 \leq t \leq t_1$ is sufficient to determine $x_0$. Therefore, observability means that we can determine the initial state of the system for a suitable measurement of the output $y(t)$. The notion of observability is dual to that of controllability. The *dual system* of (2.9) is defined as the system:

$$S(A^t, C^t, B^t, D^t) \;:\; \begin{array}{rcl} \dot{x}_d(t) &=& A^t\,x_d(t) + C^t\,u_d(t) \\ y_d(t) &=& B^t\,x_d(t) + D^t\,u_d(t) \end{array} \tag{2.16}$$

**Theorem 2.4** ([4])**.** *The system described in (2.9) is observable if and only if its dual system (2.16) is controllable. Thus, the pair $(A, B)$ is observable if and only if the pair $(A^t, C^t)$ is controllable, that is :*

$$rank\Big(\big[C^t, A^tC^t, \ldots, (A^t)^{n-1}C^t\big]\Big) = n$$

There are tests for controllability and observability that involve the eigenvalues and the eigenvectors of $A$. These tests are particularly useful both as theoretical and computational tools. We can also check controllability and observability of a system in the following ways [1] :

*Rank tests for controllability and observability :*

- The pair $(A, B)$ is controllable, if and only if

$$rank\Big(\big[\lambda\, I - A, B\big]\Big) = n$$

  for all eigenvalues $\lambda$ of $A$.

- The eigenvalue $\lambda_i$ is an *uncontrollable eigenvalue* of $A$, if and only if

$$rank\left([\lambda_i I - A, B]\right) < n$$

- The pair $(A, B)$ is observable, if and only if

$$rank\left(\begin{bmatrix} \lambda I - A \\ C \end{bmatrix}\right) = n$$

  for all eigenvalues $\lambda$ of $A$.

- The eigenvalue $\lambda_i$ is an *unobservable eigenvalue* of $A$, if and only if

$$rank\left(\begin{bmatrix} \lambda_i I - A \\ C \end{bmatrix}\right) < n$$

The uncontrollable, unobservable, uncontrollable and unobservable eigenvalues are also referred to as *input*, *output*, and *input-output decoupling zeros* (idz,odz,i-odz) [65] and the corresponding sets, including multiplicities, are denoted by $\mathcal{Z}_{ID}$, $\mathcal{Z}_{OD}$, $\mathcal{Z}_{IOD}$, respectively. There exist more definitions of controllability and observability, which can be found in [1, 38, 50, 65]. Alternative algebraic tests based on the restricted pencils are given in [39].

### ▶ Poles and Zeros, Pole and Zero polynomials

Classical control design techniques are based on the concepts of *poles* and *zeros* of a rational function. Every rational transfer function can be expressed as a polynomial matrix (i.e. a matrix whose elements are univariate polynomials), divided by a common denominator polynomial. So, every polynomial matrix can be reduced to a canonical form known as the *Smith form* [24].

**Definition 2.4.** A polynomial matrix is called *unimodular* if it has an inverse which is also a polynomial matrix.

There are three elementary operations which can be performed on polynomial matrices:

- Interchange of any two rows, or columns.
- Multiplication of one row or column by a nonzero constant.
- Addition of a polynomial multiple of one row or column to another.

Each of these elementary operations can be represented by multiplying a polynomial matrix by a suitable matrix, called an *elementary matrix*. It is easy to show that all elementary matrices are unimodular [24]. Two (polynomial or

rational) matrices $P(s)$ and $Q(s)$ are *equivalent* if there exist sequences of left $\{L_1(s), L_2(s), \ldots, L_l(s)\}$ and right $\{R_1(s), R_2(s), \ldots, R_r(s)\}$ elementary matrices such that

$$P(s) = L_1(s)\, L_2(s) \cdots, L_l(s)\, Q(s)\, R_1(s)\, R_2(s) \cdots, R_r(s)$$

The next result states that every polynomial matrix is equivalent to a diagonal polynomial matrix known as the Smith form [24].

**Theorem 2.5.** *Let $P(s)$ be a polynomial matrix of normal rank $r$ (i.e. of rank $r$ for almost all $s$). Then, $P(s)$ may be transformed by a sequence of elementary row and column operations into a pseudo-diagonal polynomial matrix $P_S(s)$ having the form:*

$$P_S(s) = \text{diag}\,\{\varepsilon_1(s), \varepsilon_2(s), \ldots, \varepsilon_r(s), 0, \ldots, 0\}$$

*in which each $\varepsilon_i(r)$, $i = 1, 2, \ldots, r$ is a monic polynomial satisfying the divisibility property $\varepsilon_i(s)|\varepsilon_{i+1}(s)$ for $i = 1, 2, \ldots, r-1$ (i.e. $\varepsilon_i(s)$ divides $\varepsilon_{i+1}(s)$ without remainder). Moreover, if we define the determinantal divisors*

$$\begin{aligned} D_0(s) &= 1 \\ D_1(s) &= \textit{GCD of all } i \times i \textit{ minors of } P(s) \end{aligned}$$

*where each GCD is normalised to be a monic polynomial, then*

$$\varepsilon_i(s) = \frac{D_i(s)}{D_{i-1}(s)}\,, \quad i = 1, 2, \ldots, r$$

*The matrix $P_S(s)$ is the Smith form of $P(s)$, and the $\varepsilon_i(s)$ are called the invariant factors of $P(s)$.*

It is clear that the Smith form of a polynomial matrix is uniquely defined, and that two equivalent polynomial matrices have the same Smith form. The Smith form is thus a canonical form for a set of equivalent polynomial matrices. This can be extended to rational matrices [65].

**Theorem 2.6.** *Let $G(s)$ be a rational matrix of normal rank $r$. Then $G(s)$ may be transformed by a series of elementary row and column operations into a pseudo-diagonal rational matrix of the form:*

$$M(s) = \text{diag}\left\{\frac{\varepsilon_1(s)}{\psi_1(s)}, \frac{\varepsilon_2(s)}{\psi_2(s)}, \ldots, \frac{\varepsilon_r(s)}{\psi_r(s)}, 0, \ldots, 0\right\}$$

*in which the monic polynomials $\{\varepsilon_i(s), \psi_i(s)\}$ are coprime for each $i$ and satisfy the divisibility properties $\varepsilon_i(s)|\varepsilon_{i+1}(s)$ and $\psi_{i+1}(s)|\psi_i(s)$ for $i = 1, 2, \ldots, r-1$. $M(s)$ is the Smith McMillan form of $G(s)$.*

We now define the poles and zeros of a transfer function matrix by means of the Smith-McMillan form [65].

**Definition 2.5.** Let $G(s)$ be a rational transfer function matrix with Smith-McMillan form $M(s)$. The *pole* $p(s)$ and *zero* $z(s)$ polynomials, respectively, are defined as

$$p(s) = \psi_1(s)\,\psi_2(s)\cdots\psi_r(s) \tag{2.17}$$

$$z(s) = \varepsilon_1(s)\,\varepsilon_2(s)\cdots\varepsilon_r(s) \tag{2.18}$$

The roots of $p(s)$ and $z(s)$ are called the *poles* and *zeros* of $G(s)$, respectively.

In other words, the poles of $G(s)$ are all the roots of the denominator polynomials $\psi_i(s)$ of the Smith-McMillan form of $G(s)$. If $p_0$ is a pole of $G(s)$, then $(s-p_0)^\nu$ must be a factor of some $\psi_i(s)$. The number $\nu$ ($\nu \geq 1$) is called the *multiplicity of the pole*, and if $\nu = 1$ we say that $p_0$ is a *simple pole*. Zeros and their multiplicity are defined similarly, in terms of the numerator polynomials $\varepsilon_i(s)$ of the Smith-McMillan form.

*REMARK* 2.2. If $G(s)$ is square, then $\det(G(s)) = c\,\dfrac{z(s)}{p(s)}$ for some constant $c$. In this case, although the pair of polynomials $\{\varepsilon_i(s), \psi_i(s)\}$ is coprime for each $i = 1, 2, \ldots, r$, it is possible that there exist common factors between $p(s)$ and $z(s)$ which cancel out in forming $\det(G(s))$.

**Definition 2.6.** The degree of the pole polynomial $p(s)$ is the *McMillan degree* of $G(s)$.

Zeros defined via the Smith-McMillan form are often called *transmission zeros*, in order to distinguish them from other kinds of zeros which have been defined. For a single input, single output (SISO) system represented by a rational transfer function $G(s)$, where $G(s) = \dfrac{n(s)}{d(s)}$ and $n(s)$, $d(s)$ are coprime polynomials with $\deg\{n(s)\} = r$ and $\deg\{d(s)\} = n$, we define as *finite poles* the roots of $d(s)$ and as *finite zeros* the zeros of $n(s)$. If $r < n$ we say that $G(s)$ has an infinite zero of order $n - r$, and if $r > n$, then $G(s)$ has a infinite pole with order $n - r$.

Poles and zeros are also related to the eigenvalues of the system matrix $A$. The eigenvalues and eigenvectors of the matrix $A$ define the internal dynamics of the system $S(A, B, C, D)$. For every eigenvalue $\lambda$ of $A$ we have two eigenvalue-eigenvector problems:

$$A\,\underline{v} = \lambda\,\underline{v}\,, \tag{2.19}$$

$$\underline{w}^t\,A = \underline{w}^t\lambda\,, \quad \underline{w}^t\underline{v} = 1 \tag{2.20}$$

The triple $(\lambda, \underline{v}, \underline{w}^t)$ is called a *system mode*. If $\phi(A)$ is the set of distinct eigenvalues, then the *structure* of $\lambda \in \phi(A)$ is defined by the $\lambda$-*Segré characteristic* $S(\lambda) =$

$\{\nu_i, i \in \tilde{q} \subset \mathbb{N}\}$, that is the sequence of dimensions of $\lambda$-Jordan blocks in the Jordan form of $A$. Alternatively, $S(\lambda)$ is defined by the set of degrees of the $(s-\lambda)^\nu$ type of the Smith form of $sI - A$. Then $\sum_{i=1}^{q} \nu_i = p$ is called the *algebraic multiplicity* and $q$ is the *geometric multiplicity* of $\lambda$. The maximal of all geometric multiplicities of the eigenvalues of $A$ is referred to as the *Segré index* of $A$.

**Definition 2.7.** The set of eigenvalues of the matrix $A$, which are the roots of the characteristic polynomial of $A$, are called the *system internal poles*, or the *system eigenvalues*. The roots of the pole polynomial of $G(s)$ are called the *external system poles* or *system poles*.

**Definition 2.8.** The zeros of the system are defined as those frequencies $s_0 \in \mathbb{C}$ for which there exists an input $u(t) = u_0\, e^{s_0 t}$ such that, given zero initial conditions for the state of the system $x(t)$, the output $y(t)$ is zero.

The above dynamic characterisation leads to a matrix pencil characterisation of zeros [44]. A linear system may also have poles and zeros at infinity $\infty$, which indicate that $G(\infty)$ loses rank. Poles are associated with resonance phenomena (explosion of the gain) and zeros are associated with antiresonance phenomena (vanishing of the gain). In this sense, the notions of poles and zeros are dual and it is this basic property that motivates a number of definitions and problems that relate to multivariable poles and zeros [41, 55].

### ▶ Internal-External and Total stability

The most important concept and property for any system is that of *stability*, which has to do with the behaviour of all trajectories which may be generated for families of initial conditions and control input. For linear, time invariant systems the notions of stability, which are more frequently used are defined next. We consider stability of equilibrium points, whereas stability of motion is always reduced to the previous case. Note that the origin ($x = 0$) is always an equilibrium point for $S(A, B, C, D)$ models.

**Definition 2.9** ([1])**.** Given a system of first-order differential equations :

$$\dot{x} = \mathcal{F}(t, x)\ ,\ x \in \mathbb{R}^n$$

a point $x_e \in \mathbb{R}^n$ is called an *equilibrium point* of the system (or simply an equilibrium) at time $t_0 > 0$, if $\mathcal{F}(t, x_0) = 0$ for all $t > t_0$.

**Definition 2.10** ([1])**.** The state-space model $S(A, B, C, D)$ will be called:

i) *Internally stable in the sense of Lyapunov* (LIS), if for any initial $x(0)$ the zero input response (free motion, $u(t) = 0$) remain bounded for all $t \geq 0$.

ii) *Asymptotically internally stable*, if for any initial state $x(0)$ the zero input response remains bounded for all $t \geq 0$ and tends to zero as $t \to \infty$. This property will be referred to in short as *internal stability* (IS).

iii) *Bounded Input Bounded Output stable* (BIBO), if for any bounded input the zero state output response ($x(0) = 0$) is bounded.

iv) *Totally stable* (TS) if for any initial state $x(0)$ and any bounded input $u(t)$, the output, as well as all state variables, are bounded.

The notion of BIBO stability refers to the transfer function description and may also be called as *external stability*. A number of criteria for these properties, based on eigenvalues-poles, are summarised below.

**Theorem 2.7** ([15])**.** *Consider the system $S(A, B, C, D)$ with $G(s)$ transfer function and let $\{\lambda_i = \sigma_i + \mathtt{i}\,\omega_i,\ i \in \tilde{n} \subset \mathbb{N}\}$, $\{p_j = \bar{\sigma}_j + \mathtt{i}\,\bar{\omega}_j,\ j \in \tilde{k} \subset \mathbb{N}\}$ be the sets of eigenvalues, poles respectively. The system has the following properties:*

*i) Lyapunov internally stable, if and only if $\sigma_i \leq 0$, for all $i \in \tilde{n}$, and those with $\sigma_i = 0$ have a simple structure (algebraic multiplicity is equal to the geometric multiplicity).*

*ii) Asymptotically internally stable, if and only if $\sigma_i < 0$, for all $i \in \tilde{n}$.*

*iii) BIBO stable, if and only if $\bar{\sigma}_j < 0$, for all $j \in \tilde{k}$.*

*iv) Totally stable, if it is Lyapunov internally stable and BIBO stable.*

Note that IS implies BIBO-stability and thus TS. BIBO-stability does not always imply IS, since transfer function and state space are not always equivalent. If the two representations are equivalent (when system is both controllable and observable), then BIBO-stability is equivalent to IS and thus TS.

Eigenvalues and poles are indicators of stability. Equivalent tests for stability, without computing the eigenvalues or poles, are defined on the characteristic or pole polynomial by the Routh-Hurwitz conditions [24].

### 2.2.3 Almost zeros of a set of polynomials

The subject of nongeneric computations has as one of its most important topics the study of *almost zeros*. A summary of the notion is given next [46]. The computational issues and the feedback significance of the notion (trapping disks for multiparameter root locus) is given in [43]. We consider the set $\mathcal{P}_{m,n}$ as defined in (2.7) (abbreviated as $\mathcal{P}$), the respective vector representative $\underline{p}(s)$, and the basis matrix $P_m$.

When $s \in \mathbb{C}$, $\underline{p}(s)$ defines a vector valued analytic function with domain $\mathbb{C}$ and co-domain $\mathbb{C}^m$; the norm of $\underline{p}(s)$ is defined as a positive real function with domain $\mathbb{C}$, such that

$$\|\underline{p}(s)\| = \sqrt{\underline{p}^t(s^*)\,\underline{p}(s)} = \sqrt{\underline{e}_n^t(s^*)\,P_m^t\,P_m\,\underline{e}_n^t(s)} \tag{2.21}$$

where $s^*$ is the complex conjugate of $s$. Note, that if $q(s) = s - c$ is a common factor of the polynomials $p_i(s)$, $i = 1, 2, \ldots, m$, then $p_i(c) = 0$, $\underline{p}(c) = \underline{0}$ and thus $\|\underline{p}(c)\| = 0$. This observation leads to the following definition.

**Definition 2.11.** Let $\mathcal{P}$ be a set of polynomials of $\mathbb{R}[s]$, $\underline{p}(s)$ be the vector representative and let $\phi(\sigma, \omega) = \|\underline{p}(s)\|$, where $s = \sigma + \mathtt{i}\,\omega \in \mathbb{C}$. An ordered pair $(z_k, \varepsilon_k)$, $z_k \in \mathbb{C}$, $\varepsilon_k \in \mathbb{R}$ and $\varepsilon_k \geq 0$, defines an *almost zero* of $\mathcal{P}$ at $s = z_k$ and of order $\varepsilon_k$, if $\phi(\sigma, \omega)$ has a minimum at $s = z_k$ with value $\varepsilon_k$. From the set $Z = \{(z_k, \varepsilon_k),\ k = 1, 2, \ldots, r\}$ of almost zeros of $\mathcal{P}$ the element $(z^*, \varepsilon^*)$ for which $\varepsilon^* = \min\{\varepsilon_k,\ k = 1, 2, \ldots, r\}$ is defined as the *prime almost zero* of $\mathcal{P}$.

It is clear that if $\mathcal{P}$ has an exact zero, then the corresponding $\varepsilon$ is zero. Clearly the previous definition is an extension of the concept of exact zero to that of the almost zero. The magnitude of $\varepsilon$ at an almost zero $s = z$ provides an indication of how well $z$ may be considered as an approximate zero of $p_i$. However, that $\varepsilon$ depends on the scaling of the polynomials $p_i(s)$ in $\mathcal{P}$ by a constant $c \in \mathbb{R} \setminus \{0\}$. The general properties of the distribution of the almost zeros of a set of polynomials $\mathcal{P}$ on the complex plane were considered in [43] and are summarized below.

**Theorem 2.8.** *The prime almost zero of $\mathcal{P}$ is always within the circle centred at the origin of the complex plane and with radius $p^*$, defined as the unique positive solution of the equation:*

$$1 + r^2 + \ldots + r^{2n} = \frac{\overline{\gamma}^2}{\gamma^2} = \theta^2$$

*where $\overline{\gamma}, \gamma$ denote the maximum and minimum singular values of $P_m$, respectively.*

The term $\theta$ will be referred to as the *condition number* of $\mathcal{P}$. The disc $[0, p^*]$ within which the prime almost zero lies, is referred to as the *prime disc* of $\mathcal{P}$. The following general results may be stated for the radius $p^*$.

**Proposition 2.1.** *If $n$ is the maximum degree and $\theta$ the condition number of $\mathcal{P}$, then the radius $p^* = f(n, \theta)$ of the prime disc is a uniquely defined function of $n$ and $\theta$ and it has the following properties:*

1. *The radius $p^*$ is invariant under the scaling of the polynomial of $\mathcal{P}$ by the same nonzero constant $c$.*
2. *The radius $p^*$ is monotonically decreasing function of $n$ and $\dfrac{1}{\theta}$.*

*3. The radius $p^*$ is within the following intervals:*

*i) If $n+1 > \theta^2$, then $0 < p^* < 1$*

*ii) If $n+1 < \theta^2$, then $2 < p^* < \sqrt{\theta}$*

*iii) If $n+1 = \theta^2$, then $p^* = 1$*

The conditioning of the polynomials plays an important role in determining the position of the prime almost zero. In fact, the prime almost zero is always in the vicinity of the origin of the complex plane. The uncertainty in its exact position is measured by the radius of the prime disc. Well conditioned sets of polynomials $\mathcal{P}$ (i.e. $\theta \approx 1$) have a very small radius prime disc even for very small values of the degree $n$. Badly conditioned sets of polynomials $\mathcal{P}$ (i.e. $\theta >> 1$) have a very large radius prime disc even for large values of the degree $n$. The computation of almost zeros may be achieved by deriving the necessary conditions for the minimum [43]. The position of the almost zero varies according to the scaling which is used. Therefore, instead of looking for *approximate common roots* we can look for *approximate common factors* and this extends to the harder problem of calculating an *approximate greatest common divisor*.

### 2.2.4 Basic concepts of numerical algorithms

In mathematics, computer science, and related subjects, an algorithm is an effective method for solving a problem expressed as a finite sequence of steps. Each algorithm is a list of well-defined instructions for completing a task. Starting from an initial state, the instructions describe a computation that proceeds through a well-defined series of successive states, eventually terminating in a final ending state.

The most important property for an algorithm is *stability*. The study of stability is done by means of *round-off error analysis* [18, 81]. There are two types of error analysis: i) *forward error analysis* and ii) *backward error analysis*.

In forward error analysis the aim is to see how the computed solution, obtained by the algorithm, differs from the exact solution based on the same data. Conversely, backward error analysis relates the error to the data of the problem rather than to the problem's solution. The following definitions for the forward and backward stability of an algorithm are given [18].

**Definition 2.12.** Given an algebraic problem :

a) An algorithm will be called *forward stable* if the computed solution is close to the exact solution, in some sense.

b) An algorithm will be called *backward stable* if it produces an exact solution to a nearby problem.

In the present study, by "stability" we will imply "backward stability" and a "backward stable" algorithm will be referred to as *numerically stable* or simply *stable*. In a general sense, a numerically stable algorithm produces satisfactory results. However, if the produced results are completely unsatisfactory, this does not necessarily means that the algorithm is unstable. The accuracy of the input data and how the solution of the problem will change if the input data contain some impurities (noise) has a significant impact on the result of an algorithm. The accuracy or inaccuracy of the computed result relies also on a property of the problem called *conditioning*.

**Definition 2.13** ([18])**.** A problem (with respect to a given set of data) is called *ill-conditioned*, if a small relative error in data causes a large relative error in the solution, regardless of the method of solution and algorithm. Otherwise, it is called *well-conditioned*.

Therefore, the conditioning of a problem is a property of the problem itself. If the problem is ill-conditioned, no matter how stable the algorithm is, the accuracy of the computed solution cannot be guaranteed. However, if a stable algorithm is applied to an ill-conditioned problem, it should not introduce more error than what the data warrants. Conversely, when a stable algorithm is applied to a well-conditioned problem, the computed solution should be near the exact solution, because stability will guarantee the exact solution of a nearby problem and well-conditioning will guarantee that the solution to the original problem and that of the nearby problem are close [18]. A number called the *condition number* is usually associated with a problem. The condition number indicates whether the problem is ill or well conditioned. More specifically, the condition number gives a bound for the relative error in the solution when a small perturbation is applied to the input data. For example, for the linear system problem $A\,\underline{x} = \underline{b}$ the condition number is $Cond(A) = \|A\|\,\|A^{-1}\|$.

Another property that characterises an algorithm is *complexity*. The complexity of an algorithm refers to the amount of the performed operations (additions, multiplications). In early computational systems with finite precision the amount of the performed numerical (floating-point) operations was measured in *flops* (1 flop = 1 addition + 1 multiplication). However, with the introduction of sophisticated symbolic-numeric computational systems this measurement is now considered outdated, but we may still use the total amount of arithmetic operations, which are performed by an algorithm, as an indicator of its computational efficiency. For example, in algorithms involving matrix computations, if we assume that $n$ is the highest matrix dimension, then an algorithm is considered to be computationally efficient when the amount of arithmetic operations (symbolic and numeric) is about $n^3$, and we write $O(n^3)$.

The formulation of an algorithm which combines accuracy and low complexity is crucial for any algebraic problem. The search for numerically stable and efficient algorithms with low complexity is also a subject of the current research.

## 2.3 Methods for the computation of the GCD of polynomials

The algorithm associated with Euclid's division method [32] is the oldest known solution to the problem of computing the GCD. The work of Sylvester in $19^{th}$ century was the next development to the problem [5]. The computational methods for computing the GCD of real univariate polynomials can be separated in two main categories:

a) The *Euclidean type methods* which rely on Euclid's division algorithm and its variations.

b) The *matrix-based methods* which are based on the processing of a matrix formed directly from the coefficients of the given polynomials.

According to the way that the matrix is processed, the matrix-based methods are separated into those which

i) form a matrix for two polynomials and work on pairwise computations iteratively,

ii) form and work in direct or iterative way with a matrix that corresponds to the whole set of polynomials.

Early GCD algorithms were developed using Euclidean-based methods, applied to two polynomials. A method, which is essentially equivalent to Euclid's algorithm, uses the Routh Array algorithm (Fryer 1959) [62]. In 1960, Weinstock [62] proposed an iterative method that involves polynomial divisions. Blankiship and Brown also proposed GCD methods for polynomials based on Euclid's division algorithm [11, 12]. In 1985, Schönhage introduced the notion of *Quasi-GCD* [70] and presented an algorithm which computes a numerical pseudo-remainder sequence $(a_i, b_i)$ for a pair of polynomials $(a, b)$ in a weakly stable way, accepting only the pairs that are well-conditioned (because the others produce instability). The maximum index $i$ for which $(a_i, b_i)$ is accepted, gives the Quasi-GCD $g = a_i$ provided that the norm-1 of $b_i$ is small enough in a sense précised in [70].

The Euclidean algorithm is efficient when the polynomials have integer coefficients, but it becomes inefficient when the polynomials have coefficients from the field of real numbers due to the use of finite precision arithmetic, which introduces numerical errors into the solution. It is proved in practise that Euclid's

algorithm does not perform well when the polynomial coefficients are inexactly known. Considering this problem, Noda and Sasaki (1989) [61, 68] described a special version of Euclid's algorithm for computing the GCD of a pair of co-prime polynomials with inexact coefficients. This approach is amongst the first attempts to define and compute an approximate GCD of polynomials by means of symbolic-numeric computations. The proposed iterative algorithm is actually a naive extension of the traditional algebraic Euclidean algorithm and it was designed to compute approximate common factors of the input polynomials with floating-point number coefficients. The principal behind this method, referred to as the *Approximate GCD* (AGCD) method [68], is that close roots are calculated as if they are approximate multiple roots. This method is first applied to solve ill-conditioned polynomial equations and then, close roots in a given equation are separated as approximate multiple roots by calculation of the approximate GCD of the equation and its derivative. The AGCD method is based on the square-free decomposition of an appropriate polynomial equation by using the polynomial remainder sequence that is produced by the iterative application of the Euclidean algorithm to the original pair of polynomials [68, 69].

In recent years there has been a substantial effort to develop effective GCD algorithms which are suitable for incorporation into a computer algebra package. The use of finite precision arithmetic in computer algebra makes the extension of the Euclidean algorithm to sets of many polynomials a rather difficult task. The iterative application of the Euclidean algorithm to two polynomials at a time, often results in a total numerical error which might exceed the machine's fixed numerical tolerance. On the other hand, the developed matrix-based methods tend to be more effective in handling sets of several polynomials and producing solutions of better numerical quality. However, the implementation of such algorithms in a software programming environment needs a lot of attention due to the accumulation of additional numerical errors other than the errors introduced by the input data. One of the first matrix-based methods was proposed by Blankinship in 1963 [11]. The use of matrices in the problem of computing the GCD of many polynomials appears also in Barnett's work [3, 5], who developed a technique of computing the degree and the coefficients of the GCD, using companion and Sylvester matrices.

The development of numerical stable GCD algorithms which can deal with polynomials of inexact data has been intensely studied the past thirty years [8, 16, 20, 42, 43, 46, 53, 61, 63, 67, 83, 85]. The various techniques, which have been developed for the computation of *approximate* solutions, are based on methodologies, where exact properties of these notions are relaxed and appropriate solutions are sought by using a variety of numerical tests. The basis of such approaches is the reduction of the general algebraic problems to equivalent linear algebra problems which are suitable for study as approximation problems. The

definition of *almost zeros* [43] gave another perspective to the computation of the GCD of polynomials. The notion of almost zeros is linked to the *almost GCD* problem and it is based on a relaxation of the exact notion of a zero. This has provided the motivation for the definition of approximate solutions to the harder problem, which is the definition and computation of *approximate GCDs*. The definition of almost zeros is now taking a different formulation with the recent definition of such problems as distance problems in a projective space [21, 42].

A fundamental problem is the difficulty in characterising the accuracy of effectiveness of such methods, as well as, determining whether such solutions are *optimal* in some sense with respect to all other techniques that may offer approximate solutions. A number of GCD algorithms specifically developed to manipulate sets of several polynomials with inexact coefficients have as common characteristic a method which primarily performs singular value decomposition (SVD) [27] to estimate the "best" degree of the GCD and continues with the computation of the coefficients of the approximate GCD.

The main methods which formulate algorithms with common characteristics the manipulation of sets of several polynomials and the use of the SVD process, will be described in the following sections.

### 2.3.1 The Matrix Pencil method

The Matrix Pencil method (MP) is a direct matrix-based method which relies on the characterisation of the GCD of a set $\mathcal{P}_{m,n}$ of $m > 2$ polynomials of maximum degree $n > 1$ as the output decoupling zero polynomial of a linear system $S(\hat{A}, \hat{C})$ that may be associated with $\mathcal{P}_{m,n}$. The theoretical basis of the Matrix Pencil methodology derives from the system properties of zeros [44, 55], where the GCD characterisation is reduced to. The computation of the GCD is reduced to finding the finite zeros of the pencil $T(s) = s\,W - \hat{A}\,W$, where $W$ is a basis matrix of the unobservable subspace $\mathcal{W}$ of $S(\hat{A}, \hat{C})$. If $k = \dim\{\mathcal{W}\}$, the GCD is determined as any nonzero entry of the $k^{th}$ compound $C_k(s\,W - \hat{A}\,W)$. The method defines the exact degree of GCD, works satisfactorily with any number of polynomials, and evaluates successfully approximate solutions.

The algorithm of the MP method uses stable algebraic processes, such as SVD for computing the right $\mathcal{N}_r$ and left $\mathcal{N}_l$ nullspaces of appropriate matrices. The main target of the MP algorithm is to form the GCD pencil $T(s)$ and specify any minor of maximal order, which gives the required GCD. This specification can be done symbolically. The MP method has been presented and analysed in [45]. An earlier comparison of the MP method with other existing methods can be found in [59] and a new approach to its numerical implementation has been given in [49, 72].

### ▶ The standard Matrix Pencil method

For a given polynomial set $\mathcal{P}_{h+1,n}$, let $P_{h+1}$ be the basis matrix of the set formed directly from the coefficients of the polynomials of the set. The following theorem underlie the Matrix Pencil method.

**Theorem 2.9** ([45]). *Let the set of univariate polynomials $\mathcal{P}_{h+1,n}$, $P_{h+1}$ a basis matrix with* $\text{rank}(P_{h+1}) = \rho < n+1$, *$M \in \mathbb{R}^{(n+1)\times\mu}$ with $\mu = n - \rho + 1$ a basis matrix for the right nullspace $\mathcal{N}_r(P_{h+1})$ of $P_{h+1}$, and $M_1 \in \mathbb{R}^{n\times\mu}$ the submatrix of $M$ obtained by deleting the last row of $M$. If $p(s) \in \mathcal{P}_{h+1,n}$ is any monic polynomial of degree $n$, $\hat{A} \in \mathbb{R}^{n\times n}$ is the associated companion matrix, and $\hat{C} \in \mathbb{R}^{(\rho-1)\times n}$ with* $\text{rank}(\hat{C}) = \rho - 1$ *is such that $\hat{C}M_1 = 0$, then the unobservable modes of the system*

$$S(\hat{A}, \hat{C}) : \ \dot{x} = \hat{A}x, \ y = \hat{C}x \tag{2.22}$$

*with multiplicities included define the roots of the GCD of $\mathcal{P}_{h+1,n}$. Then $S(\hat{A}, \hat{C})$ will be called the* associated system *of $\mathcal{P}_{h+1,n}$ and the observability matrix*

$$Q(\hat{A}, \hat{C}) = \left[\hat{C}^t, \hat{A}^t\hat{C}^t, \ldots, (\hat{A}^t)^{n-1}\hat{C}^t\right]^t \in \mathbb{R}^{n(\rho-1)\times n} \tag{2.23}$$

*will be referred to as a* reduced resultant *of $\mathcal{P}_{h+1,n}$.*

*REMARK* 2.3. If $\mathcal{N}_r(P_{h+1}) = \{0\}$, then the set $\mathcal{P}_{h+1,n}$ is coprime. Equivalently, if $S(\hat{A}, \hat{C})$ is observable, $\mathcal{P}_{h+1,n}$ is coprime.

Let now $Q(\hat{A}, \hat{C})$ be the corresponding reduced resultant, $\text{rank}(Q(\hat{A}, \hat{C})) < n$ and $\mathcal{W} \triangleq \mathcal{N}_r(Q(\hat{A}, \hat{C})) \neq \{0\}$, $k = \dim\{\mathcal{W}\}$ and $W$ a basis matrix for $\mathcal{W}$. The pencil $T(s) = s\,W - \hat{A}\,W$ characterizes the set $\mathcal{P}_{h+1,n}$ and it is called the *associated pencil* of the set. The following result forms a basis for the numerical computation of the GCD :

**Corollary 2.2** ([59]). *Let $T(s) = s\,W - \hat{A}\,W \in \mathbb{R}^{n\times k}[s]$ be the associated pencil of $\mathcal{P}_{h+1,n}$. If $v(s)$ is the GCD of $\mathcal{P}_{h+1,n}$ and $C_k(\cdot)$ denotes the $k^{th}$ compound matrix, then*

$$C_k(T(s)) = v(s) \cdot C_k(W)$$

The essence of the numerical implementation of the Matrix Pencil algorithm is the determination of the null space $\mathcal{N}_r(Q(\hat{A}, \hat{C}))$ and its nullity $n(Q(\hat{A}, \hat{C})) = \dim\{\mathcal{N}_r(Q(\hat{A}, \hat{C}))\}$. The computation of the approximate null space of a matrix is more important here, since we care about approximate solutions. The following result suggests one method for calculating the numerical $\varepsilon$-nullity of a matrix from its singular values.

**Theorem 2.10** ([18]). *For a matrix $A \in \mathbb{R}^{m\times n}$ and a specified tolerance $\varepsilon$, the numerical $\varepsilon$-nullity of $A$ is $n_\varepsilon(A) = \{$number of singular values of A $\leq \varepsilon\}$.*

The SVD can also provide a basis for the right null space. In the Matrix Pencil method the numerical $\varepsilon$-nullity defines the $\varepsilon$-GCD degree. More specifically, if $k = n_\varepsilon(Q(\hat{A}, \hat{C}))$, it can be proved [45] that, if $k = 0$, the set of polynomials is coprime. Otherwise, if $T(s) = s\,W - \hat{A}\,W = s\,W - \widetilde{W}$ is the associated pencil and $s\,W_a - \widetilde{W}_a,\ a \in Q_{k,n}$ is any minor of maximal order such that $\det(W_a) \neq 0$, then the determinant $\det(s\,W_a - \widetilde{W}_a)$ suggests an $\varepsilon$-GCD of degree $k$.

The above technique often involves the computation of many minors. Each one of these may lead to different polynomials of degree $k$, which form the $k^{th}$ compound matrix $C_k(T(s))$. Suppose that $B$ is the matrix, which is formed by the coefficients of the polynomials of $C_k(T(s))$. Then, it can be proved [45] that, if $B = \Lambda\ S\ J^t$ is the singular value decomposition of $B$, the polynomial, which derives from the first row of $J$, is the best representative of all the polynomial rows of $C_k(T(s))$. This polynomial can be accepted as an approximate $\varepsilon$-GCD of the original set, obtained by the Matrix Pencil algorithm for the specified tolerance $\varepsilon$.

---

**ALGORITHM 2.1. The standard MP Algorithm.**

Step 1 : Form the initial basis matrix $P := P_{h+1} \in \mathbb{R}^{(h+1)\times(n+1)}$.

Step 2 : Compute a base $M \in \mathbb{R}^{(n+1)\times(n-\rho+1)}$ for the right null space of $P$ and form $M_1 := \{M$ without the last row$\}$.

Step 3 : Compute a base $\hat{C} \in \mathbb{R}^{(\rho-1)\times n}$ for the left null space of $M_1$.
**If** rank$(\hat{C}) > 2$ **then**
let $P := \hat{C}$ and repeat Step 1.
**end if**

Step 4 : Form the companion matrix $\hat{A}$ and construct the observability matrix: $Q(\hat{A}, \hat{C}) = \left[\hat{C}^t, \hat{A}^t\hat{C}^t, \ldots, (\hat{A}^t)^{n-1}\hat{C}^t\right]^t \in \mathbb{R}^{n(\rho-1)\times n}$

Step 5 : Compute the SVD of $Q(\hat{A}, \hat{C}) = V\ \Sigma\ \overline{W}^t$.
Select a tolerance $\varepsilon$ according to the singular values of $Q(\hat{A}, \hat{C})$.

Step 6 : Let $k := n_\varepsilon(Q(\hat{A}, \hat{C}))$
**If** $k = 0$ **then**
gcd $= 1$ **quit**
**else**
Form $W \in \mathbb{R}^{n\times k}$ from the first $k$ columns of $\overline{W}$.
Form $T(s) := sW - \hat{A}W \in \mathbb{R}^{n\times k}[s]$

Step 7 : Compute the $k^{th}$ compound matrix $C_k(T(s))$.
Form $B$ from the polynomial rows of $C_k(T(s))$.
Compute the SVD of $B = \Lambda\ S\ J^t$.

Step 8 : Select the approximate GCD vector from the first column of $J$.
**end if**

---

Regarding the computational complexity of the algorithm, the SVD procedure generally requires $O(hn^2 + 6n^3)$ flops for the computation of the singular values and the right null space of an $h \times n$ matrix [27]. The multiplication of an $n \times n$ companion matrix with another $n \times q$ matrix demands $O(nq)$ flops. Thus, the construction of the observability matrix $Q(\hat{A}, \hat{C})$ requires about $O(n^2\rho)$ flops. Since, the computation of the $k^{th}$ compound matrix $C_k(T(s))$ may involve the evaluation of the determinant of too many minors, practically we accept the approximate solution given by anyone of them and this computation requires $O(k^3)$ flops for a $k \times k$ matrix. An optimization method can also be employed to refine the obtained solution.

In terms of numerical stability, the MP method requires two SVD calls and the construction of the observability matrix. Since the matrix $(\hat{A})^{(k)}$ is computed, the computation of the product $(\hat{A}^t)^{(k)}\hat{C}^t = (\hat{C}(\hat{A})^{(k)})^t$ is stable because the matrix $\hat{C}$ is orthonormal. For the last matrix multiplication it holds [49] :

$$fl(\hat{C}(\hat{A})^{(k)}) = \hat{C}(\hat{A})^{(k)} + E, \text{ with } \|E\|_2 \leq d^2 u_1 \|\hat{C}\|_2 \|\hat{A}^k\|_2 = d^2 u_1 \|\hat{A}^k\|_2$$

where $fl(\cdot)$ denotes the computed floating point number and $u_1$ is of order of unit round off. A more detailed analysis for the numerical stability of each step of the above algorithm is presented in [45, 49, 59].

### ▶ The Modified Resultant Matrix Pencil method

The Modified Resultant Matrix Pencil method (MRMP) [49, 72] is an improved version of the standard MP method. The MRMP algorithm is a variation of the standard MP algorithm which is based on the modified Sylvester matrix $S^*$ [72] in order to construct a different GCD pencil $Z(s)$ and specify any minor of maximal order, which gives the required GCD. This process is done symbolically by using symbolic-numeric computations. The exploitation of the properties of the modified Sylvester matrix results in the formulation of a faster and more effective algorithm based on the Matrix Pencil methodology.

---

**ALGORITHM 2.2. The MRMP Algorithm**

Step 1 : Define a basis $\widetilde{M}$ for the right nullspace of the modified Sylvester matrix $S^*$.

Step 2 : Define the Matrix Pencil $Z(s) = s\widetilde{M}_1 - \widetilde{M}_2$ for the Resultant set, where $\widetilde{M}_1$ and $\widetilde{M}_2$ are the matrices obtained from $\widetilde{M}$ by deleting the last and the first row of $\widetilde{M}$ respectively.

Step 3 : Compute any non-zero minor determinant $d(s)$ of $Z(s)$ and thus gcd $:= d(s)$.

---

The MRMP method requires [49, 72] :

$$O\left((n+p)^3\left(2\,log_2(n)-\frac{1}{3}\right)+(n+p)^2(2m\,log_2 n\,+p)+12k(n+p)^2\right)$$

flops, where $k$ is the number of the calls of the SVD step. The computed GCD is the exact GCD of a slightly perturbed set of the initial polynomials. The final error is $E = E_1 + E_2$, [49, 72] with

$$\|E_1\|_F \leq \varphi(n)\,u\,\|S\|_F \quad \text{and}$$

$$\|E_2\|_F \leq \Big(\varphi(n)+c(h,n)+c(h,n)\,\varphi(n)\,u\Big)\,u\,\|S\|_F$$

where $u$ is the unit round off error, $\varphi(n)$ is a slowly growing function of $n$ [18] and $c(h,n)$ is a constant depending on $h$, $n$.

### 2.3.2 Subspace-based methods

The subspace concept is actually very common among several methods for computing the GCD of many polynomials, including the Matrix Pencil method. The fundamental principle behind subspace methods is the following.

Let $\Phi_v$ be a Toeplitz matrix constructed from the coefficient vector $\underline{v}$ of the GCD $v(s)$. Since $\Phi_v$ has full column rank, if there are matrices $Y$ and $W$, such that $Y = \Phi_v \cdot W$ and $W$ has full row rank, then the left nullspace of $\Phi_v$ can be determined from $Y$. Since $\Phi_v$ has a Toeplitz structure, the vector $\underline{v}$ can be determined uniquely (up to a scalar) from the left nullspace of $\Phi_v$.

Based on this principle, a family of methods can be derived. The SVD process applied to a generalized Sylvester matrix is the basic tool for a subspace method, which allows the left nullspace of $\Phi_v$ to be found. A representative and rather simple algorithm, which approaches the GCD problem from the subspace concept directly, is presented in [64] and we shall refer to it as the *SS algorithm*.

Given a set $\mathcal{P}_{h+1,n}$ of univariate polynomials as defined by (1.1), the first two steps of the SS algorithm involves the construction of an $(h+1)(n+1)\times(2n+1)$ generalized Sylvester matrix $Y$ from the input polynomials and the computation of the left null space of the transposed $Y^t$ via SVD. If we denote by $U_0 \in \mathbb{R}^{(2n+1)\times k}$ the basis matrix for the computed left null space of $Y^t$ and $C$ is the $(2n+1)\times(2n+1-k)$ Toeplitz matrix of a degree $k$ polynomial with arbitrary coefficients, then the GCD vector is actually the unique (up to a scalar) solution of the system $U_0^t\,C = 0$, [64]. Obviously, the degree of the GCD is $k = \text{colspan}\{U_0\}$.

For the approximate GCD problem, an equivalent and more appropriate way to compute the GCD vector with the SS algorithm is to construct $k$ Hankel matrices $\tilde{U}_i \in \mathbb{R}^{(k+1)\times(2n+1-k)}$, $i = 1, \ldots, k$ from the columns of $U_0$, form the matrix $\tilde{U} = [\tilde{U}_1, \ldots, \tilde{U}_k] \in \mathbb{R}^{(k+1)\times k(2n+1-k)}$ and compute a basis matrix $V_0$ for the left null space of $\tilde{U}$ by singular value decomposition. The last column of $V_0$, which corresponds to the smallest singular value (expected to be zero), contains the $k+1$ coefficients of the GCD. The obtained GCD can be considered as an approximate $\epsilon$-GCD for a tolerance $\epsilon$ equal to the machine's numerical precision. However, for a different tolerance $\varepsilon$, we can select a singular value $\sigma_j$ from the singular value decomposition of $Y^t$ such that $\sigma_j > \varepsilon \cdot f(h,n)$ and $\sigma_{j+1} \leq \varepsilon$, [17], and compute an $\varepsilon$-GCD of degree $k' = 2n + 1 - j \neq k$.

The computational cost of the SS algorithm is dominated by the SVD of the generalized Sylvester matrix $Y^t$, which requires $O(2\,h^2n^3 + 5\,h^2n^2)$ flops, [27]. Additionally, the SVD calculation is numerically stable and therefore the algorithm behaves very well to inexact data.

### 2.3.3 Barnett's method

Barnett's GCD method [3, 5] is a well known method for computing the GCD of several polynomials through the construction of the companion matrix of a properly selected polynomial from the given set and the decomposition of a special controllability matrix. More precisely, given a set $\mathcal{P}_{h+1,n}$ with polynomials of the form:

$$\begin{aligned} a(s) &= s^n + a_{n-1}s^{n-1} + \ldots + a_1 s + a_0 \ , \ a_0 \neq 0 \\ b_i(s) &= b_{i,n-1}s^{n-1} + \ldots + b_{i,1}s + b_{i,0} \ , \quad i = 1, 2, \ldots, h \end{aligned}$$

the companion matrix $A$ of the monic polynomial $a(s)$ of degree $n$ is constructed and the following matrix polynomials are formed:

$$b_i(A) = b_{i,p}A^{n-1} + \ldots + b_{i,1}\,A + b_{i,0}\,I_n \ , \quad i = 1, 2, \ldots, h$$

Then, the next matrix is created:

$$R = [b_1(A), b_2(A), \ldots, b_h(A)]$$

and, *the degree of the GCD of the set $\mathcal{P}_{h+1,n}$ is $k = n - \rho(R)$*. This is the most important theoretical result that forms the basis of Barnett's method for computing the GCD of several polynomials [3]. The next theorem provides the means of creating an algorithmic procedure for the computations of the GCD.

**Theorem 2.11** ([3]). *If the rows of $R$ are denoted by $r_1, r_2, \ldots, r_n$, then $r_i$ for $i = k+1, k+2, \ldots, n$ are linearly independent if*

$$r_i = \sum_{j=k+1}^{n} x_{i,j}\, r_j \ , \quad i = 1, 2, \ldots, k \tag{2.24}$$

*Then, the unique monic GCD of $\mathcal{P}_{h+1,n}$ is $g(s) = s^k + c_1\, s^{k-1} + \ldots + c_k$, where $c_i = x_{k+1-i,k+1}$ for $i = 1, 2, \ldots, k$.*

Considering the computation of the GCD of more than two polynomials without restricting to pairwise computations, the above theorem provided for the first time an alternative to standard approaches based on Euclid's algorithm, since the GCD can be found in a single step by solving the equations (2.24). However, the method tends to be computationally ineffective for large sets of polynomials of high degree. An earlier comparison with other methods can be found in [62].

Barnett's method through Bezoutians [19] for the approximate GCD problem is a variation of Barnett's method using Bezout-like matrices and singular value decomposition, and suggests a very compact way of parametrising and representing the greatest common divisor of several univariate polynomials. For a given polynomial set $\mathcal{P}_{h+1,n}$ with inexact data, the algorithm proposed in [19] constructs an expanded Bezout-like matrix $B_P$ by the polynomials of the given set and computes its singular values $\sigma_i$. If there is an integer $k$ such that

$$\sigma_k > 2n \cdot f\big(\varepsilon, a(s), b_1(s), \ldots, b_h(s)\big) > \sigma_{k+1}$$

for a specified numerical accuracy $\varepsilon$, then an $\varepsilon$-GCD of degree $n-k$ can be obtained. The coefficients of the $n-k$ degree $\varepsilon$-GCD are computed by solving $n-k$ linear least-squares problems. In practice, the algorithm is carefully developed to solve the least-squares problem by the method of normal equations and thus reduce the problem solving symmetric positive definite linear systems of order $k \times k$. The algorithm in [19] combines numerically stable algebraic processes, but there is not an overall stability analysis which may provide bounds for the total numerical error produced by the algorithm.

### 2.3.4 Combined methods for certified approximate GCDs

One of the main issues in the approximate GCD problem is the proper estimation of the degree of the approximate GCD. Various techniques along with certification theorems ("gap" theorems) have been proposed for the degree of an approximate GCD. This approach generally involves singular value decomposition of Sylvester matrices (resultants). Among the Euclidean algorithms that compute exact GCDs, the subresultant version is claimed to be the most efficient, since it achieves a

balance between coefficient growth and computational complexity. However, the variants of Euclid's algorithm only supply a lower bound on the degree of the GCD. The efficient processing of the singular values of subresultant matrices leads to the establishment of a methodology that certifies the maximum-degree approximate GCD within a tolerance $\varepsilon$ [20]. Such methods and algorithms for the computation of certified approximate GCDs for sets of several polynomials via singular value decomposition and optimization techniques are described in [16, 17, 20].

Another recently presented GCD algorithm, designed for the computation of multiple roots of inexact polynomials and extended to the approximate case, is described in [84, 85]. The algorithm UVGCD in [84] is developed for a pair of univariate polynomials with inexact coefficients. More specifically, a sequence of Sylvester subresultants is constructed and the smallest singular values are calculated by using a modified QR decomposition [85]. When a singular value $\sigma_j \leq \varepsilon\sqrt{2j+2}$ occurs, then, for the given pair of polynomials, there exists an approximate GCD of degree $k = n - j$ within tolerance $\varepsilon > 0$. After estimating the degree $k$, an approximation $v_0(s)$ is obtained by solving a particular linear system, and the Gauss-Newton iteration is applied to refine and certify the solution. If $n$ is the maximum degree of the polynomials of the given set, the cost for solving the previously mentioned linear system is $O(n^3)$ flops and, due to a special QR updating strategy, the total flops for decomposing all Sylvester subresultant matrices is $O(n^3)$. Additionally, the iterative refinement process requires $O\big(\frac{1}{3}(n-k+1)^3\big)$ flops, [84, 85].

However, all these methods are developed and analysed for computing the GCD of two polynomials with inexactly known coefficients, based on the classical structure of Sylvester matrices [77]. But, although it is assumed that they can be efficiently extended to sets of more than two polynomials, there is no evidence that these methods can retain their efficiency to provide certified approximate GCDs for more than two polynomials, simultaneously. The basic process by which these methods are extended, involves their iterative application to two polynomials at a time, until all the polynomials of the set are processed. However, the efficiency and stability of such an iterative process is questioned, because in every iteration additional perturbations for the pairs of polynomials of all the previous steps are introduced indirectly and this accumulation might exceed the fixed tolerance, [67].

### 2.3.5 Methods for computing the nearest GCD

Another approach, which has similar characteristics with the previously described approaches to the approximate GCD problem for univariate polynomials, is to search for perturbed polynomials with a non-trivial common divisor, which are close enough to the input polynomials. This approach is usually referred to as the *nearest GCD problem* and involves the computation of the smallest real

perturbation which results in the least perturbed polynomials relative to the polynomials of the original set that have a non-trivial GCD. More precisely, given two polynomials $f, g \in \mathbb{R}[s]$ of respective degrees $m$ and $n$, and $r_1$, $r_2 \in (0, +\infty)$, we have to find polynomials $\hat{f}$, $\hat{g}$ with $\|\hat{f}\|^2 \leq r_1$, $\|\hat{g}\|^2 \leq r_2$ such that $f + \hat{f}$, $g + \hat{g}$ have a common real root and $\|\hat{f}\|^2 + \|\hat{g}\|^2$ is minimised. The effort concentrates on finding $a \in \mathbb{R}$ and $\phi$, $\gamma \in \mathbb{R}[s]$ of degrees $m-1$ and $n-1$ respectively, such that $f + \hat{f} = (s-a)\phi, \quad g + \hat{g} = (s-a)\gamma, \quad \|\hat{f}\| \leq r_1$, and $\|\hat{g}\| \leq r_2$. A study of this problem as well as its variations can be found in [52, 53].

In [67] the developed method involves the efficient processing of singular values of special generalised Sylvester matrices which correspond to the whole set of polynomials. The degree of the approximate GCD is also certified by a gap theorem which is based on the numerical properties of the singular values. Nevertheless, the computation of the approximate GCD by this method is not straightforward. The solution is given by a linear system of the form $S_y(\tilde{\mathcal{P}}) \cdot \underline{v} = 0$, where $\underline{v}$ denotes the unit vector associated to the smallest singular value of the generalised Sylvester $S_y(\mathcal{P})$, and $S_y(\tilde{\mathcal{P}})$ denotes the generalised Sylvester matrix of a perturbed set of polynomials $\tilde{\mathcal{P}}$, which is associated with the original set $\mathcal{P}$. Yet, although the theoretical results in [67] are valuable, there is no evidence for the numerical stability and performance of the corresponding algorithm.

Methods and algorithms for computing the nearest GCD and a certified $\varepsilon$-GCD are also presented in [86]. The proposed method refers to a pair of univariate polynomials with inexact coefficients and it is based on *Structured Total Least Norm* (STLN) for constructing the nearest Sylvester matrix of given lower rank. More specifically, for a given pair of polynomials $\{a, b\}$ and a positive integer $k$ the algorithm for the nearest GCD problem forms the Sylvester matrix $S$ of the input polynomials, $S = [B\, A]$, and solves the overdetermined linear system $AX \approx B$ by using STLN. A minimal Sylvester structured perturbation $[F\, E]$ is obtained, such that $B + F \in \text{range}(A + E)$, and the solution has Sylvester structure with rank $\leq n + m - k$. When $k > 1$, the algorithm uses the $k^{th}$ submatrix of $S$ rather than the whole matrix $S$, in order to avoid any stability problems. Finally, the output is a pair of perturbed polynomials $\{\tilde{a}, \tilde{b}\}$, with the Euclidean distance $\mathcal{N} = \|\tilde{a} - a\|^2 + \|\tilde{b} - b\|^2$ reduced to a minimum.

Given a tolerance $\varepsilon$, an $\varepsilon$-GCD can be computed from an appropriate Sylvester subresultant of a perturbed pair of polynomials, using an iterative process, which stops when the Euclidean distance $\mathcal{N} < \varepsilon$. Next, the obtained $\varepsilon$-GCD is tested by certification methods [17, 20] for the maximum degree. More useful details about the algorithm's function can be found in [86]. The algorithm is claimed to be efficient and stable for a pair of polynomials with inexact coefficients. However, there is not any reference to flop counts and furthermore it is not clear if the algorithm can be efficiently extended to work with sets of many polynomials.

### 2.3.6 The ERES method

The study of the invariance properties of the GCD under extended-row-equivalence and shifting operations [40] established the ERES methodology and led to the development of the ERES method [57] for computing the GCD of polynomials. The fundamental principle of the ERES methodology is that the GCD is a property of the row space of the basis matrix of the set of polynomials, and this property is also invariant under the symbolic operation of shifting. Based on this principle, the ERES method is an iterative matrix-based process where elementary row transformations and shifting of elements in the rows of a matrix are used in order to reduce a basis matrix to a unity rank matrix which provides the GCD. The method has the advantage that:

i) it can handle many polynomials simultaneously, without resorting to the successive two at a time computations of the Euclidean or other pairwise based approaches [11, 12, 17, 20, 53, 61, 84],

ii) it invokes a numerical termination criterion that allows the derivation of approximate solutions to the GCD computation problem, and

iii) it allows the combination of symbolic-numeric operations performed effectively in a mixture of numerical and symbolical steps.

The algorithm of the ERES method is based on numerically stable algebraic processes, such as Gaussian elimination with partial pivoting, normalisation, shifting, and partial singular value decomposition, which are applied iteratively on a basis matrix formed directly from the coefficients of the polynomials of the original set. The main target of the ERES algorithm is to reduce the number of the rows of the initial matrix and finally to end up to a unity rank matrix, which contains the coefficients of the GCD. The SVD method provides the ERES algorithm with an efficient termination criterion.

The ERES method is a simple and effective method which inherently has the potential to manipulate large sets of polynomials and define approximate solutions to the GCD problem. Therefore, the ERES method is central to our study which primarily focuses on the theoretical aspects and the development of a new implementation for the method using modern computer algebra software packages, and extends its applications.

## 2.4 Methods for the computation of the LCM of polynomials

The problem of computing the LCM of polynomials has widespread applications and requires implementation of algorithms computing the GCD. From the appli-

cations in Control Theory viewpoint the GCD is linked with the computation of zeros of representations whereas LCM is connected with the derivation of minimal fractional representations of rational models [46]. Existing procedures for the computation of LCM rely on the standard factorisation of polynomials, computation of a minimal basis of a special polynomial matrix [6] and use of algebraic identities, GCD algorithms and numerical factorisation of polynomials [47].

In the case of two polynomials $t_1(s)$, $t_2(s)$ with LCM $p(s)$ and GCD $z(s)$, we have the standard identity that $t_1(s)\,t_2(s) = z(s)\,p(s)$, which indicates the natural linking of the two problems. For randomly selected polynomials, the existence of a nontrivial GCD is a nongeneric property [46, 82], but the corresponding LCM always exists. This suggests that there are fundamental differences between the two computational problems. In [47], the standard algebraic identity of LCM is generalised and this provides a symbolic procedure for LCM computation, as well as the basis for a robust numerical LCM algorithm that avoids the computation of roots of the corresponding polynomials and also leads to the definition of the *approximate* LCM when the data are given inexactly or there are computational errors. The essence of this procedure is that if $p(s)$, $z(s)$ are the product of the polynomials of the original set and $z(s)$ the GCD of a set of polynomials derived from the original set, then the LCM $m(s)$ may be computed as the factor in the factorisation $p(s) = z(s)\,m(s)$. The use of algorithms for computing the GCD such as the ERES and Matrix Pencil are important for this method of computation of the LCM of polynomials. Naturally, for approximate values of the GCD the order of approximation is defined as a factor of $p(s)$ and the computation of the approximate LCM $m(s)$ is then seen as the best way of completing the approximate factorisation, which is defined as the *optimal completion problem*.

An alternative approach to the computation of LCM, which is based on standard system theory concepts and avoids root finding, as well as use of the algebraic procedure and GCD computation, has been presented in [48]. The characterisation of LCM in [48] leads to an efficient computational procedure based on properties of controllability of a linear system, associated with the given set of polynomials, and also provides a procedure for the computation of the associated set of polynomial multipliers linked to LCM. For a given set of polynomials $\mathcal{P}$ a natural realization $S(A, \underline{b}, C)$ is defined by inspection of the elements of the set $\mathcal{P}$. It is shown that the degree $r$ of LCM is equal to the dimension of the controllable subspace of the pair $(A, \underline{b})$, whereas the coefficients of LCM express the relation of $A^r\underline{b}$ with respect to the basis of the controllable space. The companion form structure of A simplifies the computation of controllability properties and leads to a simple procedure for defining the associated set of polynomial multipliers of $\mathcal{P}$ with respect to LCM. A special feature of the algorithmic procedure is that

a number of possibly difficult steps are substituted by simple closed formulae derived from the special structure of the system. An overall algorithmic procedure is formulated for computing LCM and multipliers, which is based on standard numerical linear algebra procedures. The developed algorithm [48] provides a robust procedure for the computation of LCM and enables the computation of approximate values, when the original data have some numerical inaccuracies. In such cases, the algorithm computes an approximate LCM with degree smaller than the generic degree. In fact, a generic set of polynomials is coprime and thus their LCM is their product. The existence of an LCM with degree different than that of the product of polynomials occurs only when the given set of polynomials is not coprime.

## 2.5 Discussion

The numerical computation of GCD or LCM of sets of many polynomials has been considered so far by transforming it to an equivalent problem of constant matrix computations. The advantage of real matrix computations is that we can discuss the problem of approximate solutions. Several methods have been proposed for the exact GCD and approximate GCD. They rely on the Euclidean algorithm [70, 20, 61, 69], computations with subresultant matrices [17, 20, 67], various optimization techniques, least-squares computations and quadratic programming [16, 17, 52, 53], and matrix pencils [45, 59]. Other approaches are based on Padé approximation and the approximation of polynomial zeros [8, 63].

In the present study, we focus on the ERES method [40, 57, 58] for computing the GCD and particularly the "best" approximate GCD of a set of several real univariate polynomials with numerical inaccuracies in their coefficients. The ERES method is an iterative matrix-based method which inherently has the potential to manipulate large sets of polynomials and define approximate solutions to the GCD problem. Therefore, the ERES method is central to our study which primarily focuses on the theoretical aspects and the development of a new implementation for the method using modern computer algebra software packages, and extends its applications.

The evaluation of the numerical quality, or *strength of approximation* for GCD computations has been an important drawback for all matrix-based methods dealing simultaneously with many polynomials. A rigorous definition of the approximate GCD has been given in [21, 42] that allows the computation of the strength of approximation and sets up a framework for computing the *optimal approximate GCD*. This approach is based on recent results [21, 22] on the representation of the GCD of several polynomials in terms of the factorisation of the generalised resultant and a Toeplitz matrix representation of the GCD. These

results allow the parametrisation of all perturbations, which are required to make a selected approximate GCD an exact GCD of the perturbed set of polynomials. The evaluation of the strength of approximation is equivalent to an evaluation of a distance problem in a projective space and it is thus reduced to an optimization problem. However, this is a nonconvex optimization problem and therefore cannot be solved easily. In the sequel, an efficient implementation of the procedure for evaluating the strength of an approximate GCD will be given by exploiting some of the special aspects of the respective distance problem.

The new implementation of the ERES algorithm, which will be presented and analysed in the following chapters, combines in an optimal setup the symbolical application of rows transformations and shifting, and the numerical computation of an appropriate termination criterion, which can provide the required approximate solutions. This combination highlights the *hybridity* of the ERES method. However, the overall numerical stability of the method is a very important issue that needs special attention. For this reason, the theoretical investigation of the following problems is critical:

- The matrix representation of the Shifting operation for non-singular matrices.
- The overall matrix representation of the ERES method and the connection amongst the involved algebraic processes.
- The ERES representation of the remainder and quotient of the Euclidean division of two polynomials.

The results of the analysis of these problems, improve the theory of the ERES methodology and give the motivation to study other related algebraic problems, such as the approximate LCM problem of sets of several polynomials, where the basic concept of the ERES method can give new or alternative ways to treat them properly.

# Chapter 3

# The ERES method

## 3.1 Introduction

The ERES method is an iterative matrix-based method developed in [40, 57], which exploits the invariance of the GCD of a set of several polynomials under *Extended-Row-Equivalence and Shifting* operations (ERES operations). The main concept of this method is to transform a basis matrix, formed directly from the coefficients of the polynomials of a given set, into a simpler matrix containing the vector of coefficients of the GCD using iterative elementary row transformations and partial column shifting. The theoretical and numerical properties of the ERES method have been studied in [40, 57, 58]. These properties reveal the advantage of the ERES method to handle large sets polynomials and to invoke an efficient termination criterion that allows the computation of approximate solutions when the initial data have numerical inaccuracies. From a theoretical point of view, ERES is a robust algebraic method which creates a special matrix "equivalence". However, the overall algebraic representation of the ERES method remained an open issue due to the iterative nature of the method and the luck of an algebraic expression for the Shifting transformation.

The main objective of this chapter is to highlight the theoretical value and robustness of the ERES method by establishing an algebraic connection between the initial basis matrix of a given set of several polynomials and the last matrix which occurs after the iterative application of the ERES operations and provides the GCD.

First, the definition and the most important properties of the ERES operations are presented. Then, the major issue of having an algebraic representation of the Shifting operation, applied to nonsingular matrices (Matrix Shifting), is analysed and discussed thoroughly. The results from the study of the matrix Shifting transformation are used to introduce the overall algebraic representation of the ERES method and the ERES representation of the GCD of a set of several polynomials. Furthermore, a relation between the Euclidean division of two

polynomials and the ERES method is developed, which allows the introduction of new algebraic expressions for the remainder and quotient of the division of two polynomials by ERES transformations. Finally, an algorithm computing the remainder and the quotient of the division of a pair of univariate real polynomials by using the ERES operations is formed and its properties discussed.

## 3.2 Definition of the ERES operations

Let us consider the set of univariate polynomials

$$\mathcal{P}_{h+1,n} = \Big\{ \; a(s), b_i(s) \in \mathbb{R}[s],\ i = 1, 2, \ldots, h \text{ with } n = \deg\{a(s)\},\ p = \max_i\big(\deg\{b_i(s)\}\big) \le n \text{ and } h, n \ge 1 \Big\} \quad (3.1)$$

We represent the polynomials $a(s),\ b_i(s)$ with respect to the highest degrees $(n, p)$ as

$$\begin{aligned} a(s) &= a_n s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0\ ,\ a_n \neq 0 \\ b_i(s) &= b_{i,p} s^p + \ldots + b_{i,1} s + b_{i,0}\ ,\ i = 1, 2, \ldots, h \end{aligned} \quad (3.2)$$

The set $\mathcal{P}_{h+1,n}$ will be called an $(n, p)$-*ordered polynomial set.*

**Definition 3.1.** For any $\mathcal{P}_{h+1,n}$ set, we define a vector representative (vr), $\underline{p}_{h+1}(s)$ and a basis matrix $P_{h+1}$ represented as

$$\underline{p}_{h+1}(s) = [\, p_1(s), \ldots, p_{h+1}(s) \,]^t = [\, \underline{p}_1, \ldots, \underline{p}_{m-1}, \underline{p}_{h+1} \,] \cdot \underline{e}_n(s) = P_{h+1} \cdot \underline{e}_n(s)$$

where $P_{h+1} \in \mathbb{R}^{(h+1)\times(n+1)}$, $\underline{e}_n(s) = [1, s, \ldots, s^{n-1}, s^n]^t$ and $\underline{p}_i \in \mathbb{R}^{n+1}$ for all $i = 1, \ldots, h+1$.

The matrix $P_{h+1}$ is formed directly from the coefficients of the polynomials of the set $\mathcal{P}_{h+1,n}$ and it has the least possible dimensions.

**Definition 3.2.** If $c$ is the integer for which $\underline{p}_1 = \ldots = \underline{p}_{c-1} = \underline{0}$ and $\underline{p}_c \neq 0$, then $c = w(\mathcal{P}_{h+1,n})$ is called the *order* of $\mathcal{P}_{h+1,n}$ and $s^c$ is an elementary divisor of the GCD. The set $\mathcal{P}_{h+1,n}$ is considered to be a $c$-order set and will be called *proper* if $c = 0$, and *nonproper* if $c \ge 1$.

If we have a nonproper set $\mathcal{P}_{h+1,n}$ with $w(\mathcal{P}_{h+1,n}) = c$, then we can always consider the corresponding proper one $\mathcal{P}_{h+1,\,n-c}$ by dismissing the $c$ leading zero columns. Then

$$\gcd\{\mathcal{P}_{h+1,n}\} = s^c \cdot \gcd\{\mathcal{P}_{h+1,\,n-c}\} \quad (3.3)$$

Therefore, in the following and without loss of generality, we assume that $\mathcal{P}_{h+1,n}$ is proper.

**Definition 3.3** (ERES operations)**.** Given a set $\mathcal{P}_{h+1,n}$ of many polynomials with a basis matrix $P_{h+1}$ the following operations are defined [40] :

a) Elementary row operations with scalars from $\mathbb{R}$ on $P_{h+1}$.

b) Addition or elimination of zero rows on $P_{h+1}$.

c) If $\underline{a}^t = [0, \ldots, 0, a_l, \ldots, a_{n+1}] \in \mathbb{R}^{n+1},\ a_l \neq 0$ is a row of $P_{h+1}$ then we define as the *Shifting* operation

$$shf : \mathrm{shf}(\underline{a}^t) = [a_l, \ldots, a_{n+1}, 0, \ldots, 0] \in \mathbb{R}^{n+1}$$

By $\mathrm{shf}(\mathcal{P}_{h+1,n})$, we shall denote the set obtained from $\mathcal{P}_{h+1,n}$ by applying shifting to every row of $P_{h+1}$ (Matrix Shifting).

Type (a), (b) and (c) operations are referred to as *Extended-Row-Equivalence and Shifting (ERES) operations.*

*REMARK* 3.1. The ERES operations without applying the Shifting operation will be referred to as *ERE operations.*

The following theorem describes the properties characterising the GCD of any given $\mathcal{P}_{h+1,n}$.

**Theorem 3.1** ([40])**.** *For any set $\mathcal{P}_{h+1,n}$, with a basis matrix $P_{h+1}$, $\rho(P_{h+1}) = r$ and $\gcd\{\mathcal{P}_{h+1,n}\} = \phi(s)$ we have the following properties :*

*i) If $\mathcal{R}_P$ is the row space of $P_{h+1}$, then $\phi(s)$ is an invariant of $\mathcal{R}_P$ (e.g. $\phi(s)$ remains invariant after the execution of elementary row operations on $P_{h+1}$). Furthermore if $r = \dim(\mathcal{R}_P) = n + 1$, then $\phi(s) = 1$.*

*ii) If $w(\mathcal{P}_{h+1,n}) = c \geq 1$ and $\mathcal{P}^*_{h+1,n} = \mathrm{shf}(\mathcal{P}_{h+1,n})$, then*

$$\phi(s) = \gcd\{\mathcal{P}_{h+1,n}\} = s^c \cdot \gcd\left\{\mathcal{P}^*_{h+1,n}\right\}$$

*iii) If $\mathcal{P}_{h+1,n}$ is proper, then $\phi(s)$ is invariant under the combined ERES set of operations.*

The GCD of any set of polynomials is a property of the row space of the basis matrix of the set. This property indicates that not all polynomials are required for the computation of the GCD [40]. Thus, the computation of the GCD requires selection of a base that is best suited for such computations. The already known methods for finding bases for given sets of vectors are based on the fact that they virtually transform the original data by using mostly Gaussian or orthogonal techniques (Gram-Schmidt, Householder *etc*) [18, 27]. Evidently, they obtain new sets and amongst the new vectors they choose the required ones that span the

original set. Thus, the base will be consisted of vectors completely different from the given ones. Furthermore, the numerical transformation of the original data always introduce round-off errors which in many cases can affect the quality of the final results very badly, especially in nongeneric computations.

### ▶ The selection of the "best uncorrupted base"

The issue of selecting the best possible base from all those vectors provided by the rows of the basis matrix without transforming the original data is critical for nongeneric computations and this problem is referred to as the *selection of the best uncorrupted base*.

**Definition 3.4.** Let $\mathcal{A} = \{\underline{a}_1, \underline{a}_2, \ldots, \underline{a}_m\}$ be a set of $m$ vectors in $\mathbb{R}^n$. Then, a subset $\mathcal{B} = \{\underline{b}_1, \underline{b}_2, \ldots, \underline{b}_r\}$ of $\mathcal{A}$ with $r < m$ vectors is an *uncorrupted base* of $\mathcal{A}$, if $\mathcal{B}$ consists of the original vectors of $\mathcal{A}$, (i.e. $\underline{b}_j \in \{\underline{a}_1, \underline{a}_2, \ldots, \underline{a}_m\}$ for every $j = 1, 2, \ldots, r$), all $\underline{b}_j$ for $j = 1, 2, \ldots, r$ are linearly independent, and $\mathcal{B}$ spans $\mathcal{A}$.

The set $\mathcal{A}$ can be expressed in terms of a matrix $A = [\underline{a}_1, \underline{a}_2, \ldots, \underline{a}_m]^t \in \mathbb{R}^{m\times n}$. Then, the problem of finding an uncorrupted base for the set $\mathcal{A}$ is transferred into finding an uncorrupted base for the row space of the matrix $A$. Therefore, for a given set $\mathcal{P}_{h+1,n}$ with a basis matrix $P_{h+1}$ and $\mathcal{R}$ the row space of $P_{h+1}$, an uncorrupted base of $\mathcal{R}$ is defined by the rows of $P_{h+1}$ without being transformed. If vector orthogonality is used to characterise the "best" selection of an uncorrupted base, then the *best uncorrupted base* of $\mathcal{R}$ is defined from the rows of $P_{h+1}$ which are orthogonal. However, such a base is not uniquely defined [60] and a procedure for the selection of the "best orthogonal" (or "most orthogonal") subset of $\mathcal{R}$ requires an appropriate quantitative numeric indicator that defines the *degree of orthogonality* of the selected set of vectors.

Such a procedure for the selection of the best uncorrupted base of the row space of a matrix has been presented in [57] and aims at the construction of a base that contains vectors that are *mostly orthogonal*, i.e. they form a set with the highest degree of orthogonality. This method relies on the properties of the Gram matrix and uses tools from the theory of compound matrices (Section 2.2.1).

**Definition 3.5.** Let $\mathcal{A} = \{\underline{a}_1, \underline{a}_2, \ldots, \underline{a}_m\}$ be a set of $m$ given vectors $\underline{a}_i \in \mathbb{R}^n$, $i = 1, 2, \ldots, m$. The matrix defined by

$$G_{\mathcal{A}} = \begin{bmatrix} (\underline{a}_1 \cdot \underline{a}_1) & (\underline{a}_1 \cdot \underline{a}_2) & \ldots & (\underline{a}_1 \cdot \underline{a}_m) \\ (\underline{a}_2 \cdot \underline{a}_1) & (\underline{a}_2 \cdot a_2) & \ldots & (\underline{a}_2 \cdot \underline{a}_m) \\ \vdots & \vdots & \vdots & \vdots \\ (\underline{a}_m \cdot \underline{a}_1) & (\underline{a}_m \cdot \underline{a}_2) & \ldots & (\underline{a}_m \cdot \underline{a}_m) \end{bmatrix} \in \mathbb{R}^{m\times m}$$

where $(\underline{a}_i \cdot \underline{a}_j)$ denotes the inner product of the vectors $\underline{a}_i$, $\underline{a}_j$, is called the *Gram matrix* of $\mathcal{A}$ and the determinant $\det\{G_{\mathcal{A}}\}$ is called the *Grammian* of $\mathcal{A}$.

The Grammian provides us with an important criterion about the linear independence of vectors.

**Theorem 3.2** ([60]). *The vectors $\underline{a}_1, \underline{a}_2, \ldots, \underline{a}_m$ are linearly independent if and only if their Grammian is positive and not equal to zero.*

**Theorem 3.3** ([60]). *For any set $\mathcal{A} = \{\underline{a}_1, \underline{a}_2, \ldots, \underline{a}_m\}$ with $\|\underline{a}_i\|_2 = 1$ for all $i = 1, 2, \ldots, m$ we have*

$$0 \leq \det\{G_{\mathcal{A}}\} \leq 1$$

*where the left equality holds when the set is linearly dependent and the right holds when the set is orthogonal.*

**Definition 3.6.** If $A = [\underline{a}_1, \underline{a}_2, \ldots, \underline{a}_m]^t \in \mathbb{R}^{m\times n}$, then the *normalization* of $A$ is a matrix $A_N = [\underline{v}_1, \underline{v}_2, \ldots, \underline{v}_m]^t \in \mathbb{R}^{m\times n}$ with the property $\underline{v}_i = \frac{\underline{a}_i}{\|\underline{a}_i\|_2}$, $i = 1, 2, \ldots, m$.

The next proposition gives the outline of the procedure that defines an indicator of the degree of orthogonality for a given set of vectors and computes the most orthogonal uncorrupted base of the set (or the matrix).

**Proposition 3.1** ([46, 57]). *Let $A = [\underline{a}_1, \underline{a}_2, \ldots, \underline{a}_m]^t \in \mathbb{R}^{m\times n}$, $\rho(A) = r \leq \min\{m, n\}$, $A_N = [\underline{v}_1, \underline{v}_2, \ldots, \underline{v}_m]^t \in \mathbb{R}^{m\times n}$ the normalization of $A$. Suppose $G \in \mathbb{R}^{m\times n}$ the Gram matrix of the vectors $\underline{v}_1, \underline{v}_2, \ldots, \underline{v}_m$ and $C_r(G) = [c_{i,j}] \in \mathbb{R}^{\binom{m}{r}\times\binom{m}{r}}$ the $r^{th}$ compound matrix of $G$. If $c_{ii} = \det(G[a/a])$, $a = (i_1, i_2, \ldots, i_r) \in \mathbb{Q}_{r,m}$[1] is the maximal diagonal element of $C_r(G)$, then a most orthogonal uncorrupted base for the row space of $A$, consists of the vectors $\{\underline{a}_{i_1}, \underline{a}_{i_2}, \ldots, \underline{a}_{i_r}\}$.*

Obviously, the maximal diagonal element of the $r^{th}$ compound matrix of the Grammian $G$ defines the best degree of orthogonality. The main advantage of this procedure is that it does not alter the elements of the original basis matrix $P_{h+1}$. It just indicates the best (most orthogonal) combination of linearly independent rows of $P_{h+1}$ according to the largest diagonal element of an $r$-order compound matrix ($r = rank(P_{h+1})$) associated with the Gram matrix, which is created by the rows of $P_{h+1}$. The selected best combination of the most orthogonal, linearly independent row vectors forms a base for $\mathcal{P}_{h+1,n}$ which is represented by a new matrix $P_r \in \mathbb{R}^{r\times(n+1)}$.

### ▶ The formulation of the ERES method

From Theorem 3.1 it is evident that ERES operations preserve the GCD of any $\mathcal{P}_{h+1,n}$ and thus can be easily applied in order to obtain a modified basis matrix with much simpler structure. The successive application of these operations on a

[1] $\mathbb{Q}_{r,m}$ denotes the set of strictly increasing sequences of $r$ integers ($1 \leq r \leq m$) chosen from $1, 2, \ldots, m$.

basis matrix of a set of polynomials leads to the formulation of the ERES method for computing the GCD of a set of polynomials [57]. After successive applications of ERES operations on an initial basis matrix, the maximal degree of the resulting set of polynomials is reduced and after a finite number of steps the resulting matrix has rank 1. At this stage, the process is terminated and considering that all the arithmetic operations are performed accurately (symbolic-rational operations), any row of the last obtained matrix specifies the coefficients of the required GCD of the set.

Therefore, from a theoretical point of view, the ERES method in its simplest form consists of three basic procedures:

1. Computation of the best uncorrupted base for the set $\mathcal{P}_{h+1,n}$.
2. Application of elementary row operations to the processed matrix, which practically involves row reordering, triangularisation, and elimination of zero rows (ERE operations).
3. Application of the Shifting operation to the nonzero rows of the processed matrix.

The iterative application of the process of triangularisation and Shifting is actually the core of the ERES method and we shall refer to it as the *main procedure* of the method. Conversely, the computation of the best uncorrupted base of $\mathcal{P}_{h+1,n}$ is necessary only when the row dimension of $P_{h+1}$ is larger than its column dimension and it is performed only once before the main procedure in order to get a more concrete set of polynomials. The problem that will be considered next is the formulation of an algebraic expression which will represent the relation between the initial basis matrix $P_{h+1}$ and the final matrix, which occurs after the iterative application of the ERES operations.

The matrix $B_r \in \mathbb{R}^{r\times(h+1)}$, which corresponds to the selection of the best uncorrupted base of $P_{h+1}$, is actually a simple permutation matrix, which allows us to work with $r < h+1$ independent rows of $P_{h+1}$ and thus starting ERES with a matrix $P_r \in \mathbb{R}^{r\times(n+1)}$ of shorter dimensions.

$$P_r = B_r \cdot P_{h+1} \tag{3.4}$$

The ERE row operations, i.e. triangularisation, deletion of zero rows and reordering of rows, can be represented by a matrix $R \in \mathbb{R}^{r_1\times r}$ [18, 27], which converts the initial rectangular matrix $P_r$ into an upper trapezoidal form. However, the matrix representation of the Shifting operation is not straightforward. This problem has to do with the connectivity of the matrices, which are obtained after the process of triangularisation in each iteration of the main procedure of the ERES method.

In [57] and related work the problem of the matrix representation of the Shifting operation for real matrices remained open. Solving this problem is crucial for establishing an overall matrix representation of the ERES method which will allow us to study in more detail the numerical stability of the method not only for a single iteration of the main procedure of the method as in [57], but for all the performed iterations. Therefore, the problem that we will study in the following section is to find the simplest possible algebraic relation between a matrix and its shifted form.

## 3.3 The Shifting operation for real matrices

The Shifting operation is a special algebraic transformation which is not very common in the literature of algebra. In Definition 3.3 the Shifting operation was defined for real vectors as the permutation of the leading consecutive zeros of a vector at the end of the vector. Specifically, having a real vector

$$\underline{a}^t = [\underbrace{0, \ldots, 0}_{k \text{ elements}}, a_{k+1}, \ldots, a_n] \in \mathbb{R}^n,\ a_{k+1} \neq 0$$

the Shifting operation is defined as

$$shf : \mathrm{shf}(\underline{a}^t) = [a_{k+1}, \ldots, a_n, 0, \ldots, 0] \in \mathbb{R}^n$$

and we will simply refer to it as *vector Shifting*. Naturally, the definition of the Shifting operation can be extended to the case of real matrices.

**Definition 3.7.** Given a matrix $A = [\underline{a}_1^t, \underline{a}_2^t, \ldots, \underline{a}_n^t]^t \in \mathbb{R}^{m \times n}$, the Shifting operation for matrices is defined as the application of vector Shifting to every row of $A$. This transformation will be referred to as *matrix Shifting* and the shifted form of $A$ will be denoted by

$$\mathrm{shf}(A) \triangleq A^* = [\mathrm{shf}(\underline{a}_1^t), \mathrm{shf}(\underline{a}_2^t), \ldots, \mathrm{shf}(\underline{a}_n^t)]^t \in \mathbb{R}^{m \times n}$$

It is important to notice that the Shifting operation, as defined here, permutes the elements of a vector without changing their values and this is a basic requirement for the Shifting operation in our present study. Regarding the algebraic representation, the vector Shifting can be represented by the multiplication:

$$\mathrm{shf}(\underline{a}^t) = \underline{a}^t \cdot J_{k,n}$$

where $J_{k,n}$ is an appropriate $n \times n$ permutation matrix which is actually a square binary matrix that has exactly one entry 1 in each row and each column and zeros

elsewhere. Each such matrix represents a specific permutation of $k$ elements and for the vector Shifting it has the form:

$$J_{k,n} = \left[\begin{array}{c|c} O_{n-k} & I_k \\ I_{n-k} & O_k \end{array}\right] \in \mathbb{R}^{n\times n} \tag{3.5}$$

where $I_i$ denotes the $i \times i$ identity matrix and $O_i$ denotes the $i \times i$ zero matrix for $i = k,\, n-k$.

Although it is rather simple to represent the vector Shifting with a simple vector-matrix multiplication, it is not obvious how to represent the matrix Shifting transformation, because in general the application of vector Shifting to the rows of a matrix alters the structure of the columns in a non uniform way. The problem of representing the matrix Shifting by using an appropriate matrix-matrix multiplication is very challenging, especially when the modification of the original data is undesired. For the purposes of our study relating to the ERES method, we will investigate the problem of finding an algebraic relation between a real matrix and its shifted form in the class of upper trapezoidal matrices.

Upper trapezoidal matrices occur after the application of Gaussian elimination or other triangularisation methods and they have the following generic form:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} & \dots & a_{1n} \\ 0 & a_{22} & \dots & a_{2m} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & a_{mm} & \dots & a_{mn} \end{bmatrix} \in \mathbb{R}^{m\times n}, \quad m < n \tag{3.6}$$

Then, the shifted form of $A$, which is obtained by the matrix Shifting transformation as defined in Definition 3.7, is

$$A^* = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} & \dots & a_{1n} \\ a_{22} & \dots & a_{2m} & \dots & a_{2n} & 0 \\ \vdots & \vdots & \vdots & \iddots & \vdots & \vdots \\ a_{m\,m} & \dots & a_{m\,n} & 0 & \dots & 0 \end{bmatrix} \in \mathbb{R}^{m\times n}, \quad m < n \tag{3.7}$$

In order to simplify the problem, we will focus on finding an algebraic relation between a $2\times k$ matrix, with $k > 2$, and its shifted form. The proposed constructive method in the following proposition underlies the algebraic representation of matrix Shifting.

**Proposition 3.2.** *If $U \in \mathbb{R}^{2\times k}$, $k > 2$, is an upper trapezoidal matrix with rank $\rho(U) = 2$ and $U^* \in \mathbb{R}^{2\times k}$ is the matrix obtained from $U$ by applying the Shifting operation to its rows, then there exists a matrix $S \in \mathbb{R}^{k\times k}$ such that*

$$U^* = U \cdot S \tag{3.8}$$

*Proof.* Let

$$U = \begin{bmatrix} u_{1\,1} & u_{1\,2} & ... & u_{1\,k-1} & u_{1\,k} \\ 0 & u_{2\,2} & ... & u_{2\,k-1} & u_{2\,k} \end{bmatrix} \in \mathbb{R}^{2\times k} \tag{3.9}$$

with $u_{ii} \neq 0,\ i = 1, 2$ and the shifted matrix

$$U^* = \begin{bmatrix} u_{1\,1} & u_{1\,2} & ... & u_{1\,k-1} & u_{1\,k} \\ u_{2\,2} & u_{2\,3} & ... & u_{2\,k} & 0 \end{bmatrix} \in \mathbb{R}^{2\times k} \tag{3.10}$$

We can construct the matrix $S$ by following the process:

1. Construct the matrix

$$H = [I_k|J] \in \mathbb{R}^{k\times 2k} \tag{3.11}$$

where $I_k\ \in \mathbb{R}^{k\times k}$ is the $k^{th}$ identity matrix and $J$ is a permutation matrix of the form:

$$J = \begin{bmatrix} 0 & 0 & \dots & 1 \\ & & & 0 \\ & I_{k-1} & & \vdots \\ & & & 0 \end{bmatrix} \in \mathbb{R}^{k\times k} \tag{3.12}$$

which permutes the columns of the input matrix $U$ and gives the proper shifting to the $2^{nd}$ row of $U$.

2. Multiply the matrices $H$ and $U$ as follows:

$$U^{(1)} = U \cdot H = \tag{3.13}$$

$$= \left[\begin{array}{ccccc|ccccc} u_{1\,1} & u_{1\,2} & \dots & u_{1\,k-1} & u_{1\,k} & u_{1\,2} & u_{1\,3} & \dots & u_{1\,k} & u_{1\,1} \\ 0 & u_{2\,2} & \dots & u_{2\,k-1} & u_{2\,k} & u_{2\,2} & u_{2\,3} & \dots & u_{2\,k} & 0 \end{array}\right]$$

Hence, the matrix $U^{(1)}$ has the form:

$$U^{(1)} = \left[\begin{array}{c|c} \underline{v}^t_{11} & \underline{v}^t_{12} \\ \hline \underline{v}^t_{21} & \underline{v}^t_{22} \end{array}\right] \in \mathbb{R}^{2\times 2k} \tag{3.14}$$

where $\underline{v}^t_{ij} \in \mathbb{R}^k$. The diagonal vectors $\underline{v}^t_{ii},\ i = 1, 2$ represent the rows of the shifted matrix $U^*$. The next step is to find a way to extract those two vectors.

3. Denote by

$$U_{12} = \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix}$$

the first $2 \times 2$ submatrix of $U$. Since it is assumed that the diagonal elements of $U$ are $u_{ii} \neq 0$, the submatrix $U_{12}$ is invertible with inverse matrix:

$$U_{12}^{-1} = \begin{bmatrix} \dfrac{1}{u_{11}} & -\dfrac{u_{12}}{u_{11}\, u_{22}} \\ 0 & \dfrac{1}{u_{22}} \end{bmatrix} \tag{3.15}$$

and hence, the matrix $U$ is right invertible. The right inverse of $U$ can be the matrix:

$$\widetilde{U} = \begin{bmatrix} U_{12}^{-1} \\ O \end{bmatrix} \in \mathbb{R}^{k \times 2} \tag{3.16}$$

where $O \in \mathbb{R}^{(k-2)\times 2}$ is a matrix with zero elements.

4. Expand the previously defined matrix $H$ such that

$$\widetilde{H} = \left[ I_k \,|\, J \,|\, \widetilde{U} \right] \in \mathbb{R}^{k \times (2k+2)} \tag{3.17}$$

and multiply it by $U$ :

$$U^{(2)} = U \cdot \widetilde{H} = \left[\begin{array}{c|c|cc} \underline{v}_{11}^t & \underline{v}_{12}^t & 1 & 0 \\ \underline{v}_{21}^t & \underline{v}_{22}^t & 0 & 1 \end{array}\right] \in \mathbb{R}^{2 \times (2k+2)} \tag{3.18}$$

5. Add proper multiples of the last two columns to all the other columns of $U^{(2)}$ and eliminate the unnecessary entries. For this task, define the matrices:

$$\widehat{U} = \begin{bmatrix} u_{12} & u_{13} & \ldots & u_{1\,k} & u_{11} \\ 0 & u_{22} & \ldots & u_{2\,k-1} & u_{2\,k} \end{bmatrix} = \begin{bmatrix} \underline{v}_{12}^t \\ \underline{v}_{21}^t \end{bmatrix} \in \mathbb{R}^{2 \times k} \tag{3.19}$$

and

$$\widetilde{V} = \begin{bmatrix} I_k \\ I_k \\ -\widehat{U} \end{bmatrix} \in \mathbb{R}^{(2k+2)\times k} \tag{3.20}$$

Then,

$$U^* = U^{(2)} \cdot \widetilde{V} = U \cdot \widetilde{H} \cdot \widetilde{V} \tag{3.21}$$

and therefore, the shifting matrix is given by

$$S = \widetilde{H} \cdot \widetilde{V} \tag{3.22}$$

□

*REMARK* 3.2. a) The above constructive method requires the original matrix to have full rank. However, as it is obvious in step 3, the selection of the inverse matrix $\widetilde{U}$ is not unique. The main goal here is to create the $2 \times 2$ identity matrix, and for this task we can take at least $k-1$ different pairs of columns of $U$ to form its inverse. Furthermore, we can replace the matrix $O$ in $\widetilde{U}$ with any other randomly selected $(k-2) \times 2$ matrix. Evidently, these changes provide different shifting matrices which transform $U$ into its shifted form $U^*$. Therefore, the shifting matrix $S$ is not unique.

b) The result in Proposition 3.2 can also be applied to $2 \times k$ matrices with more than one consecutive zeros at the beginning of the second row of the given matrix, if we choose the matrices $H$ and $\widetilde{U}$ appropriately.

c) If the shifted matrix $U^*$ has full rank, the result of Proposition 3.2 can also be applied to itself and then the process is reversed. The shifting matrix $S$ is right invertible and it holds:

$$U = U^* \cdot S^{-1}$$

**Example 3.1.** In this example, we shall demonstrate the steps for constructing the shifted form of the matrix:

$$U = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \end{bmatrix}$$

which is the matrix :

$$U^* = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 0 \end{bmatrix}$$

We will follow the next steps, according to the proof of Proposition 3.2 :

1.

$$H = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{array}\right] \in \mathbb{R}^{3\times 6}$$

2.
$$U^{(1)} = U \cdot H = \left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 2 & 3 & 1 \\ 0 & 4 & 5 & 4 & 5 & 0 \end{array}\right] \in \mathbb{R}^{2\times 6}$$

3.
$$\widetilde{U} = \begin{bmatrix} 1 & -\frac{1}{2} \\ 0 & \frac{1}{4} \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 2}$$

4.
$$\widetilde{H} = \left[\begin{array}{ccc|ccc|cc} 1 & 0 & 0 & 0 & 0 & 1 & 1 & -\frac{1}{2} \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & \frac{1}{4} \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{array}\right] \in \mathbb{R}^{3\times 8}$$

$$U^{(2)} = U \cdot \widetilde{H} = \left[\begin{array}{ccc|ccc|cc} 1 & 2 & 3 & 2 & 3 & 1 & 1 & 0 \\ 0 & 4 & 5 & 4 & 5 & 0 & 0 & 1 \end{array}\right] \in \mathbb{R}^{3\times 8}$$

5.
$$\widetilde{V} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2 & -3 & -1 \\ 0 & -4 & -5 \end{bmatrix} \in \mathbb{R}^{8\times 3} \quad \text{and} \quad S = \widetilde{H}\cdot\widetilde{V} = \begin{bmatrix} -1 & -1 & \frac{5}{2} \\ 1 & 0 & -\frac{5}{4} \\ 0 & 1 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3}$$

The matrix $S$ is the shifting matrix and it really holds:

$$U \cdot S = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \end{bmatrix} \cdot \begin{bmatrix} -1 & -1 & \frac{5}{2} \\ 1 & 0 & -\frac{5}{4} \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 0 \end{bmatrix} = U^*$$

However, we could have the same result, if we had used the matrix :

$$S' = \begin{bmatrix} -\frac{37}{45} & -\frac{31}{45} & \frac{119}{45} \\ \frac{13}{9} & \frac{7}{9} & -\frac{8}{9} \\ -\frac{16}{45} & \frac{17}{45} & \frac{32}{45} \end{bmatrix}$$

where we have taken into account the pseudo-inverse matrix [18] of $U$. □

### ▶ Matrix Shifting for full rank upper trapezoidal matrices

The process that we used in Proposition 3.2 can be extended in the case of an upper trapezoidal matrix $A \in \mathbb{R}^{m\times n}$ with $m \leq n$ and $a_{ii} \neq 0$ for all $i = 1, 2, \ldots, m$ in the form (3.6).

**Definition 3.8.** a) If the matrix $A$ has the form:

$$A = \begin{bmatrix} \underline{a}_1^t \\ \underline{a}_2^t \\ \vdots \\ \underline{a}_m^t \end{bmatrix} \tag{3.23}$$

where $\underline{a}_i^t$ is the $i^{th}$ row of $A$, then we can define the matrices :

$$A_i = \begin{bmatrix} \underline{0} \\ \vdots \\ \underline{a}_i^t \\ \vdots \\ \underline{0} \end{bmatrix} \in \mathbb{R}^{m\times n}, \quad i = 1, 2, \ldots, m \tag{3.24}$$

so that

$$A = \sum_{i=1}^{m} A_i \tag{3.25}$$

where $\underline{0} \in \mathbb{R}^n$ is a zero vector.

b) We define the permutation matrices $J_i \in \mathbb{R}^{n\times n}$ for $i = 1, 2, \ldots, m$, so that every $J_i$ gives the appropriate shifting to each $A_i$ respectively. Therefore,

$$\mathrm{shf}(A) = \sum_{i=1}^{m} A_i\, J_i \tag{3.26}$$

Since $a_{11} \neq 0$, we note that $J_1 = I_n$, where $I_n$ is the $n \times n$ identity matrix.

If $A$ has full rank, then, since it is defined as an upper trapezoidal with $a_{ii} \neq 0$ for all $i = 1, \ldots, m$, it is right invertible. Let us denote its right inverse by $A_r^{-1} \in \mathbb{R}^{n\times m}$. The following theorem establishes the connection between a nonsingular upper trapezoidal matrix and its shifted form.

**Theorem 3.4.** *If $A \in \mathbb{R}^{m\times n}$, $2 \leq m < n$, is a non-singular upper trapezoidal matrix with rank $\rho(A) = m$ and $\mathrm{shf}(A) \in \mathbb{R}^{m\times n}$ is the matrix obtained from $A$ by applying Shifting to its rows, then there exists a square matrix $S \in \mathbb{R}^{n\times n}$ such that*

$$\mathrm{shf}(A) = A \cdot S$$

*The matrix $S$ will be referred to as the shifting matrix of $A$.*

*Proof.* Let $A^* = \mathrm{shf}(A)$. We shall use the notation described in Definition 3.8 and we will follow the next method to determine the shifting matrix $S \in \mathbb{R}^{n\times n}$.

1. Apply to the original matrix $A$ the block matrix

$$S^{(1)} = \left[\begin{array}{cccc} J_1 & \dots & J_m & A_r^{-1} \end{array}\right] \in \mathbb{R}^{n\times n(m+1)} \tag{3.27}$$

such that

$$A^{(1)} = A \cdot S^{(1)}$$

2. Multiply the matrix $A^{(1)}$ by the block matrix

$$S^{(2)} = \left[\begin{array}{ccc} I_n & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & I_n \\ (A_1 - A)\, J_1 & \dots & (A_m - A)\, J_m \end{array}\right] \in \mathbb{R}^{m(n+1)\times mn} \tag{3.28}$$

and hence,

$$A^{(2)} = A^{(1)} \cdot S^{(2)}$$

3. Multiply the matrix $A^{(2)}$ by the block matrix

$$S^{(3)} = \left[\begin{array}{c} I_n \\ \vdots \\ I_n \end{array}\right] \in \mathbb{R}^{mn\times n} \tag{3.29}$$

and hence,

$$A^{(3)} \triangleq A^* = A^{(2)} \cdot S^{(3)}$$

The final matrix $S = S^{(1)} \cdot S^{(2)} \cdot S^{(3)}$ has the form:

$$S = \sum_{i=1}^{m} \left(I_n - A_r^{-1}A + A_r^{-1}A_i\right) J_i \tag{3.30}$$

and satisfies the equation: $A^* = A \cdot S$ □

In the proof of Theorem 3.4 the right inverse matrix $A_r^{-1}$ of $A$ is not unique when $m < n$. Conversely, the pseudo-inverse matrix $A^\dagger \in \mathbb{R}^{n\times m}$ of $A$ can be uniquely determined by calculating the singular value decomposition of $A$ [27], such that

$$A\, A^\dagger = I_m$$

Therefore, an alternative expression of the previous representation (3.30) of the shifting matrix $S$ can be given, if we use the pseudo-inverse matrix of $A$. This is

$$S = \sum_{i=1}^{m} \left( I_n - A^{\dagger}A + A^{\dagger}A_i \right) J_i \tag{3.31}$$

The expression (3.31) is more appropriate for the numerical computation of the shifting matrix $S$.

**Example 3.2.** Consider the following randomly selected matrix:

$$A = \begin{bmatrix} 2 & -8 & 6 & 10 & -5 \\ 0 & 7 & -2 & 1 & 8 \\ 0 & 0 & 12 & -9 & 4 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}$$

According to (3.30), the corresponding shifting matrix is:

$$S_1 = \begin{bmatrix} -\frac{107}{28} & -\frac{437}{84} & -\frac{87}{7} & -\frac{122}{21} & -\frac{751}{42} \\ \frac{19}{14} & -\frac{73}{42} & -\frac{6}{7} & \frac{1}{21} & -\frac{23}{21} \\ \frac{7}{4} & -\frac{13}{12} & \frac{3}{2} & \frac{7}{6} & \frac{7}{6} \\ 1 & 0 & 2 & 2 & 2 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

but according to (3.31), the shifting matrix is:

$$S_2 = \begin{bmatrix} \frac{1517909}{1722612} & -\frac{144305}{574204} & \frac{115706}{143551} & \frac{459329}{430653} & \frac{709807}{861306} \\ \frac{728666}{430653} & -\frac{198896}{143551} & \frac{12239}{143551} & \frac{231368}{430653} & \frac{101936}{430653} \\ \frac{1200059}{861306} & -\frac{418847}{287102} & \frac{71221}{143551} & \frac{277816}{430653} & -\frac{108584}{430653} \\ \frac{55772}{143551} & -\frac{92417}{143551} & \frac{40064}{143551} & \frac{158752}{143551} & -\frac{62048}{143551} \\ -\frac{87779}{287102} & \frac{194685}{287102} & \frac{20032}{143551} & \frac{79376}{143551} & -\frac{31024}{143551} \end{bmatrix}$$

Both $S_1$ and $S_2$ shift the rows of $A$ properly, but they are very different. The computation of their Frobenius norm [18] shows that $\|S_1\|_F = 24.1376$ and $\|S_2\|_F = 4.0267$. Therefore, it seems that the matrix $S_2$, which is computed by using the concept of the pseudo-inverse matrix, is more well-balanced. □

If $A$ is a real upper trapezoidal matrix and $A^*$ denotes its shifted form, then Theorem 3.4 is also applicable to the shifted matrix $A^*$, provided that $A^*$ has full rank. However, the Shifting is an operation that alters the structure of a matrix. Therefore, even if the original matrix has full rank, the corresponding shifted matrix may not have full rank. An example is:

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 2 & 4 & 6 \end{bmatrix} \quad , \quad A^* = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 0 \\ 2 & 4 & 6 & 0 & 0 \end{bmatrix}$$

where $\rho(A) = \rho(A^*) = 3$ and

$$B = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 0 & 2 & 4 & 6 \end{bmatrix} \quad , \quad B^* = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 0 & 0 \\ 2 & 4 & 6 & 0 & 0 \end{bmatrix}$$

where $\rho(B) = 3$ and $\rho(B^*) = 2$.

In the case where both $A$ and its shifted form $A^*$ have full rank we obtain the following result.

**Corollary 3.1.** *If $A \in \mathbb{R}^{m\times n}$, $2 \leq m < n$, is a nonsingular upper trapezoidal matrix with rank $\rho(A) = m$ and $A^* \in \mathbb{R}^{m\times n}$ is the shifted matrix of $A$ with rank $\rho(A^*) = m$, then there exists an invertible matrix $S \in \mathbb{R}^{n\times n}$ with rank $\rho(S) = n$, such that*

$$A^* = A \cdot S \;\Leftrightarrow\; A = A^* \cdot S^{-1}$$

*where $S^{-1}$ denotes the inverse of $S$.*

The previous corollary can be easily proven by following the same steps as in the proof of Theorem 3.4. We only have to

i) change appropriately the set of permutation matrices $J_i$, $i = 1, 2, \ldots, m$ to achieve the proper shifting, and

ii) compute the inverse or pseudo-inverse of $A^*$.

Therefore, we conclude that the matrix Shifting of a nonsingular upper trapezoidal matrix is a reversible process, unless the shifted matrix is rank deficient.

*REMARK* 3.3. a) The shifted matrix $A^*$ has full rank if and only if the shifting matrix $S$ has full rank.

b) In general, for two matrices $A$, $B \in \mathbb{R}^{m\times n}$, $m \leq n$ with $\rho(A) = \rho(B) = m$, we have that

$$\text{shf}(A + B) \neq \text{shf}(A) + \text{shf}(B)$$

$$\text{shf}(A \cdot B) \neq \text{shf}(A) \cdot \text{shf}(B)$$

So far, we have analysed the matrix shifting transformation for nonsingular upper trapezoidal matrices and we have established for it a simple algebraic representation by using a matrix-matrix multiplication. This representation has a

key role in the overall algebraic representation of the ERES method, since in every iteration of the main procedure there is always a nonsingular upper trapezoidal matrix, which is formed after the application of the ERE operations.

## 3.4 The overall algebraic representation of the ERES method

As it is already mentioned, the ERES is an iterative matrix-based method where, until now, only the ERE operations (i.e. triangularisation, deletion of zero rows and reordering of rows) could be represented by a matrix $R \in \mathbb{R}^{r_1 \times (h+1)}$. With the introduction of the representation of the Shifting operation as a matrix product, which has been described in Theorem 3.4, it is now possible to form an algebraic expression representing not only a single iteration of the main procedure of the ERES method, but all the required iterations until a rank-1 matrix is reached. This leads to a new matrix representation which provides a link between the initial basis matrix and the last matrix, which gives the coefficients of the GCD. This algebraic relationship is described in the following theorem.

**Theorem 3.5** (The overall representation of the ERES method)**.** *Given a set $\mathcal{P}_{h+1,n}$ of $h+1$ real univariate polynomials of maximum degree $n \in \mathbb{N}$ and its basis matrix $P_{h+1} \in \mathbb{R}^{(h+1)\times(n+1)}$, the application of the ERES operations to $P_{h+1}$ results in a matrix $P_{r_1} \in \mathbb{R}^{r_1 \times (n+1)}$ with row dimension $r_1 \leq n+1$ and rank $\rho(P_{r_1}) = 1$, which satisfies the equation:*

$$P_{r_1} = R \cdot P_{h+1} \cdot S \tag{3.32}$$

*where $R \in \mathbb{R}^{r_1 \times (h+1)}$ and $S \in \mathbb{R}^{(n+1)\times(n+1)}$ represent the applied row transformations (ERE operations) and the application of the Shifting operation, respectively. The GCD of $\mathcal{P}_{h+1,n}$ is then represented by the equality:*

$$\gcd\{\mathcal{P}_{h+1,n}\} = \underline{e}_1 \cdot R \cdot P_{h+1} \cdot S \cdot \underline{e}_n(s) \tag{3.33}$$

*where $\underline{e}_1 = [1, 0, \ldots, 0] \in \mathbb{R}^{r_1}$ and $\underline{e}_n(s) = [1, s, s^2, \ldots, s^n]^t$ is a basis vector of $\mathbb{R}[s]$.*

*Proof.* Given a set $\mathcal{P}_{h+1,n}$ of $h+1 > 2$ polynomials and its basis matrix $P_{h+1}$, the ERES operations are applied to $P_{h+1}$ with the following order:

1. Construction of the permutation matrix $B_r \in \mathbb{R}^{r \times (h+1)}$, $r \leq \min\{h+1, n+1\}$, which indicates the best uncorrupted base of $\mathcal{P}_{h+1,n}$.

$$P^{(1)} = B_r \cdot P_{h+1} \tag{3.34}$$

2. Construction of the permutation matrix $J^{(1)} \in \mathbb{R}^{r\times r}$ which reorders the rows of the initial matrix such that the first row corresponds to the polynomial with the lowest degree in the set.

3. Application of elementary row transformations (ERE operations) by using an appropriate lower triangular matrix $L^{(1)}$.

4. Elimination of the zero rows by using a block matrix $Z^{(1)}$.

5. Application of the Shifting operation by using a proper square matrix $S^{(1)}$.

$$P^{(2)} = Z^{(1)} \cdot L^{(1)} \cdot J^{(1)} \cdot P^{(1)} \cdot S^{(1)} \tag{3.35}$$

If we set $R^{(1)} = Z^{(1)} \cdot L^{(1)} \cdot J^{(1)}$, then

$$P^{(2)} = R^{(1)} \cdot P^{(1)} \cdot S^{(1)} \tag{3.36}$$

The superscript "$(k)$", $k = 1, 2, \ldots$, is used in all matrices to indicate the number of iteration of the main procedure. The equation (3.36) represents the first complete iteration of the main procedure of the ERES method. The whole process terminates when the final matrix has rank equal to one and this can be practically achieved in less than $n + 1$ iterations. Therefore, after the $k^{th}$ iteration,

$$P^{(k+1)} = R^{(k)} \cdot P^{(k)} \cdot S^{(k)}, \quad k = 1, 2, \ldots \tag{3.37}$$

and if the final number of iterations is $\eta \in \mathbb{N}$, then

$$P^{(\eta+1)} = R^{(\eta)} \cdots R^{(1)} \cdot B_r \cdot P_{h+1} \cdot S^{(1)} \cdots S^{(\eta)} \;\Leftrightarrow$$

$$P_{r_1} = R \cdot P_{h+1} \cdot S$$

where $P_{r_1} = P^{(\eta+1)}$, $R = \left(\prod_{k=1}^{\eta} R^{(k)}\right) \cdot B_r$ and $S = \prod_{k=1}^{\eta} S^{(k)}$.

Obviously, the final matrix $P_{r_1}$ does not necessarily have the same dimensions as the initial matrix $P_{h+1}$ due to the frequent deletion of the produced zero rows during the main iterative procedure of the method and thus $r_1 < h + 1$.

Since the final matrix has rank equal to 1, every row gives the coefficients of the GCD. If we choose to acquire the GCD from the first row of $P_{r_1}$, then we may set $\underline{e}_1 = [1, 0, \ldots, 0] \in \mathbb{R}^{r_1}$ and $\underline{e}_n(s) = [1, s, s^2, \ldots, s^n]^t$ and the GCD is given by the equation :

$$\gcd\{\mathcal{P}_{h+1,n}\} = \underline{e}_1 \cdot R \cdot P_{h+1} \cdot S \cdot \underline{e}_n(s)$$

□

The next example demonstrates the application of the ERES method to a set of four polynomials.

**Example 3.3.** Consider the set of polynomials:

$$\mathcal{P}_{h+1,n} = \left\{ \begin{array}{rcl} a(s) & = & s^3 - 3s^2 + 4 \\ b_1(s) & = & 2s^2 - s - 6 \\ b_2(s) & = & s^2 - 3s + 2 \\ b_3(s) & = & 2s^2 - 2s - 4 \end{array} \right\}, \quad h = 3,\ n = 3$$

with $\gcd\{\mathcal{P}_{h+1,n}\} = s - 2$ and initial basis matrix :

$$P_{h+1} = \begin{bmatrix} 4 & 0 & -3 & 1 \\ -6 & -1 & 2 & 0 \\ 2 & -3 & 1 & 0 \\ -4 & -2 & 2 & 0 \end{bmatrix} \in \mathbb{R}^{4\times 4}, \quad \underline{e}_n(s) = \begin{bmatrix} 1 \\ s \\ s^2 \\ s^3 \end{bmatrix} \tag{3.38}$$

The rank of $P_{h+1}$ is $r = 3$ and the best uncorrupted base of $\mathcal{P}_{h+1,n}$ consists of the first, the third and the fourth rows of $P_{h+1}$. The following permutation matrix $B_r$,

$$B_r = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 4}$$

leads to the next transformation of the original basis matrix $P_{h+1}$ :

$$P^{(1)} = B_r \cdot P_{h+1} = \begin{bmatrix} 2 & -3 & 1 & 0 \\ -4 & -2 & 2 & 0 \\ 4 & 0 & -3 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 4} \tag{3.39}$$

The iterative main procedure of the ERES method will start with the matrix $P^{(1)}$. After two iterations of the main prcedure, the final matrix will have rank 1 and its rows give the vector of coefficients of the GCD. The matrix which represents all the necessary row operations has the form:

$$R = \begin{bmatrix} 1 & 0 & -\frac{1}{2} & \frac{3}{4} \\ 1 & 0 & \frac{3}{2} & \frac{7}{4} \end{bmatrix} \in \mathbb{R}^{2\times 4} \tag{3.40}$$

and the matrix which represents the application of the Shifting operation during the iterations has the form:

$$S = \begin{bmatrix} \frac{15}{8} & -\frac{3}{4} & \frac{5}{16} & \frac{1}{8} \\ \frac{7}{4} & -\frac{1}{2} & \frac{5}{8} & \frac{1}{4} \\ \frac{3}{2} & 0 & \frac{5}{4} & \frac{1}{2} \\ 1 & 1 & \frac{5}{2} & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \tag{3.41}$$

The final matrix is

$$P_{r_1} = \begin{bmatrix} -2 & 1 & 0 & 0 \\ -10 & 5 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{2\times 4}, \quad r_1 = 2 \tag{3.42}$$

and

$$P_{r_1} = R \cdot P_{h+1} \cdot S$$

If $\underline{e}_n(s) = [1, s, s^2, s^3]^t$ and $\underline{e}_1 = [1, 0]$, then the GCD of the set $\mathcal{P}_{h+1,n}$ can be expressed with the following relation:

$$\gcd\{\mathcal{P}_{h+1,n}\} = \underline{e}_1 \cdot R \cdot P_{h+1} \cdot S \cdot \underline{e}_n(s) = s - 2 \tag{3.43}$$

Obviously, the second row of $P_{r_1}$ can also provide us with the coefficients of the GCD, if we divide the elements by 5. □

Apart from its theoretical value, the overall algebraic representation of the ERES method serves as a tool for the study of the overall numerical stability of the entire method. More particularly, this representation will allow us to study in more detail the numerical stability of the method not only for a single iteration of its main procedure as in [57], but for all the performed iterations. The issues relating to this approach are discussed in Chapter 4.

## 3.5 The ERES representation of the polynomial Euclidean division

The Euclidean algorithm (also called Euclid's algorithm) is the oldest method for computing the GCD of two integer numbers or two polynomials with integer coefficients. It is named after the Greek mathematician Euclid, who described it in Books VII and X of his *Elements* [32], written in Alexandria of Egypt around 300 BC. The Euclidean algorithm has many theoretical and practical applications and it appears in several modern integer factorization algorithms. It is used in constructing continued fractions, in the Sturm chain method for finding real roots

of a polynomial and it is a basic tool for proving theorems in modern number theory. It is also used to solve Diophantine equations and of course there are algorithms for the computation of the GCD of two polynomials, which are based on Euclid's algorithm to perform the division of polynomials [11, 12, 62].

In this section, we shall consider the algebraic representation of the Euclidean algorithm for real polynomials using the ERES methodology. Specifically, the remainder from the division of two polynomials will be represented as a matrix product, where the matrices correspond to the applied ERES operations. The developed method will be referred to as the *ERES Division*.

Consider two real polynomials:

$$a\,(s) = \sum_{i=0}^{m} a_i\, s^i,\ a_m \neq 0 \quad \text{and} \quad b\,(s) = \sum_{i=0}^{n} b_i\, s^i,\ b_n \neq 0, \quad m, n \in \mathbb{N} \tag{3.44}$$

with degrees $\deg\{a(s)\} = m$ , $\deg\{b(s)\} = n$ respectively, and $m \geq n$.

**Definition 3.9.** We define the set

$$\mathcal{D}_{m,n} = \Big\{(a(s), b(s)) : a(s), b(s) \in \mathbb{R}[s],\ m = \deg\{a(s)\} \geq \deg\{b(s)\} = n\Big\}$$

For any pair $\mathcal{P} = (a(s), b(s)) \in \mathcal{D}_{m,n}$, we define a vector representative $\underline{p}(s)$ and a basis matrix $P_m$ represented as:

$$\underline{p}(s) = \begin{bmatrix} a(s) \\ b(s) \end{bmatrix} = \begin{bmatrix} \underline{a}^t \\ \underline{b}^t \end{bmatrix} \cdot \underline{e}^{'}_m(s) = P_m \cdot \underline{e}^{'}_m(s)$$

where $P_m \in \mathbb{R}^{2\times(m+1)}$, $\underline{a}^t = [a_m, \ldots, a_0] \in \mathbb{R}^{m+1}$, $\underline{b}^t = [\underbrace{0, \ldots, 0}_{m-n}, b_n, \ldots, b_0\,] \in \mathbb{R}^{m+1}$ and $\underline{e}^{'}_m(s) = [s^m, s^{m-1}, \ldots, s, 1]^t$.

The matrix $P_m$ is formed directly from the coefficients of the given polynomials $a(s)$ and $b(s)$ and it has the following form:

$$P_m = \begin{bmatrix} a_m & \ldots & a_{n+1} & a_n & \ldots & a_0 \\ 0 & \ldots & 0 & b_n & \ldots & b_0 \end{bmatrix} \tag{3.45}$$

If we have a pair of polynomials $\mathcal{P} = (a(s), b(s)) \in \mathcal{D}_{m,n}$, then, according to Euclid's algorithm,

$$\frac{a(s)}{b(s)} = \frac{a_m}{b_n}\, s^{m-n} + \frac{r_1(s)}{b(s)}$$

or

$$a(s) = \frac{a_m}{b_n}\, s^{m-n}\, b(s) + r_1(s) \tag{3.46}$$

This is the first and basic step of the Euclidean algorithm. The polynomial $r_1(s) \in \mathbb{R}[s]$ is given by

$$r_1(s) = \sum_{i=m-n}^{m-1} \left( a_i - \frac{a_m}{b_n} b_{i-(m-n)} \right) s^i + \sum_{i=0}^{m-n-1} a_i \, s^i \tag{3.47}$$

In the following, we will show that the remainder $r_1(s)$ can be computed by applying ERES operations to the basis matrix $P_m$ of the pair $\mathcal{P}$. The next corollary derives from Proposition 3.2 and will help us to represent the first step of the Euclidean division of two polynomials in terms of ERES operations.

**Corollary 3.2.** *If $P_m \in \mathbb{R}^{2\times(m+1)}$ is the basis matrix of a pair of real polynomials $\mathcal{P} = (a(s), b(s)) \in \mathcal{D}_{m,n}$, then $P_m^* \in \mathbb{R}^{2\times(m+1)}$ is the basis matrix of the pair $\mathcal{P}^* = (a(s), s^{m-n}\, b(s)) \in \mathcal{D}_{m,m}$ and there exists a matrix $S_{\mathcal{P}} \in \mathbb{R}^{(m+1)\times(m+1)}$ such that*

$$P_m^* = P_m \cdot S_{\mathcal{P}} \tag{3.48}$$

The matrix $P_m^*$ is the shifted form of $P_m$ :

$$P_m^* = \begin{bmatrix} a_m & \dots & a_{n+1} & a_n & \dots & a_0 \\ b_n & \dots & b_0 & 0 & \dots & 0 \end{bmatrix} \tag{3.49}$$

**Proposition 3.3** (The matrix representation of the first remainder of the Euclidean division)**.** *Applying the algorithm of the Euclidean division to a pair $\mathcal{P} = (a(s), b(s)) \in \mathcal{D}_{m,n}$ of real polynomials, there exists a polynomial $r_1(s) \in \mathbb{R}[s]$ with $deg\{r_1(s)\} < m$ such that*

$$a(s) = \frac{a_m}{b_n} s^{m-n} b(s) + r_1(s)$$

*Then, the remainder $r_1(s)$ can be represented in matrix form as:*

$$r_1(s) = \underline{v}^t \cdot E_1 \cdot \underline{e}_m^{'}(s)$$

*where $E_1 \in \mathbb{R}^{2\times(m+1)}$ is the matrix which occurs after the application of the ERES operations to the basis matrix $P_m$ of the pair $\mathcal{P}$, and $\underline{v} = [0,\ 1]^t$.*

*Proof.* If we consider the division $\frac{a(s)}{b(s)}$, then, according to Euclid's algorithm, there is a polynomial $r_1(s)$ with degree $0 \leq deg\{r_1(s)\} < m$ such that

$$r_1(s) = a(s) - \frac{a_m}{b_n} s^{m-n} b(s) \tag{3.50}$$

Then,

$$\begin{aligned} r_1(s) &= [0,\ 1] \cdot \begin{bmatrix} 0 & 1 \\ 1 & -\dfrac{a_m}{b_n} \end{bmatrix} \cdot \begin{bmatrix} a(s) \\ s^{m-n}\, b(s) \end{bmatrix} \\ &= [0,\ 1] \cdot \begin{bmatrix} 0 & 1 \\ 1 & -\dfrac{a_m}{b_n} \end{bmatrix} \cdot P_m^* \cdot \underline{e}_m^{'}(s) \end{aligned} \tag{3.51}$$

Using the result in Corollary 3.2, we will have that

$$r_1(s) = \underline{v}^t \cdot C \cdot P_m \cdot S_{\mathcal{P}} \cdot \underline{e}_m^{'}(s) \tag{3.52}$$

where $\underline{v}^t = [0,\ 1]$, $P_m$ is the basis matrix of the polynomials $a(s)$ and $b(s)$,

$$C = \begin{bmatrix} 0 & 1 \\ 1 & -\dfrac{a_m}{b_n} \end{bmatrix}$$

and $S_{\mathcal{P}}$ the respective shifting matrix. Therefore, there exists a matrix $E_1 \in \mathbb{R}^{2\times(m+1)}$ such that

$$E_1 = C \cdot P_m \cdot S_{\mathcal{P}} \quad \text{and} \quad r_1(s) = \underline{v}^t \cdot E_1 \cdot \underline{e}_m^{'}(s) \tag{3.53}$$

We consider now the basis matrix $P_m$ of the polynomials $a(s)$ and $b(s)$ as defined in (3.45) such as

$$\begin{bmatrix} a\,(s) \\ b\,(s) \end{bmatrix} = \begin{bmatrix} a_m & a_{m-1} & ... & a_{n+1} & a_n & a_{n-1} & ... & a_0 \\ 0 & 0 & ... & 0 & b_n & b_{n-1} & ... & b_0 \end{bmatrix} \cdot \underline{e}_m^{'}(s) \tag{3.54}$$

and we will show that the above matrix $E_1$ is produced by applying the ERES operations to the basis matrix $P_m$ of the polynomials $a(s)$ and $b(s)$. We follow the steps:

1. Apply Shifting to the rows of $P_m$. Let $S_{\mathcal{P}} \in \mathbb{R}^{(m+1)\times(m+1)}$, be the proper shifting matrix:

$$P_m^{(1)} = P_m \cdot S_{\mathcal{P}} = \begin{bmatrix} a_m & a_{m-1} & ... & a_{m-n+1} & a_{m-n} & a_{m-n-1} & ... & a_0 \\ b_n & b_{n-1} & ... & b_1 & b_0 & 0 & ... & 0 \end{bmatrix}$$

2. Reorder the rows of the matrix $P_m^{(1)}$. If $J$ is a column permutation matrix:

$$J = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{3.55}$$

then

$$P_m^{(2)} = J \cdot P_m^{(1)} = \begin{bmatrix} b_n & b_{n-1} & ... & b_1 & b_0 & 0 & ... & 0 \\ a_m & a_{m-1} & ... & a_{m-n+1} & a_{m-n} & a_{m-n-1} & ... & a_0 \end{bmatrix}$$

3. Apply stable row operations to $P_m^{(2)}$ (LU factorization [18, 27]).

   If

$$L = \begin{bmatrix} 1 & 0 \\ \frac{a_m}{b_n} & 1 \end{bmatrix} \tag{3.56}$$

   then the inverse of $L$ is

$$L^{-1} = \begin{bmatrix} 1 & 0 \\ -\frac{a_m}{b_n} & 1 \end{bmatrix} \tag{3.57}$$

   and therefore,

$$\begin{aligned} P_m^{(3)} &= L^{-1} \cdot P_m^{(2)} \\ &= \begin{bmatrix} 1 & 0 \\ -\frac{a_m}{b_n} & 1 \end{bmatrix} \cdot \begin{bmatrix} b_n & b_{n-1} & ... & b_0 & 0 & ... & 0 \\ a_m & a_{m-1} & ... & a_{m-n} & a_{m-n-1} & ... & a_0 \end{bmatrix} \\ &= \begin{bmatrix} b_n & b_{n-1} & ... & b_0 & 0 & ... & 0 \\ 0 & a_{m-1} - b_{n-1}\frac{a_m}{b_n} & ... & a_{m-n} - b_0\frac{a_m}{b_n} & a_{m-n-1} & ... & a_0 \end{bmatrix} \end{aligned}$$

   Note that the term $\frac{a_m}{b_n}$ emerges from the LU factorization.

The above process can be described by the following equation:

$$P_m^{(3)} = L^{-1} \cdot J \cdot P_m \cdot S_{\mathcal{P}} \tag{3.58}$$

which represents the steps of the ERES method. Obviously, we have

$$L^{-1} \cdot J = \begin{bmatrix} 1 & 0 \\ -\frac{a_m}{b_n} & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -\frac{a_m}{b_n} \end{bmatrix} = C$$

and thus, if we consider (3.53), we can conclude that $P_m^{(3)} = E_1$. □

The following theorem establishes the connection between the ERES method and the Euclidean division of two real polynomials.

**Theorem 3.6** (The matrix representation of the remainder of the Euclidean division)**.** *Applying the Euclidean algorithm to a pair* $\mathcal{P} = (a(s), b(s)) \in \mathcal{D}_{m,n}$ *of real polynomials, there are unique real polynomials* $q(s)$, $r(s)$ *with degrees* $\deg\{q(s)\} = m - n$ *and* $0 \leq \deg\{r(s)\} < n$ *respectively, such that*

$$a(s) = q(s) \cdot b(s) + r(s)$$

*and the final remainder* $r(s)$ *can be represented in matrix form as*

$$r(s) = \underline{v}^t \cdot E_\eta \cdot \underline{e}_n^{'}(s)$$

*where* $E_\eta \in \mathbb{R}^{2\times(n+1)}$ *is the matrix which results from the successive application of the ERES operations to the basis matrix* $P_m$ *of the pair* $\mathcal{P}$*, and* $\underline{v} = [0,\ 1]^t$*,* $\underline{e}_n^{'}(s) = [s^n, s^{n-1}, \ldots, s, 1]^t$*.*

*Proof.* Consider two polynomials $a(s)$ and $b(s)$ with degrees $m$, $n$, respectively, with $m > n$. The Euclidean division $\frac{a(s)}{b(s)}$ includes the following steps:

$$\begin{aligned}
a(s) &= l_1\, s^{m-n}\, b(s) + r_1(s) \\
r_1(s) &= l_2\, s^{k_1-n}\, b(s) + r_2(s) \\
&\vdots \\
r_i(s) &= l_{i+1}\, s^{k_i-n}\, b(s) + r_{i+1}(s) \\
&\vdots \\
r_{\eta-1}(s) &= l_\eta\, s^{k_{\eta-1}-n}\, b(s) + r_\eta(s)
\end{aligned}$$

where $r_i(s) \in \mathbb{R}[s]$ is a polynomial with degree $k_i = \deg\{r_i(s)\}$, $i = 1, 2, \ldots, \eta$ and $\eta$ is the total number of steps in Euclid's algorithm for which

$$\eta = m - n + 1$$

Normally, $k_i > n$ for $i = 1, 2, \ldots, \eta - 2$ and $k_{\eta-1} = n$, whereas $k_\eta < n$.

If $P_m \in \mathbb{R}^{2\times(m+1)}$ is the basis matrix of the pair $\mathcal{P} = (a(s), b(s))$, then, by the result in Proposition 3.3,

$$r_1(s) = \underline{v}^t \cdot E_1 \cdot \underline{e}_m^{'}(s)$$

where $E_1 \in \mathbb{R}^{2\times(m+1)}$ is the matrix, which occurs after the application of the ERES operations to the basis matrix $P_m$, and $\underline{v} = [0, 1]^t$. But, from (3.58) we have

$$E_1 = L_1^{-1} \cdot J \cdot P_m \cdot S_1 \tag{3.59}$$

where

$$L_1 = \begin{bmatrix} 1 & 0 \\ l_1 & 1 \end{bmatrix}$$

the matrix $S_1$ is the proper shifting matrix and $J$ is the column permutation matrix as defined in (3.55). If $P_{k_1} \in \mathbb{R}^{2\times(k_1+1)}$ is the basis matrix of the pair $(r_1(s), b(s))$, then there are shifting matrices $S_{E_1}$ and $S_2$ such that

$$P_{k_1} \cdot S_2 = J \cdot E_1 \cdot S_{E_1} \cdot Z_{k_1} \tag{3.60}$$

where the matrix $Z_{k_1} \in \mathbb{R}^{(m+1)\times(k_1+1)}$ is actually used to reduce the column dimension by deleting the last $m - k_1$ zero columns. Therefore, for the second step of the process we shall have:

$$\begin{aligned} E_2 &= L_2^{-1} \cdot J \cdot P_{k_1} \cdot S_2 \\ &\overset{(3.60)}{=} L_2^{-1} \cdot J \cdot J \cdot E_1 \cdot S_{E_1} \cdot Z_{k_1} \\ &= L_2^{-1} \cdot E_1 \cdot S_{E_1} \cdot Z_{k_1} \end{aligned}$$

Hence, the matrices $E_1$ and $E_2$ are linked and, generally,

$$E_i = L_i^{-1} \cdot E_{i-1} \cdot S_{E_{i-1}} \cdot Z_{k_{i-1}}, \quad i = 2, 3, \ldots, \eta \tag{3.61}$$

and

$$r_i(s) = \underline{v}^t \cdot E_i \cdot \underline{e}'_{k_{i-1}}(s), \quad i = 2, 3, \ldots, \eta \tag{3.62}$$

The final matrix $E_\eta$ corresponds to the remainder $r(s)$ of the Euclidean division $\frac{a(s)}{b(s)}$ and thus it holds:

$$r(s) \triangleq r_\eta(s) = \underline{v}^t \cdot E_\eta \cdot \underline{e}'_n(s) \tag{3.63}$$

□

*REMARK* 3.4. In the proof of the previous theorem we notice that the terms $l_i,\ i = 1, 2, \ldots, \eta$ that we obtain from the matrices $L_i$ respectively, give the coefficients of the polynomial $q(s)$, which is actually the quotient of the division $\frac{a(s)}{b(s)}$. Specifically, $\underline{q} = [q_{m-n}, \ldots, q_0]^t$ with $q_i = l_{m-n+1-i}$, for $i = 0, 1, \ldots, m-n$ and

$$q(s) = \sum_{i=0}^{m-n} q_i\, s^i = \sum_{i=0}^{m-n} l_{m-n+1-i}\, s^i \tag{3.64}$$

Therefore, given two polynomials $a(s),\ b(s) \in \mathbb{R}[s]$ with degrees $\deg\{a(s)\} = m$,

$\deg\{b(s)\} = n$ respectively and $m > n$, we can compute the Euclidean division:

$$\frac{a(s)}{b(s)} = q(s) + \frac{r(s)}{b(s)}$$

by applying ERES operations to their basis matrix, iteratively. The final matrix gives the coefficients of the remainder polynomial $r(s)$ of the division $\frac{a(s)}{b(s)}$.

**Definition 3.10.** Given two polynomials $a(s),\ b(s) \in \mathbb{R}[s]$ with degrees $\deg\{a(s)\} = m$, $\deg\{b(s)\} = n$, respectively, and $m > n$ the transformation

$$\begin{bmatrix} a(s) \\ b(s) \end{bmatrix} \overset{ERES}{- - - - - >} \begin{bmatrix} r(s) \\ b(s) \end{bmatrix} \tag{3.65}$$

represents the Euclidean division of two polynomials using ERES operations and is referred to as *ERES Division*.

The following algorithm, termed *ERES Division algorithm*, corresponds to the transformation (3.65) and represents the division of a pair of real univariate polynomials using ERES operations.

---

**ALGORITHM 3.1. The ERES Division Algorithm**

Input : $a(s) = \underline{a}^t \cdot \underline{e}'_m(s), \quad b(s) = \underline{b}^t \cdot \underline{e}'_n(s), \quad m > n$

Step 1 : Form the basis matrix $P_m = [\underline{a}^t,\ \underline{b}^t]^t \in \mathbb{R}^{2\times(m+1)}$.
Set $\eta := m - (n-1)$.

Step 2 : Apply Shifting to $P_m$.
$P := P_m \cdot S$

**For** $i = 1, 2, \ldots, \eta$ **do**

Step 3 : Reorder the rows of $P$.
$P := J \cdot P$

Step 4 : Apply elementary row operations to $P$ (Gaussian elimination).
$P := L_i \cdot P$
Save the (2,1) element of $L_i$.
$q_{\eta-i} := -l^i_{21}$
**If** $i < \eta$ **then**

Step 5 : Apply Shifting to $P$.
$P := P \cdot S_i$

Step 6 : Delete the last zero columns of $P$.
$P := P \cdot Z_i$

Step 7 : Reorder the rows of $P$.

$P := J \cdot P$

**end if**

**end for**

Step 8 : Reorder the rows of $P$.

$P := J \cdot P$

Output : $P = [\,\underline{r}^t,\ \underline{b}^t\,]^t \in \mathbb{R}^{2\times(n+1)}$.

$\underline{q} = [\,q_{\eta-1}, \ldots, q_1, q_0\,]^t \in \mathbb{R}^{\eta}$

---

► **Computational complexity**

The above algorithm requires :

$$\begin{aligned} fl(m,n) &= (n+2)(m-n+1) \\ &= m(n+2) - (n^2+n-1) \end{aligned} \tag{3.66}$$

operations (additions or multiplications) to produce the final result. These operations mainly relate to Gaussian elimination, because in a software programming environment the reordering of the rows, the deletion of columns and the Shifting operation can be implemented without matrix multiplications, since it is a simple matter of changing the position of the elements.

In the following example, we will demonstrate the steps of the ERES Division Algorithm 3.1.

**Example 3.4.** Consider two real polynomials with the following symbolic form:

$$\begin{aligned} a(s) &= a_3\,s^3 + a_2\,s^2 + a_1\,s + a_0, \quad a_3 \neq 0, \qquad deg\{a(s)\} = m = 3 \\ b(s) &= b_2\,s^2 + b_1\,s + b_0, \qquad\qquad\ \ b_2 \neq 0, \qquad deg\{b(s)\} = n = 2 \end{aligned}$$

Then $\underline{a} = [\,a_3, a_2, a_1, a_0\,]^t$ and $\underline{b} = [\,b_2, b_1, b_0\,]^t$ and the basis matrix has the form:

$$P_m = \begin{bmatrix} a_3 & a_2 & a_1 & a_0 \\ 0 & b_2 & b_1 & b_0 \end{bmatrix} \in \mathbb{R}^{2\times 4}$$

Using $P_m$ as the initial matrix, the ERES Division algorithm will perform

$$\eta = m - n + 1 = 2$$

iterations of the steps 3 to 7 in order to compute the remainder $r(s)$ and the quotient $q(s)$ of the division $\frac{a(s)}{b(s)}$. Then,

*Step 2 : Shifting*

$$P := P_m \cdot S = \begin{bmatrix} a_3 & a_2 & a_1 & a_0 \\ b_2 & b_1 & b_0 & 0 \end{bmatrix}$$

*Iteration 1 :*

*Step 3 : Row reordering*

$$P := J \cdot P = \begin{bmatrix} b_2 & b_1 & b_0 & 0 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix}$$

*Step 4 : Gaussian Elimination*

$$P := L_1 \cdot P = \begin{bmatrix} b_2 & b_1 & b_0 & 0 \\ 0 & a_2 - \frac{b_1 a_3}{b_2} & a_1 - \frac{b_0 a_3}{b_2} & a_0 \end{bmatrix}$$

$$q_1 := \frac{a_3}{b_2}$$

*Step 5 : Shifting*

$$P := P \cdot S_1 = \begin{bmatrix} b_2 & b_1 & b_0 & 0 \\ a_2 - \frac{b_1 a_3}{b_2} & a_1 - \frac{b_0 a_3}{b_2} & a_0 & 0 \end{bmatrix}$$

*Step 6 : Deletion of zero columns*

$$P := P \cdot Z_1 = \begin{bmatrix} b_2 & b_1 & b_0 \\ a_2 - \frac{b_1 a_3}{b_2} & a_1 - \frac{b_0 a_3}{b_2} & a_0 \end{bmatrix}$$

*Iteration 2 :*

*Step 3 : Row reordering*

$$P := J \cdot P$$

*Step 4 : Gaussian Elimination*

$$P := L_2 \cdot P = \begin{bmatrix} b_2 & b_1 & b_0 \\ 0 & \left(a_1 - \frac{b_0 a_3}{b_2}\right) - b_1 \left(\frac{a_2}{b_2} - \frac{b_1 a_3}{b_2^2}\right) & a_0 - b_0 \left(\frac{a_2}{b_2} - \frac{b_1 a_3}{b_2^2}\right) \end{bmatrix}$$

$$q_0 := \frac{a_2}{b_2} - \frac{b_1 a_3}{b_2^2}$$

*Step 8 : Row reordering*

$$P := J \cdot P = \begin{bmatrix} 0 & \left(a_1 - \dfrac{b_0 a_3}{b_2}\right) - b_1 \left(\dfrac{a_2}{b_2} - \dfrac{b_1 a_3}{b_2^2}\right) & a_0 - b_0 \left(\dfrac{a_2}{b_2} - \dfrac{b_1 a_3}{b_2^2}\right) \\ b_2 & b_1 & b_0 \end{bmatrix}$$

Finally, we obtain a matrix which contains the coefficients of the remainder $r(s)$ in its first row. The final matrix has the form:

$$P = \begin{bmatrix} 0 & r_1 & r_0 \\ b_2 & b_1 & b_0 \end{bmatrix} \in \mathbb{R}^{2\times 3}$$

Therefore, the vector of coefficients of the remainder $r(s)$ is

$$\underline{r} = \begin{bmatrix} 0 \\ r_1 \\ r_0 \end{bmatrix} = \begin{bmatrix} 0 \\ \dfrac{a_1 {b_2}^2 - b_2 a_3 b_0 - b_1 a_2 b_2 + a_3 {b_1}^2}{{b_2}^2} \\ \dfrac{a_0 {b_2}^2 - b_0 a_2 b_2 + b_0 a_3 b_1}{{b_2}^2} \end{bmatrix} \tag{3.67}$$

and

$$r(s) = \frac{a_1 {b_2}^2 - b_2 a_3 b_0 - b_1 a_2 b_2 + a_3 {b_1}^2}{{b_2}^2}\, s + \frac{a_0 {b_2}^2 - b_0 a_2 b_2 + b_0 a_3 b_1}{{b_2}^2} \tag{3.68}$$

Simultaneously, we obtain the coefficients of the quotient $q(s)$ during the process of Gaussian elimination in the fourth step of the ERES Division algorithm. Here, we will have:

$$\underline{q} = \begin{bmatrix} q_1 \\ q_0 \end{bmatrix} = \begin{bmatrix} \dfrac{a_3}{b_2} \\ \dfrac{a_2 b_2 - a_3 b_1}{{b_2}^2} \end{bmatrix} \tag{3.69}$$

and therefore,

$$q(s) = \frac{a_3}{b_2}\, s + \frac{a_2 b_2 - a_3 b_1}{{b_2}^2} \tag{3.70}$$

If we consider now the numeric polynomials:

$$\begin{aligned} a(s) &= 3\,s^3 - 2\,s^2 + 4\,s - 3, & m = \deg\{a(s)\} = 3 \\ b(s) &= s^2 + 3\,s + 3, & n = \deg\{b(s)\} = 2 \end{aligned}$$

with

$$\begin{array}{rcl} \underline{a} & = & [\, a_3, a_2, a_1, a_0 \,]^t \quad = \quad [3, -2, 4, -3]^t \\ \underline{b} & = & [\, b_2, b_1, b_0 \,]^t \quad = \quad [1, 3, 3]^t \end{array}$$

and if we substitute the terms $a_i$ and $b_j$ into the formulae (3.68) and (3.70), then

$$r(s) = 28\,s + 30 \quad \text{and} \quad q(s) = 3\,s - 11$$

and thus the final rational expression of the division $\frac{a(s)}{b(s)}$ can be written as:

$$\frac{3\,s^3 - 2\,s^2 + 4\,s - 3}{s^2 + 3\,s + 3} = (3\,s - 11) + \frac{28\,s + 30}{s^2 + 3\,s + 3}$$

□

## 3.6 Discussion

In this chapter the basic definitions and properties of the ERES method for computing the GCD of a set of several polynomials has been given. The main objective was to introduce a general algebraic expression which represents the application of the ERES method to a basis matrix of a given set of polynomials and the consequent ERES representation of its GCD. The formulation of such an algebraic relation has as prerequisite the representation of the matrix Shifting transformation as a matrix product. Therefore, a thorough study of the properties of the Shifting operation, applied to nonsingular matrices, was presented and has resulted in the introduction of a proper algebraic relationship equation between a nonsingular upper trapezoidal matrix and its shifted form (Theorem 3.4). The obtained representation of the matrix Shifting has a key role in finding an algebraic relation between the initial basis matrix of a given set of several polynomials and the last rank-1 matrix, which occurs after the iterative application of the ERES operations and contains the vector of coefficients of the GCD (Theorem 3.5). Moreover, the study of the properties of the ERES method led to the investigation of the link between the ERES operations and the Euclidean division of two polynomials, which brought about: a) an ERES representation of the remainder and quotient of the division of two polynomials, and b) the development of an ERES-based algorithm for computing the quotient and the remainder of the division of two polynomials, which is referred to as the ERES Division algorithm. The developed ERES matrix representation of the Euclidean division suggests that the ERES method is actually the equivalent of Euclid's algorithm for more than two polynomials.

# Chapter 4

# The hybrid implementation of the ERES method for computing the GCD of several polynomials

## 4.1 Introduction

The main subject of this chapter is the development of an appropriate algorithm for computing the greatest common divisor of sets of several real univariate polynomials derived from the ERES methodology. A numerical algorithm of the ERES method has been developed in [57, 58]. This *numerical ERES algorithm* was originally designed to be implemented in a typical finite precision computational environment using numerically stable algebraic processes. The analysis of the produced results immediately showed that the ERES algorithm was capable of handling large sets of polynomials quite effectively. The main advantage of the ERES algorithm over other existing GCD algorithms is its ability to reduce the amount of data during the processing, which results in fast data processing and lower usage of computer memory. However, the iterative nature of the algorithm and the use of finite precision arithmetic often caused an undesirable accumulation of rounding errors, which affected the quality of the results very badly. To prevent these catastrophic results, additional parameters (numerical tolerances $\varepsilon_G, \varepsilon_t$) were introduced in the ERES algorithm in order to keep the data bounded. Unfortunately, the proper value of those parameters was arbitrary chosen and evidently reduced the efficiency of the algorithm.

The above problems motivated the search for a new kind of implementation for the ERES method, which will improve its performance and reliability. A major step towards this direction is to use different types of arithmetic. Many modern mathematical programming environments offer rational or variable precision numeric types of arithmetic which allow the users to perform computations with greater accuracy. The benefits from the mixture of symbolic and numerical

computations, known as *hybrid computations*, are significant and thus hybrid computations are widespread nowadays.

In a hybrid arithmetic system both exact symbolic and numerical computations can be carried out simultaneously. Symbolic computations refer to arithmetic operations either with arbitrary variables or fractions of integers to represent the numerical input data. The symbolic computations that involve only numerical data in rational format are also referred to as *rational computations* and they are always performed in almost infinite accuracy, depending on the symbolic kernel of the programming environment. Conversely, numerical computations refer to arithmetic operations with numbers in floating-point format (decimal numbers). However, the accuracy of the performed numerical computations is limited to a specific number of decimal digits which gives rise to numerical rounding errors. Therefore, the different algebraic procedures, which form an algorithm, can be implemented independently either using exact rational computations, or finite precision numerical computations in order to increase the accuracy of the produced results and the overall performance of the algorithm. This kind of implementation is referred to as *hybrid implementation*.

In this chapter we shall analyse the development of the hybrid implementation of the ERES algorithm. This kind of implementation treats the problem of numerical error accumulation and data handling properly, and results in a numerically stable algorithm. Apart from the improvement of the numerical stability by using hybrid computations, a considerable effort has been made to enhance the termination criterion of the ERES algorithm by finding a simplified method for detecting a rank 1 matrix. The developed method (PSVD1 method) is based on the partial singular value decomposition method and enables the effective termination of the algorithm as well as the computation of different approximate solutions. Finally, the new ERES algorithm in its hybrid form is presented and its efficiency and performance are discussed thoroughly.

## 4.2 The basics of the ERES algorithm

In order to develop an effective numerical algorithm for the ERES method, the treatment of the following problems is required:

**P1** Selection of a base for the given set of polynomials.

**P2** Application of the ERES operations.

**P3** Development of a proper termination criterion.

**P4** Selection of the representative row containing the coefficients of the GCD.

The above requirements are actually the most essential parts of the ERES algorithm and their proper implementation determines the overall behaviour of the algorithm. In the context of numerical implementation in a floating-point computational environment, the problems P1–P4 can be handled as follows [46]:

P1: A base of a set of several polynomials usually consists of less polynomials than the original set. Therefore, it can be used in order to decrease the amount of the input data and potentially improve the performance of the developed ERES algorithm. There exists various methods for finding a base for a given set of vectors such as LU factorisation, QR decomposition and others [18, 27, 38]. Most of them are based on the fact that they transform the original data by using Gaussian or orthogonal techniques. Thus, the obtained base will consist of vectors completely different from the original ones. But, when dealing with nongeneric computations and especially when we are interested in evaluating the GCD of a given set, it is extremely important to begin the calculating process using the concrete set of data or a subset of it. Therefore, an "uncorrupted" base of this set is required, which can be found without transforming the original data and apparently avoiding the introduction of round-off errors even before the method starts.

A method for the selection of the "most orthogonal uncorrupted base" is proposed in [57]. This method relies on the properties of the Gram matrix and uses tools from the theory of compound matrices [56]. However, for large sets of polynomials this method becomes inefficient due to its high complexity [58, 59]. Alternatively, it is simpler to get a base for the original set of polynomials by applying a triangularisation method to the original basis matrix of the set, such as Gaussian elimination or Householder QR factorisation. Of course, these methods transform the data of the set and introduce round-off errors but, when orthogonal techniques are used, this error is not significant [81]. Yet again, it would be preferable to be avoided in order to get more reliable results.

P2: Having a basis matrix of a set of polynomials, the ERES method involves row addition or row multiplication, row switching, elimination of zero rows and Shifting. The most reliable and stable numerical tool for applying elementary row operations is the method of Gaussian elimination with partial pivoting. Thus, after each iteration an upper triangular or trapezoidal form of the basis matrix will be computed. The obtained matrix can be changed so that its first row corresponds to the least degree polynomial. Simultaneously, the matrix is scaled appropriately in order to retain the least degree polynomial in the first row during the partial pivoting. The Shifting operation is merely a permutation of the leading consecutive zero elements in a row. Since

Gaussian elimination preserve the structure of the first row of the matrix, after a finite number of iterations a quick reduction of the maximal degree of the original polynomials of the set can be achieved. We shall refer to this procedure as the *main procedure* of the ERES algorithm.

P3: The algorithm's termination criterion relies on the proper detection of the final unity rank matrix. However, when dealing with numerical computations and inaccuracies exist in the input data, the rank of a matrix must be specified according to an appropriate tolerance $\varepsilon > 0$ and it is referred to as *numerical $\varepsilon$-rank*. A simplified condition for the determination of the numerical $\varepsilon$-rank ($\rho_{\varepsilon(A)}$) and numerical $\varepsilon$-nullity ($n_{\varepsilon(A)}$) is given next [23].

**Theorem 4.1** ([57]). *For a matrix $A \in \mathbb{R}^{\mu\times\nu}$ and a specified accuracy $\varepsilon$ we have:*

*i) The numerical $\varepsilon$-rank of $A$ is given by:*

$$\rho_\varepsilon(A) = \{\textit{number of singular values of } A \textit{ that are} > \varepsilon\}$$

*ii) The numerical $\varepsilon$-nullity of $A$ is given by:*

$$n_\varepsilon(A) = \{\textit{number of singular values of } A \textit{ that are} \leq \varepsilon\}$$

*iii)* $\rho_\varepsilon(A) = \nu - n_\varepsilon(A)$

The above results suggest a method for calculating the numerical $\varepsilon$-rank and numerical $\varepsilon$-nullity of a matrix via singular value decomposition (SVD). To avoid further numerical complications, it is preferable to apply the SVD to a normalized matrix $A$ where the elements are bounded by unity [27]. The normalisation of a matrix is a numerically stable process [81] and, therefore, the detection of a rank-1 matrix in the ERES method can be based on the numerical computation of the singular values of an associated normalized matrix obtained at the end of each iteration of the main procedure. We shall refer to it as the *Rank-1 procedure* of the ERES algorithm. This property can be detected numerically according to the following theorem [57].

**Theorem 4.2** ([57]). *Let $A = [\underline{a}_1, \ldots, \underline{a}_\mu]^t \in \mathbb{R}^{\mu\times\nu}$, $\mu \leq \nu$, $\underline{a}_1 \neq 0$, $i = 1, \ldots, \mu$. Then for an appropriate accuracy $\varepsilon_t > 0$ the numerical $\varepsilon_t$-rank of $A$ equals to one ($\rho_{\varepsilon_t}(A) = 1$) if and only if the singular values $\sigma_\mu \leq \sigma_{\mu-1} \leq \cdots \leq \sigma_1$ of the normalization $A_N = [\underline{u}_1, \ldots, \underline{u}_\mu]^t \in \mathbb{R}^{\mu\times\nu}$, $\underline{u}_i = \dfrac{\underline{a}_i}{\|\underline{a}_i\|_2}$ of $A$ satisfy the conditions:*

$$|\sigma_1 - \sqrt{\mu}| \leq \varepsilon_t \textit{ and } \sigma_i \leq \varepsilon_t,\ i = 2, 3, \ldots, \mu$$

The tolerance $\varepsilon_t$ will be referred to as the *termination accuracy* of the ERES algorithm.

P4: When the computation of the final unity rank matrix is achieved, the problem that arises is the proper selection of the vector containing the coefficients of the GCD. Theoretically, every row of the last matrix gives the coefficients of the GCD unless numerical rounding error has been added during the process. The following proposition gives an alternative way to acquire the GCD vector.

**Proposition 4.1** ([57])**.** *Let $A = U \cdot \Sigma \cdot W^t$ be the singular value decomposition of a given matrix $A \in \mathbb{R}^{\mu \times \nu}$, $\rho(A) = 1$. Then a "best" rank one approximation to $A$ in the Frobenius norm is given by $A_1 = \sigma_1 \cdot \underline{u} \cdot \underline{w}^t$, where $\sigma_1$ is the largest singular value of $A$ and $\underline{u}$ and $\underline{w}$ are the first columns of the orthogonal matrices $U$ and $W$ of the singular value decomposition of $A$ respectively. The vector $\underline{w}$ is the "best" representative of the rows of matrix $A$ in the sense of the rank one approximation.*

A numerical algorithm of the ERES method has been presented and analysed in [57, 58]. Various tests on sets of a moderate number of polynomials showed that the numerical ERES algorithm behaves quite well producing sufficiently accurate results [58]. The main advantage of the ERES method is the quick reduction of the size of the processed matrix. However, there are cases where the iterative nature of the method acts as disadvantage. Theoretically, the number of iterations does not exceed the size of the maximum degree of the polynomials of a set. Practically, this number is much less than the maximum polynomial degree. However, it has been observed that several iterations are performed in cases of sets of polynomials with high polynomial degree. In these cases a basis matrix with column dimension much greater than its row dimension is formed and then the iterative application of Gaussian elimination with partial pivoting often causes an excessive accumulation of numerical rounding error. The Gaussian elimination with partial pivoting is a quite stable numerical method, but, although pivoting keeps the multipliers bounded by unity, the elements in the reduced matrices still can grow arbitrarily [18, 81] during the iterations. Thus, additional procedures that control the magnitude of the elements of the processed matrices according to a specific small accuracy $\varepsilon_G$ have been added to the numerical ERES algorithm. The *Gaussian accuracy* $\varepsilon_G$ and the *termination accuracy* $\varepsilon_t$ both influence the correctness of the achieved results, but the proper specification of these numerical tolerances in order to get reliable results cannot be easily determined by the user.

**► Motivation for the hybrid implementation of the ERES algorithm**

The proper specification of the Gaussian accuracy $\varepsilon_G$ and the termination accuracy $\varepsilon_t$ is crucial in defining reliable approximate GCDs. This fact motivated the present study for the development of a new improved ERES algorithm for the computation of an approximate GCD of a given set of several polynomials. The improvement of the existing numerical ERES algorithm is actually based on how these accuracies influence the data. The Gaussian accuracy $\varepsilon_G$ controls the magnitude of the elements of the processed matrix during the iterations of the main procedure of the ERES algorithm and its value is affected by

a) the numerical inaccuracy of the original input data, and

b) the produced numerical error from the process of Gaussian elimination with partial pivoting.

The latter can cause some serious complications in the determination of the proper value of the Gaussian accuracy. If we denote by $P^{(\kappa)}$ the matrix resulting after the $\kappa^{th}$ iteration of the main procedure of the ERES algorithm ($\kappa = 0, 1, 2, \ldots$ and $P^{(0)} := P_{h+1}$), then by following the equations (3.36) and (3.37) we have:

$$P^{(\kappa+1)} = Z^{(\kappa)} \cdot L^{(\kappa)} \cdot J^{(\kappa)} \cdot P^{(\kappa)} \cdot S^{(\kappa)} \tag{4.1}$$

where $J^{(\kappa)}$ reorders the rows of $P^{(\kappa)}$, $L^{(\kappa)}$ is a lower triangular matrix, $Z^{(\kappa)}$ deletes any zero rows, and $S^{(\kappa)}$ is the shifting matrix. The row reordering, the Shifting, and the deletion of rows can be considered as error-free transformations, since they do not alter the values of the data. Thus, the numerical error mainly comes from the application of the Gaussian elimination. The backward error analysis of the Gaussian elimination with partial pivoting [81] shows that the computed upper and lower triangular matrices $L^{(\kappa)}$ and $U^{(\kappa)}$ satisfy:

$$L^{(\kappa)} \cdot U^{(\kappa)} = P^{(\kappa)} + E^{(\kappa)} \tag{4.2}$$

$$\|E^{(\kappa)}\|_\infty \leq (n')^2 \, \rho \, \mathbf{u} \, \|P^{(\kappa)}\|_\infty \, , \quad \|L^{(\kappa)}\|_\infty \leq n' \tag{4.3}$$

where $E^{(\kappa)}$ is the error matrix and $n'$ the column dimension $P^{(\kappa)}$. The term $\rho$ denotes the growth factor, which is defined by

$$\rho = \frac{\max_{i,j,l} |p_{ij}^{(\kappa)(l)}|}{\max_{i,j} |p_{ij}^{(\kappa)}|} \tag{4.4}$$

where $p_{ij}^{(\kappa)(l)}$ are the elements of the matrix $P^{(\kappa)}$ during the $l^{th}$ step of the process of triangularisation and $p_{ij}^{(\kappa)}$ are the elements of the initial matrix $P^{(\kappa)}$ [18, 81].

The upper triangular matrix $U^{(\kappa)}$ will eventually give the next matrix $P^{(\kappa+1)}$ after the deletion of its zero rows and Shifting. As it is proven in Theorem 3.5,

the produced matrices $P^{(\kappa)}$ from every iteration are linked together. Therefore, the total numerical error from the process of the Gaussian elimination for $k \in \mathbb{N}$ iterations is

$$\widetilde{E} = \sum_{i=0}^{k} \left( \prod_{j=i}^{k} (L^{(j)})^{-1} \right) E^{(i)} \tag{4.5}$$

If we take into account the reduction of the size of the dimensions of the processed matrices ($n' < n+1$), an upper bound for the above total numerical error for $k$ iterations of the main procedure of the ERES algorithm can be established:

$$\|\widetilde{E}\|_\infty \leq k\,(n+1)^3\,\rho\,\mathbf{u}\,\|P_{h+1}\|_\infty \tag{4.6}$$

Conversely, the termination accuracy $\varepsilon_t$ characterises the approximate GCD and its value is affected by:

a) the accumulated numerical error from the main procedure of the algorithm,

b) the produced numerical error from the process of computing the singular values, according to the termination criterion in Theorem 4.2.

The singular value decomposition is applied to the processed matrix $P^{(\kappa)}$ when it is required. The preliminary stage in this algorithm is the bidiagonal reduction of $P^{(\kappa)}$ and in most bidiagonal reduction methods the error is expressed in the following form [18, 26, 27]:

$$P^{(\kappa)} + \delta P^{(\kappa)} = U\,B\,V^t\,, \tag{4.7}$$

$$\|\delta P^{(\kappa)}\|_2 \leq \mathbf{u}\,f(h', n')\,\|P^{(\kappa)}\|_2 \tag{4.8}$$

where $B$ is bidiagonal, $U$ and $V$ are orthogonal, $\mathbf{u}$ is the machine's precision and $f(h', n')$ is a modestly growing function of the dimensions of $P^{(\kappa)}$ [18, 27], where $h' < h+1$ and $n' < n+1$.

It is obvious that a major step towards the improvement of the numerical stability of the ERES algorithm is to eliminate the accumulation of numerical errors during the iterations. This can be achieved by using hybrid computations instead of finite precision floating-point computations, which will help to avoid the propagation of errors during the iterations and to maintain at the same time the ability of the ERES to produce approximate solutions. The new approach for the hybrid implementation of the ERES algorithm will be discussed next and the algorithm in its new formulation will be referred to as the *Hybrid ERES algorithm*.

## 4.3 The implementation of the ERES method using hybrid computations

The construction of the algorithm for the ERES method is based on stable algebraic processes, which are applied iteratively on the initial basis matrix $P_{h+1}$. The main target of the ERES method is to reduce the number of the rows of $P_{h+1}$ and finally to end up with a unity rank matrix, which contains the coefficients of the GCD. The new implementation of the ERES algorithm involves the use of hybrid computations in order reduce the amount of the numerical errors arising from the different procedures of the ERES method and evidently improve the quality of the results.

The implementation of an algorithm in a hybrid computational environment depends mostly on the nature of the algorithm itself and the selection of the appropriate data structures to represent the input data. In a symbolic-numeric programming environment the type of data structures suggests the type of arithmetic operations. Arithmetic operations with integers or fractions of integers (rational operations) can be performed in infinite accuracy and this is an important feature to take into advantage.

A special characteristic of the ERES method is that it can be separated into two independent parts, the main procedure and the Rank-1 procedure, which can be implemented either using symbolic or numerical computations. But the question that arises here is which type of computations is best for each part of the ERES method. Since we want to avoid the accumulation of rounding errors during the iterations of the method and at the same time we are interested in the computation of approximate solutions, the answer to the previous question seems to be pretty direct. Due to the iterative nature of the ERES method the processes of Gaussian elimination and Shifting is preferable to be treated symbolically, because, although the data are constantly transformed, the introduction of rounding errors is avoided and the computation of the GCD remains unaffected. Conversely, the process of computing the singular values is better to be treated numerically. This allows us to control the magnitude of the singular values and hence the numerical rank of the processed matrix by setting an appropriate numerical tolerance. Therefore, a numerically efficient termination criterion can be created for detecting a matrix with rank approximately equal to 1 and the given solutions are considered as approximate GCDs.

The hybrid implementation of the ERES algorithm involves the use of exact rational (symbolic) operations for its main procedure and numerical (floating-point) operations for the Rank-1 procedure. The details of this combination and its effect on the ERES algorithm will be discussed in the following.

### 4.3.1 The formulation of the Hybrid ERES Algorithm

Having a set $\mathcal{P}_{h+1,n}$ and its basis matrix $P_{h+1} \in \mathbb{R}^{(h+1)\times(n+1)}$, a necessary preliminary step is to convert the given floating-point data to rational format (fractions of integers). After that, the next steps are implemented symbolically, using rational operations:

1. Reorder the rows of $P_{h+1}$ such that its first row corresponds to the polynomial of the lowest degree.
2. Scale the first row of $P_{h+1}$ to have the maximum pivot.
3. Apply Gaussian elimination with partial pivoting to $P_{h+1}$.
4. Apply Shifting to every row of $P_{h+1}$ .
5. Delete the zero rows and columns and form a new basis matrix $P_{h+1}^{(\cdot)}$ with reduced dimensions.

The steps 1–5 underlie the main procedure of the Hybrid ERES algorithm and they are implemented using symbolic-rational operations.

Denote by $P_{h+1}^{(\kappa)}$ the matrix which occurs after the $\kappa^{th}$ iteration of the main procedure of the algorithm ($\kappa = 1, 2, \ldots$). The following cases appear:

- The produced matrix $P_{h+1}^{(\kappa)}$ has one or more zero elements in its last column (i.e. the polynomials which correspond to the rows of the matrix have different degrees). Then, the steps 1–5 of the main procedure go over $P_{h+1}^{(\kappa)}$.
- The matrix $P_{h+1}^{(\kappa)}$ in the $\kappa^{th}$ iteration has no zero elements in its last column (i.e. the polynomials which correspond to the rows of the matrix have the same degree). Then, a numerical copy of $P_{h+1}^{(\kappa)}$ is made and the next steps are implemented numerically using finite precision floating-point operations:
  6. Normalisation of the rows of the matrix $P_{h+1}^{(\kappa)}$ using the Euclidean norm.
  7. Computation of the singular values of $P_{h+1}^{(\kappa)}$.

The steps 6 and 7 underlie the Rank-1 procedure of the Hybrid ERES algorithm and they are implemented using numerical floating-point operations.

If the matrix $P_{h+1}^{(\kappa)}$ has numerical $\varepsilon_t$-rank equal to 1 according to a small specified accuracy $\varepsilon_t > 0$, the algorithm stops and gives an appropriate solution. Otherwise, the algorithm starts a new iteration of the main procedure using the rational matrix $P_{h+1}^{(\kappa)}$. All the above steps are illustrated in Figure 4.1.

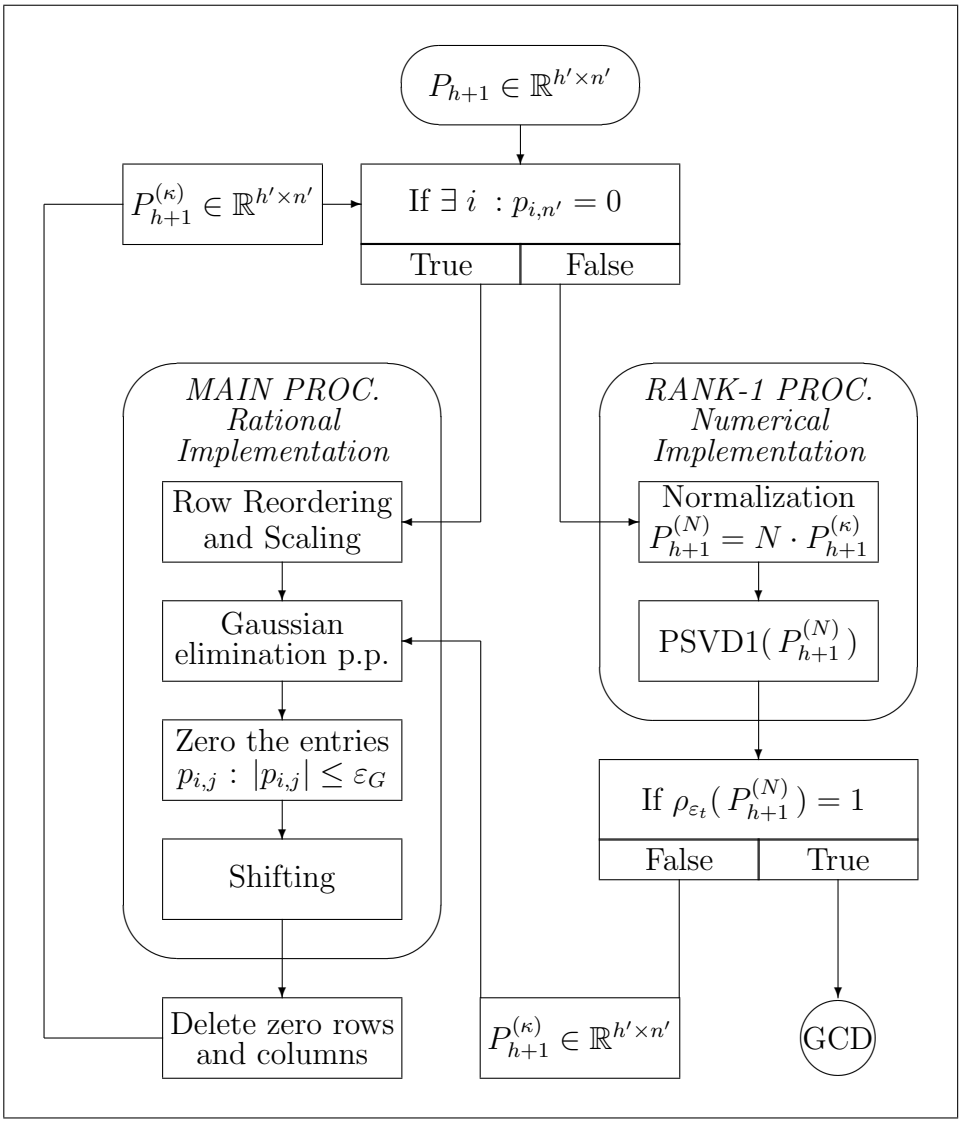


Figure 4.1: The Hybrid ERES Algorithm

### 4.3.2 Computation of the GCD with the Hybrid ERES algorithm

The computation of the GCD with the Hybrid ERES algorithm depends on the accuracy of the input data and the performed operations. If the coefficients of the polynomials of the given set are integer numbers (or fractions with integer numbers) and the GCD exists, then the successive symbolic application of the subprocedures of the main procedure to the basis matrix of the set will lead to a final matrix with rank equal to 1. Thus, any row of this matrix can give the coefficients of the GCD.

Otherwise, we must search for an approximate numerical solution, which is actually provided by the Rank-1 procedure of the algorithm. The numerical computation of the singular values of $P_{h+1}^{(\kappa)}$ is a typical process to estimate the rank of a matrix and provides the ERES algorithm with a termination criterion. This criterion is applied when the polynomials, which correspond to the rows of the matrix, have the same degree and it is described in Theorem 4.2. Every time the algorithm reaches this stage, there is a potential $\varepsilon_t$-rank 1 matrix for a specific accuracy $\varepsilon_t$. If we accept values of $\varepsilon_t \leq 10^{-1}$, we can obtain a series of matrices, that yield an $\varepsilon_t$-GCD. The produced approximate $\varepsilon_t$-GCD can be determined according to Proposition 4.1. Therefore, the polynomial that comes from the first row of the right singular matrix of $P_{h+1}^{(N)}$, can be considered as the numerical output of the Hybrid ERES algorithm.

### 4.3.3 The partial SVD method for approximate rank-1 matrices

Of course, the singular value decomposition (SVD) [26, 27] is undoubtedly a robust numerical procedure and, since we seek a unity rank matrix to terminate the Hybrid ERES algorithm, the only essential information we need is concerned with the first two singular values of the matrix $P_{h+1}^{(N)}$. Thus, it is not necessary to perform the whole singular value decomposition. The development of a partial singular value decomposition algorithm is presented in [75, 76]. The outline of a variation of the classical singular value decomposition method, especially developed for the efficient computation of the unique singular value and its right singular vector of an approximate $\varepsilon_t$-rank 1 matrix, is presented here and we shall refer to it as the *PSVD1 method*.

Consider a matrix $A$ with dimensions $m \times n$ and let $\sigma_1 > \sigma_2 > \ldots > \sigma_k$ be its singular values, $k = \min\{m, n\}$. For a specified numerical tolerance $\varepsilon_t << 1$, the matrix $A$ will be an approximate $\varepsilon_t$-rank 1 matrix if $\sigma_1 > \varepsilon_t \geq \sigma_2 > \ldots > \sigma_k$. The following algorithm establishes a efficient numerical procedure for the detection of a unity rank matrix.

**ALGORITHM 4.1. The PSVD1 Algorithm**

Input : Initial matrix $A \in \mathbb{R}^{m\times n}$, $k = \min\{m, n\}$,
a numerical tolerance (bound), $0 < \varepsilon_t < 1$

Step 1 : *Bidiagonalization phase.*

**If** $m \geq n$ **then**
transform $A$ into upper bidiagonal form $B_u$ by Householder transformations : $A = U_u \cdot B_u \cdot V_u^t$.
**else**
transform $A$ into lower bidiagonal form $B_l$ by Householder transformations : $A = U_l \cdot B_l \cdot V_l^t$.
**end if**

Step 2 : *Rank 1 detection phase.*

Given a bidiagonal matrix $B_u$ (or $B_l$), we only need to partition the bidiagonal into unreduced sub-bidiagonals so that only one singular value is greater than $\varepsilon_t$. In order to detect such a property, we construct a $2n \times 2n$ symmetric tridiagonal matrix $T$ with zero main diagonal from the elements of $B_u$ (or $B_l$) and compute the Sturm sequence for $T$ and $\varepsilon_t$ [26]. The positive symmetric eigenvalues of $T$ are the singular values of $B_u$ (or $B_l$) and the number of sign agreements in Sturm sequence correspond to the number of singular values, which are greater or equal to $\varepsilon_t$, [18].

Let $a = [a_i],\ i = 1, \ldots, k$ be the elements of the main diagonal of $B_u$ (or $B_l$), and $b = [b_i],\ i = 1, \ldots, k-1$ be the elements of the superdiagonal of $B_u$ (or the subdiagonal of $B_l$).
Construct $\underline{c} = [a_1, b_1, a_2, b_2, \ldots, b_{k-1}, a_k] \in \mathbb{R}^{2k-1}$.
Compute the Sturm sequence for $\theta := \varepsilon_t$
$p_0(\theta) = 1,\ p_1(\theta) = -\theta$
**For** $i = 2, 3, \ldots, 2k$ **do**
$p_i(\theta) = -\theta\, p_{i-1}(\theta) - c_{i-1}^2\, p_{i-2}(\theta)$
**end for**
Let $\eta$ be the number of sign agreements between the sequential terms $p_i(\theta)$ of the Sturm sequence.
Convention: If $p_i(\theta) = 0$ then $p_i(\theta)$ is in sign agreement with $p_{i-1}(\theta)$.

**If** $\eta \geq 2$ **then**

use a bisection method to compute a new bound $\varepsilon_t$.
(We use the bisection method to find an estimation of the second larger singular value $\sigma_2$ [18].)

**else**

Step 3 : *Back transformation phase.*

Find the largest (by absolute value) diagonal element $a_h$ for $h = 1, \ldots, k$ of $B_u$ (or $B_l$).
Construct an appropriate Givens Rotation matrix $G$ for the $h^{th}$ row of $B_u$ (or the $h^{th}$ column of $B_l$).
**If** $m < n$ **then**
$\sigma_1 :=$ the $h^{th}$ diagonal element of $S := G \cdot B_l$.
$w :=$ the $h^{th}$ row of the matrix $V_l^t$.
**else**
$\sigma_1 :=$ the $h^{th}$ diagonal element of $S' := B_u \cdot G^t$.
$w :=$ the $h^{th}$ row of the matrix $W := G \cdot V_u^t$.
**end if**
**end if**

Output : Singular value, singular vector, (new) numerical tolerance: ($\sigma_1$, $w$, $\varepsilon_t$)

---

The PSVD1 algorithm can detect a unity rank matrix very efficiently. The procedure that dominates the algorithm is the bidiagonalization of the initial matrix using Householder transformations. It is a numerically stable procedure which requires about $O(2mn^2 - \frac{2}{3}n^3)$ multiplications if $m < \frac{5}{3}n$, or $O(mn^2 + n^3)$ multiplications if $m \geq \frac{5}{3}n$ [76]. The technical advantage of this algorithm is that, when the initial matrix does not have an $\varepsilon_t$-rank equal to 1, it is not necessary to compute all the sequential terms of the Sturm sequence, because in this case we only need a couple of sign agreements to conclude that we do not have an $\varepsilon_t$-rank 1 matrix. This simple test helps to save more computational time and thus, we can have a fast and efficient way to detect an approximate $\varepsilon_t$-rank 1 matrix. In case we do have a unity rank matrix, the unique singular value and right singular vector can be computed explicitly without matrix products. Finally, the overall cost in computational operations is about the same magnitude of that of the bidiagonalization method. The bidiagonal reduction may also be applied to an upper triangular matrix $R$, obtained from $A$ by orthogonal transformations

such that $A = Q\,R$, [13]. This step improves the performance of the algorithm. A method for a more accurate bidiagonal reduction which combines Householder and Givens transformations is presented in [2].

The PSVD1 algorithm is a quick and effective tool for the detection of an approximate unity rank matrix. It can improve the performance of other methods, such as the ERES method, and it can be easily implemented in any software programming environment.

### 4.3.4 Behaviour of the Hybrid ERES Algorithm

The combination of symbolic-rational and numerical operations aims at the improvement of the stability of the ERES algorithm and the presence of "good" approximate solutions, the "good" in a sense to be precise later. The main procedure of the algorithm and especially the process of Gaussian elimination with partial pivoting, is entirely implemented by using symbolic-rational operations. With this technique any additional errors from the elementary row operations are avoided completely. The computations are always performed accurately and if the input data are exactly known and the polynomials of the given set are not coprime, then the GCD is given by any row of the final rank 1 matrix.

Obviously, symbolic-rational operations do not reveal the presence of approximate solutions. In cases of sets of polynomials with inexact coefficients, the presence of an approximate GCD relies on the proper determination of a numerical $\varepsilon_t$-rank 1 matrix for a specific small tolerance $\varepsilon_t$. The tolerance $\varepsilon_t$ is linked with the accuracy of the solution which is obtained from the Rank-1 procedure of the Hybrid ERES algorithm. An initial value of $\varepsilon_t$ can be set by the user as an input to the Hybrid ERES algorithm (usually $\varepsilon_t \approx \mathbf{u}$) and then, every time when the algorithm reaches the Rank-1 procedure, an $\varepsilon_t$-GCD can be obtained according to a new value of $\varepsilon_t$, which is actually determined from the PSVD1 procedure. Thus, at the end of the Hybrid ERES algorithm we can have a series of $\varepsilon_t$-GCDs for all the different values of $\varepsilon_t$ computed by the algorithm itself. Unlike the previous version of the numerical ERES algorithm [57] where the choice of the $\varepsilon_t$ was absolutely arbitrary, in the present hybrid algorithm of the ERES method, the numerical accuracy $\varepsilon_t$ is proposed by the algorithm and this helps us to develop a better strategy for the best selection of the GCD.

The numerical accuracy $\varepsilon_G$ controls the magnitude of the elements of the matrix that we obtain at the end of the main procedure of the Hybrid ERES algorithm in each iteration. The elements of the matrix that are less than $\varepsilon_G$ by absolute value, are set equal to zero in order to avoid possible underflow errors during the transition from the main procedure to the Rank-1 procedure of the algorithm. In most cases, $\varepsilon_G$ can be set equal to $2^{-1}\mathbf{u}$, where $\mathbf{u}$ is the machine's precision (hardware numerical accuracy). Otherwise, it can be set appropriately

in relation to a norm of the associated matrix (usually the infinity norm).

Therefore, the singular value decomposition together with the normalization process of the matrix $P_{h+1}^{(\kappa)}$ are performed by using floating-point operations. The polynomial that comes from Proposition 4.1 can be considered as a GCD approximation and represents the numerical output of the Hybrid ERES algorithm. The normalization of the rows of any matrix $P_{h+1}^{(\kappa)}$ (by the Euclidean norm) does not introduce significant errors and in fact the following result has been proven [57, 81]:

**Proposition 4.2.** *The normalization $P_{h+1}^{(N)}$ of a matrix $P_{h+1}^{(\kappa)} \in \mathbb{R}^{h' \times n'}$, computed by the ERES method in the $\kappa^{th}$ iteration, using floating-point arithmetic with unit round-off $u$, satisfies the properties :*

$$P_{h+1}^{(N)} = N \cdot P_{h+1}^{(\kappa)} + E_N, \quad \|E_N\|_\infty \leq 3.003 \cdot n' \cdot u \tag{4.9}$$

*where $N = diag(\nu_1, \nu_2, \ldots, \nu_{h'}) \in \mathbb{R}^{h' \times h'}$, $\nu_i = \left( \left\| P_{h+1}^{(\kappa)}[i, 1 \ldots n'] \right\|_2 \right)^{-1}$ for $i = 1, \ldots, h'$ is the matrix accounting for the performed transformations and $E_N \in \mathbb{R}^{h' \times n'}$ the error matrix.*

The PSVD1 method is applied to the matrix $P_{h+1}^{(N)}$. The preliminary stage in this algorithm is the bidiagonal reduction of $P_{h+1}^{(N)}$ and in most bidiagonal reduction methods the error is expressed in the following form [18, 26, 27]:

$$P_{h+1}^{(N)} + \delta P_{h+1}^{(N)} = U\, B\, V^t \,,$$

$$\|\delta P_{h+1}^{(N)}\|_2 \leq \mathbf{u}\, f(h', n')\, \|P_{h+1}^{(N)}\|_2 \tag{4.10}$$

where $B$ is bidiagonal, $U$ and $V$ are orthogonal, $\mathbf{u}$ is the machine precision and $f(h', n')$ is a modestly growing function of the dimensions of $P_{h+1}^{(N)}$ [18, 27], where $h' < h + 1$ and $n' < n + 1$.

It is important to notice that the Rank-1 procedure is actually applied to a numerical copy of the matrix $P_{h+1}^{(\kappa)}$ and thus the performed transformations during the Rank-1 procedure do not affect the matrix $P_{h+1}^{(\kappa)}$ when returning to the main procedure. For this reason, there is no accumulation of floating-point errors. The only numerical errors appearing are from the Rank-1 procedure and concern the normalization and the partial singular value decomposition of the last matrix $P_{h+1}^{(\kappa)}$. The total numerical error of the Hybrid ERES algorithm is actually represented by the relations (4.9) and (4.10).

The combination of symbolic-rational and numerical computations ensures the numerical stability of the Hybrid ERES algorithm and gives to the ERES the characteristics of a *hybrid computational method*.

**Computational complexity.** For a set of polynomials the number of multiplications or divisions which are performed in the $\kappa^{th}$ iteration of the algorithm, depends on the size of the matrix $P_{h+1}^{(\kappa)}$ and it is summarized in Table 4.1. The first iteration is the most computationally expensive iteration since the initial basis matrix is larger than any $P_{h+1}^{(\kappa)}$. Unless we know exactly the degree of the GCD of the set we cannot specify from the beginning the number of iterations required by the algorithm. Practically, the number of iterations is about $O(n)$. The computational cost of the PSVD1 method is dominated by the bidiagonalization of the input matrix.

| Gaussian elimination | Normalization | PSVD1 |
|---|---|---|
| $O(\frac{z^3}{3}),\ z = \min(h'-1, n')$ | $O(2h'n')$ | $O(2h'n'^2 - \frac{2}{3}n'^3)$ |

Table 4.1: Required operations for the matrix $P_{h+1}^{(\kappa)} \in \mathbb{R}^{h' \times n'}$ in the Hybrid ERES algorithm.

**Computational examples.** In the following examples we want to compute the GCD of a set of 5 real polynomials in one variable and we will demonstrate how the Hybrid ERES algorithm works. For the purposes of this study, the algorithm was implemented in the mathematical programming environment of Maple as described in the appendix A.1.2 .

**Example 4.1.** We consider the set $\mathcal{P}_{h+1,n} = \mathcal{P}_{5,7}$ of 5 polynomials with maximum degree 7 :

$$\begin{aligned}
p_1(s) &= s^7 + 711\,s^6 - 37830\,s^5 + 167014\,s^4 - 308538\,s^3 \\
&\quad +325453\,s^2 - 193993\,s + 47182 \\
p_2(s) &= -10\,s^7 - 7580\,s^6 + 22880\,s^5 + 37902\,s^4 - 112691\,s^3 \\
&\quad -47768\,s^2 + 133141\,s - 25874 \\
p_3(s) &= -3\,s^7 - 2200\,s^6 + 62935\,s^5 - 173295\,s^4 + 161898\,s^3 \\
&\quad -148265\,s^2 + 98930\,s \\
p_4(s) &= 98\,s^7 + 74294\,s^6 - 215976\,s^5 + 112702\,s^4 + 56365\,s^3 \\
&\quad +38550\,s^2 - 198447\,s + 132414 \\
p_5(s) &= 22\,s^7 + 16763\,s^6 + 15764\,s^5 - 165057\,s^4 + 61089\,s^3 \\
&\quad +159080\,s^2 + 23445\,s - 111106
\end{aligned}$$

The set is constructed such that the polynomial

$$g(s) = s^3 + 758\,s^2 - 2281\,s + 1522$$

is an exact GCD. The corresponding basis matrix $P_{h+1} = P_5$ has dimensions $5 \times 8$ and rank $\rho(P_5) = 5$.

$$P_5 = \begin{bmatrix} 47182 & -193993 & 325453 & -308538 & 167014 & -37830 & 711 & 1 \\ -25874 & 133141 & -47768 & -112691 & 37902 & 22880 & -7580 & -10 \\ 0 & 98930 & -148265 & 161898 & -173295 & 62935 & -2200 & -3 \\ 132414 & -198447 & 38550 & 56365 & 112702 & -215976 & 74294 & 98 \\ -111106 & 23445 & 159080 & 61089 & -165057 & 15764 & 16763 & 22 \end{bmatrix}$$

We set the arithmetic system's accuracy (software accuracy) to double precision (16 digits). The input data are the coefficients of the 5 polynomials given in the matrix form $P_5$. The termination and Gaussian accuracies are initially set to

$$\varepsilon_t = \varepsilon_G = 4.440892 \cdot 10^{-16}$$

which are approximately equal to Maple's 16-digits software accuracy,

$$eps = 2^{-51} = 4.440892098500626 \cdot 10^{-16}$$

Considering the initial type of data, we will examine two cases:

**Case 1:** The input data are given in their original data type as integers. The results given next, were obtained.

**Output 1:** $g(s) := HEresGCD(P5)$;

GCD degree = 3, tolS > 3.701184e-16, tolG < 1.000000e-02, Iterations = 3
Parameters = { tolS–> 4.440892e-16, tolG–> 4.440892e-16, digits–> 16 }
Statistics = { Iterations = 3, SVDcalls = 2, Order = 0, Time = 0.078 sec }
Distance from rational solution = 0.000000e+00
Minimum termination tolerance = 3.701184e-16
$g(s) = s^3 + 758s^2 - 2281s + 1522$

The above Maple output 1 shows that the Hybrid ERES algorithm has performed 3 iterations of its main procedure in order to compute the GCD of the set. Since the input data were given exactly as integer numbers with no additional numerical error, the algorithm is designed to extract the GCD from the rows of the last rank 1 matrix in its exact form. As we can see, the "Distance from rational solution" is zero. The accuracies $\varepsilon_t$ and $\varepsilon_G$ are represented by the parameters tolS and tolG, respectively. The termination criterion was checked twice, as the "SVDcalls" indicates. The minimum termination tolerance was set by the algorithm itself (from the PSVD1 procedure) equal to $3.701184 \cdot 10^{-16}$. This value is extremely close to the accuracy of the system *eps* and it means that the algorithm terminated correctly.

**Case 2:** The input data are given now in numerical data type as 16-digits floating-point numbers. The next Maple output 2 shows that the Hybrid ERES algorithm has performed 3 iterations of its main procedure in order to compute the GCD of the set. Since the input data were given as floating-point numbers, the algorithm is designed to extract the GCD from the right singular vector of the last $\varepsilon_t$-rank 1 matrix in its numerical form. However, the GCD can also be derived from the rows of the last matrix in its rational form. This is the "rational solution". The distance from rational solution is $1.166190 \cdot 10^{-12}$, which is actually the Euclidean distance between the two vector solutions. But the relative error between the obtained numerical GCD and the exact GCD is only $4.099091 \cdot 10^{-16}$, approximately equal to the specified system's accuracy *eps*, which means that it is an acceptable numerical solution. The termination criterion was checked twice, as the "SVDcalls" indicates. The minimum termination tolerance was set again equal to $3.701184 \cdot 10^{-16}$, close to the selected accuracy of the system *eps*, which shows that the algorithm terminated correctly as before.

**Output 2:** $g(s) := HEresGCD(evalf(P5))$;

GCD degree = 3, tolS > 3.701184e-16, tolG < 1.000000e-02, Iterations = 3
Parameters = { tolS–> 4.440892e-16, tolG–> 4.440892e-16, digits–> 16 }
Statistics = { Iterations = 3, SVDcalls = 2, Order = 0, Time = 0.063 sec }
Distance from rational solution = 1.166190e-12
Minimum termination tolerance = 3.701184e-16
$g(s) = 1.000000000000000s^3 + 757.9999999999994s^2 - 2280.999999999999s$
$+1522.000000000000$

□

We shall examine now the sensitivity of the Hybrid ERES algorithm to small perturbations in the data.

**Example 4.2.** In the previous set $\mathcal{P}_{5,7}$ we add some perturbation $\epsilon = 10^{-8}$, such that the first and the last polynomial of the set have exact GCD :

$$\begin{aligned} g_1(s) &= (s+761)(s-1+\epsilon)(s-2-\epsilon) = \\ &= s^3 + 758.00000000\,s^2 - 2281.000000010000\,s + 1521.999992390000 \end{aligned}$$

and the rest of them have exact GCD :

$$\begin{aligned} g_2(s) &= (s+761)(s-1-\epsilon)(s-2+\epsilon) = \\ &= s^3 + 758.00000000\,s^2 - 2280.999999990000\,s + 1522.000007610000 \end{aligned}$$

and hence, we construct a new polynomial set $\mathcal{P}'_{5,7}$ with a perturbed basis matrix

$P'_5$. The Frobenius matrix norm of this perturbation is

$$\frac{\|P_5 - P'_5\|_F}{\|P_5\|_F} = 2.692206146827510 \cdot 10^{-9} \tag{4.11}$$

Obviously, the exact GCD of the perturbed set $\mathcal{P}'_{5,7}$ is $g(s) = s+761$. However, the next Maple output 3 shows that the Hybrid ERES algorithm detected two different solutions for this set; a GCD of degree 3 and another one of degree 1, which correspond to different values of the termination accuracy $\varepsilon_t$. Of course, according to the initial value $\varepsilon_t = 4.440892 \cdot 10^{-16}$, the algorithm terminated after 6 iterations of its main procedure when it reached the numerical threshold of the system *eps* and gave the expected GCD of degree 1 with negligible numerical error.

**Output 3:** $g(s) := HEresGCD(evalf(P5))$;

GCD degree = 3, tolS > 7.496191e-07, tolG < 1.000000e-02, Iterations = 3
GCD degree = 1, tolS > 1.000000e-15, tolG < 1.000000e-01, Iterations = 5
GCD degree = 1, tolS > 0.000000e+00, tolG < 1.000000e+00, Iterations = 6
Parameters = { tolS–> 4.440892e-16, tolG–> 4.440892e-16, digits–> 16 }
Statistics = { Iterations = 6, SVDcalls = 4, Order = 0, Time = 0.140 sec }
Distance from rational solution = 2.000000e-13
Minimum termination tolerance = 0.000000e+00
$g(s) = 1.000000000000000s + 761.0000000000001$

The suggested termination accuracy for a $3^{rd}$ degree GCD is $7.496191 \cdot 10^{-07}$. We set $\varepsilon_t = 8 \cdot 10^{-7}$ and, as the next Maple output 4 shows, after 3 iterations of its main procedure the Hybrid ERES algorithm gives the polynomial :

$$g'(s) = s^3 + 757.9999996097447\, s^2 - 2281.000296203550\, s + 1522.000593800050$$

with relative distance from the polynomials $g_1(s)$ and $g_2(s)$ approximately equal to $2.3 \cdot 10^{-7}$.

$$\frac{\|g_1(s) - g'(s)\|_2}{\|g_1(s)\|_2} = 2.308547251688193 \cdot 10^{-7}$$

$$\frac{\|g_2(s) - g'(s)\|_2}{\|g_2(s)\|_2} = 2.356387220252584 \cdot 10^{-7}$$

If we change again the value of the termination accuracy to $\varepsilon_G = 2 \cdot 10^{-15}$, a new GCD will be obtained after 5 iterations of the main procedure, as it is shown in the Maple output 5. This is a GCD of degree 1 with relative distance from the expected exact GCD about $3.942178 \cdot 10^{-16}$. This value is extremely close to the accuracy of the system *eps* and thus the given GCD can be considered as a numerically adequate solution.

**Output 4:** $g(s) := HEresGCD(evalf(P5), tolS = 8e-7, stopit = 3);$

GCD degree = 3, tolS > 7.496191e-07, tolG < 1.000000e-02, Iterations = 3
Parameters = { tolS–> 8.000000e-07, tolG–> 4.440892e-16, digits–> 16 }
Statistics = { Iterations = 3, SVDcalls = 2, Order = 0, Time = 0.078 sec }
Distance from rational solution = 3.310262e-03
Minimum termination tolerance = 7.496191e-07
$g(s) = 1.000000000000000s^3 + 757.9999996097447s^2 - 2281.000296203550s$
$+1522.000593800050$

**Output 5:** $g(s) := HEresGCD(evalf(P5), tolS = 2e-15, stopit = 5);$

GCD degree = 3, tolS > 7.496191e-07, tolG < 1.000000e-02, Iterations = 3
GCD degree = 1, tolS > 1.000000e-15, tolG < 1.000000e-01, Iterations = 5
Parameters = { tolS–> 2.000000e-15, tolG–> 4.440892e-16, digits–> 16 }
Statistics = { Iterations = 5, SVDcalls = 3, Order = 0, Time = 0.125 sec }
Distance from rational solution = 4.000000e-13
Minimum termination tolerance = 1.000000e-15
$g(s) = 1.000000000000000s + 761.0000000000003$

In all the above Maple outputs, note that there are also suggestions for new values for the Gaussian accuracy $\varepsilon_G$ (parameter tolG). However, we do not pay any attention to them, because they are considerably high in terms of the initial perturbation of the data. □

## 4.4 The performance of the ERES method computing the GCD of polynomials

The ERES method is quite effective when properly implemented in a programming environment. We can have large sets of real polynomials without restrictions to the type of data. Actually the method proves to be faster, when the polynomials of a given large set $\mathcal{P}_{h+1,n}$ are linearly depended. An appropriate selection of a base of the original set $\mathcal{P}_{h+1,n}$, helps ERES to reduce dramatically the row dimension of the initial basis matrix $P_{h+1}$ and hence proceed with a smaller set of polynomials. This reduction always takes place in the first iteration of the method. In fact, for any polynomial set, considering the vector of coefficients for each polynomial, there is a way to find the most orthogonal linearly independent representatives of the set, without transforming the original data, and form a base of polynomials, which can give us the GCD of the whole set. Such a base is referred to as a *best uncorrupted base* [57, 61]. However, the process of computing such a base is very demanding in terms of time and memory and it can benefit the ERES method only if the basis matrix of $\mathcal{P}_{h+1,n}$ is highly rank deficient ($h \gg n$).

The ERES method can be implemented by using different types of arithmetic

systems. Such arithmetic systems are :

**SFP** Standard Floating-Point arithmetic, where the internal accuracy of the system is fixed and often limited to 16 digits (double precision[1]).

**VFP** Variable Floating-Point arithmetic, where the internal accuracy of the system is determined by the user (variable precision[2]).

**ES** Exact Symbolic arithmetic, where the system performs the arithmetical operations in arbitrary precision.

**HY** The combination of VFP and ES arithmetic will be referred to as Hybrid arithmetic.

**ERES in SFP arithmetic.** In computing, floating-point describes a system for representing numbers that would be too large or too small to be represented as integers. Numbers are in general represented approximately to a fixed number of significant digits and scaled using an exponent. The approximation of a number to a floating-point format is known as *rounding*. The standard floating-point representation of a number (double precision number) is considered at approximately 16 decimal digits[3] of precision. SFP operations are fast and reliable but various tests have showed that, when using SFP arithmetic, the standard 16-digits accuracy (hardware accuracy) is not always enough for the ERES method to produce good results.

**ERES in VFP arithmetic.** Variable precision operations, in VFP arithmetic, can always be our choice since they are faster and more economical in memory bytes than exact symbolic operations, especially if there is no need to use enough digits to have a reliable software accuracy. But, it has been observed that, when the basis matrix $P_{h+1}$ has large dimensions and full rank, we must assign a lot of digits to the software accuracy in order to avoid great numerical errors. This is absolutely necessary especially when the column dimension of $P_{h+1}$ is large and the degree of the GCD is small, because this will invoke ERES to perform many iterations and hence, increase the propagation of errors. Additionally, if we increase the number of digits of the software accuracy, the time and memory requirements will also increase. Various tests on several polynomial sets with floating-point data in VFP arithmetic showed that we can obtain quite satisfactory results from ERES, if we allow the system to perform the internal operations with more than 20 digits of accuracy and simultaneously assign large values to

[1] It is also referred to as hardware floating-point precision or hardware accuracy.

[2] It is also referred to as software floating-point precision or software accuracy.

[3] The exact number is $53 \cdot \log_{10} 2 \approx 15.955$. Thus, in many mathematical programs the default hardware accuracy is set to 15 decimal digits.

the accuracies $\varepsilon_t$ and $\varepsilon_G$, $(\varepsilon_t, \varepsilon_G \geq 10^{-8})$. But still, it is not easy for the user to determine in advance the proper number of digits that are necessary for a successful computation of the GCD.

**ERES in ES arithmetic.** Exact symbolic operations, in ES arithmetic, could have been our first and only choice, since they produce excellent results with minimal or no error at all. But, they are very costly regarding time and memory (Table 4.7). This problem is more obvious, when the basis matrix $P_{h+1}$ is large and dense. Indeed, the successive matrix transformations, performed by the ERES method, take enough time to complete and consume a lot of memory bytes. However, if we recall that ERES decreases the size of the basis matrix during the iterations, the symbolical manipulation of the data can rarely be prohibitively expensive. Exact symbolic operations are considered a good choice for the ERES algorithm, when the basis matrix $P_{h+1}$ is sparse or when we seek an exact solution.

**ERES in HY arithmetic.** Since both variable precision operations and symbolic operations have advantages and disadvantages, it would be best if we could combined them appropriately – hybrid computations (HC) – in order to achieve good performance and stability for the ERES algorithm in every case. Several mathematical software packages, such as Maple, Mathematica, or Matlab, allow us to choose freely whether to use VFP operations or ES operations, by converting the initial data to an appropriate type. Note that *the type of the initial data suggests the type of operations*. For example, if our initial data are type of rational or radical, then ES operations will be performed. However, there are some restrictions, which oblige us to convert our data to floating-point format in VFP or SFP arithmetic, especially when we are not certain about the existence of a nontrivial GCD. When we work with ERES using HY operations, we can select any row of the last matrix $P_{h+1}^{(\kappa)}$ as a GCD, but when the initial data are given inexactly, the solution must be sought by using the SVD termination criterion performing VFP operations.

**Computational results.** The above remarks about the numerical and symbolical behaviour of the ERES method are illustrated with the following examples.

**Example 4.3.** We consider first a set $\mathcal{P}_{h+1,n} = \mathcal{P}_{11,20}$ of 11 polynomials in one variable with integer coefficients (max. 3 digits), maximum degree 20, and exact GCD, $g(s) = s^3 + 3s^2 + 4s + 2$, [57, 59]. The polynomials of the set $\mathcal{P}_{11,20}$ are:

$$\begin{aligned} p_1(s) \quad = \quad & s^{20} + 4\,s^{19} + 7\,s^{18} + 21\,s^{17} + 54\,s^{16} + 82\,s^{15} + 61\,s^{14} + 29\,s^{13} \\ & +36\,s^{12} + 47\,s^{11} + 26\,s^{10} + 7\,s^9 + 15\,s^8 + 20\,s^7 + 12\,s^6 + 6\,s^5 \\ & +27\,s^4 + 131\,s^3 + 286\,s^2 + 318\,s + 140 \end{aligned}$$

$$
\begin{aligned}
p_2(s) &= s^{20} + 3\,s^{19} + 4\,s^{18} + 2\,s^{17} + 3\,s^{14} + 9\,s^{13} + 12\,s^{12} + 6\,s^{11} \\
&\quad +5\,s^{10} + 15\,s^9 + 22\,s^8 + 16\,s^7 + 9\,s^6 + 7\,s^5 + 4\,s^4 + 2\,s^3 \\
p_3(s) &= s^{20} + 3\,s^{19} + 4\,s^{18} + 2\,s^{17} + s^{13} + 3\,s^{12} + 4\,x^{11} + 2\,s^{10} + s^6 \\
&\quad +3\,s^5 + 15\,s^4 + 35\,s^3 + 44\,s^2 + 22\,s \\
p_4(s) &= 5\,s^{20} + 15\,s^{19} + 20\,s^{18} + 10\,s^{17} + 4\,s^{13} + 12\,s^{12} + 16\,s^{11} + 8\,s^{10} \\
&\quad +2\,s^8 + 6\,s^7 + 8\,s^6 + 4\,s^5 + 10\,s^3 + 30\,s^2 + 40\,s + 20 \\
p_5(s) &= -s^{20} - 3\,s^{19} - 4\,s^{18} - 2\,s^{17} - s^8 - 3\,s^7 - 4\,s^6 - 2\,s^5 + 30\,s^3 \\
&\quad +90\,s^2 + 120\,s + 60 \\
p_6(s) &= s^{20} + 3\,s^{19} + 4\,s^{18} + 2\,s^{17} - 2\,s^{16} - 6\,s^{15} - 8\,s^{14} - 4\,s^{13} + s^{12} + 3\,s^{11} \\
&\quad +4\,s^{10} - s^9 - 9\,s^8 - 12\,s^7 - 6\,s^6 + 11\,s^3 + 33\,s^2 + 44\,s + 22 \\
p_7(s) &= s^{20} + 3\,s^{19} + 4\,s^{18} + 2\,s^{17} + 11\,s^{10} + 33\,s^9 + 44\,s^8 + 22\,s^7 + 20\,s^3 \\
&\quad +60\,s^2 + 80\,s + 40 \\
p_8(s) &= s^{20} + 3\,s^{19} + 7\,s^{18} + 11\,s^{17} + 12\,s^{16} + 8\,s^{15} + 6\,s^{14} + 8\,s^{13} + 4\,s^{12} \\
&\quad +5\,s^9 + 15\,s^8 + 20\,s^7 + 10\,s^6 + 9\,s^3 + 27\,s^2 + 36\,s + 18 \\
p_9(s) &= s^{20} + 3\,s^{19} + 4\,s^{18} + 3\,s^{17} + 3\,s^{16} + 4\,s^{15} + 5\,s^{14} + 9\,s^{13} + 13\,s^{12} \\
&\quad +9\,s^{11} + 9\,s^{10} + 17\,s^9 + 20\,s^8 + 10\,s^7 + s^6 + 3\,s^5 + 4\,s^4 + 5\,s^3 \\
&\quad +9\,s^2 + 12\,s + 6 \\
p_{10}(s) &= s^{20} + 2\,s^{19} + s^{18} - 2\,s^{17} - 2\,s^{16} + s^{12} + 3\,s^{11} + 4\,s^{10} + 2\,s^9 \\
&\quad -s^8 - 3\,s^7 - 4\,s^6 - 2\,s^5 - 4\,s^3 - 12\,s^2 - 16\,s - 8 \\
p_{11}(s) &= s^{20} + 3\,s^{19} + 15\,s^{18} + 35\,s^{17} + 44\,s^{16} + 22\,s^{15} + 3\,s^{14} + 9\,s^{13} \\
&\quad +13\,s^{12} + 9\,s^{11} + 4\,s^{10} + 2\,s^9 + 30\,s^3 + 90\,s^2 + 120\,s + 60
\end{aligned}
$$

The associated basis matrix $P_{11}$ has dimensions $11 \times 21$ and rank $\rho(P_{11}) = 11$. For this polynomial set, we studied the performance of the ERES method using SFP, VFP and HY types of arithmetic. The results are presented in Table 4.4.

| | |
|---|---|
| Type of data | float |
| Accuracies | SFP, Digits = 15, $\varepsilon_t = 10^{-15}$, $\varepsilon_G = 10^{-15}$ |
| GCD | 1.0 |
| Iterations | 10 |
| Type of data | float |
| Accuracies | SFP, Digits = 15, $\varepsilon_t = 10^{-15}$, $\varepsilon_G = 10^{-8}$ |
| GCD | $1.0\,s^3 + 2.99999999608220\,s^2 +$ $3.99999999215816\,s + 1.99999999215450$ |
| Relative error | $0.215\ 10^{-8}$ |
| Iterations | 9 |

| | |
|---|---|
| Type of data | float |
| Accuracies | SFP, Digits = 15, $\varepsilon_t = 10^{-8}$, $\varepsilon_G = 10^{-8}$ |
| GCD | $1.0\,s^3 + 2.99999996640441\,s^2 +$ |
| | $3.99999993280994\,s + 1.99999993280184$ |
| Relative error | $0.184\ 10^{-7}$ |
| Iterations | 9 |
| Type of data | float |
| Accuracies | VFP, Digits = 18, $\varepsilon_t = 10^{-12}$, $\varepsilon_G = 10^{-8}$ |
| GCD | $1.0\,s + 1.00000204413857072$ |
| Relative error | 0.816 |
| Iterations | 9 |
| Type of data | float |
| Accuracies | VFP, Digits = 20, $\varepsilon_t = 10^{-16}$, $\varepsilon_G = 10^{-12}$ |
| GCD | $1.0\,s^3 + 3.0000000000001436500\,s^2 +$ |
| | $4.0000000000002876900\,s + 2.0000000000002882608$ |
| Relative error | $0.788\ 10^{-13}$ |
| Iterations | 8 |
| Type of data | float |
| Accuracies | VFP, Digits = 23, $\varepsilon_t = 10^{-16}$, $\varepsilon_G = 10^{-16}$ |
| GCD | 1.0 |
| Iterations | 10 |
| Type of data | float |
| Accuracies | VFP, Digits = 23, $\varepsilon_t = 10^{-16}$, $\varepsilon_G = 10^{-12}$ |
| GCD | $1.0\,s^3 + 2.999999999999997573495\,s^2 +$ |
| | $3.999999999999995261066\,s + 1.999999999999995238086$ |
| Relative error | $0.130\ 10^{-15}$ |
| Iterations | 7 |
| Type of data | float |
| Accuracies | VFP, Digits = 24, $\varepsilon_t = 10^{-16}$, $\varepsilon_G = 10^{-16}$ |
| GCD | $1.0\,s^3 + 2.99999999999999999998275\,s^2 +$ |
| | $3.99999999999999999996562\,s + 1.99999999999999999996565$ |
| Relative error | $0.942\ 10^{-20}$ |
| Iterations | 8 |
| Type of data | rational |
| Accuracies | HY, Digits = 16, $\varepsilon_t = 10^{-16}$, $\varepsilon_G = 10^{-16}$ |
| GCD | $s^3 + 3\,s^2 + 4\,s + 2$ |
| Relative error | 0 |
| Iterations | 5 |

Table 4.4: Numerical behaviour of the ERES algorithm for the set $\mathcal{P}_{11,20}$ in Example 4.3.

The results in Table 4.4 confirm that:

a) SFP operations produce less accurate results than VFP operations.

b) The accuracy of the computational system represented by the term "Digits" and the Gaussian accuracy $\varepsilon_G$ both influence the produced GCD.

c) Since the coefficients of the polynomials are given as integer numbers, HY operations help ERES to produce the desired exact GCD in less iterations and of course without rounding errors.

□

*REMARK* 4.1. The relative errors were estimated using the Euclidean norm in the form:

$$Rel = \frac{\|\underline{g} - \underline{g}'\|_2}{\|\underline{g}\|_2} \tag{4.12}$$

where $\underline{g}$ is the coefficient vector of the exact GCD and $\underline{g}'$ is the coefficient vector of the produced GCD.

**Example 4.4.** Consider now a set $\mathcal{P}_{h+1,n} = \mathcal{P}_{10,9}$ of 10 polynomials in one variable with coefficients of various type (integers, rational numbers, decimal numbers), maximum degree 9, and exact GCD, $g(s) = s^2 + 0.125\,s + 0625$. The polynomials of the set $\mathcal{P}_{10,9}$ are:

$$\begin{aligned}
p_1(s) &= 13\,s^{11} - \tfrac{13}{8}\,s^{10} + 8.125\,s^9 + 11\,s^6 - \tfrac{11}{8}\,s^5 - 6.125\,s^4 + \tfrac{45}{8}\,s^3 \\
&\quad -5.6250\,s^2 + 2.1250\,s + 1.875 \\
p_2(s) &= -24\,s^{11} + 3\,s^{10} - 18.0\,s^9 + \tfrac{83}{8}\,s^8 - 0.1250\,s^7 + 5.8750\,s^6 \\
&\quad -4.125\,s^5 + \tfrac{3}{4}\,s^4 - 9.750\,s^3 + \tfrac{3}{4}\,s^2 - 3.750\,s \\
p_3(s) &= 2\,s^{10} + \tfrac{19}{4}\,s^9 + 0.62500\,s^8 + 7.125\,x^7 - \tfrac{1}{2}\,s^6 - 12.500\,s^5 \\
&\quad -\tfrac{1}{8}\,s^4 - 7.1250\,s^3 - 1.5000\,s^2 + 1.250\,s \\
p_4(s) &= 16\,s^{11} - 2\,s^{10} + 10.0\,s^9 - 3\,s^8 + \tfrac{3}{8}\,s^7 - 6.875\,s^6 - \tfrac{35}{8}\,s^5 \\
&\quad -0.5000\,s^4 - 3.3750\,s^3 - 3.750\,s^2 + \tfrac{5}{8}\,s - 3.125 \\
p_5(s) &= s^{10} + \tfrac{63}{8}\,s^9 - 0.375\,s^8 - 1.0\,s^7 - \tfrac{29}{4}\,s^6 - 3.750\,s^5 + 2.1250\,s^4 \\
&\quad +4.5000\,s^3 + 3.6250\,s^2 + 3.750\,s \\
p_6(s) &= -s^{11} - \tfrac{7}{8}\,s^{10} - 2.5000\,s^9 - 5.3750\,s^8 - 0.62500\,s^7 + 4.875\,s^6 \\
&\quad +8\,s^5 + 3.8750\,s^4 + 1.625\,s^3 + \tfrac{1}{2}\,s^2 - 2.500\,s \\
p_7(s) &= -5\,s^{11} - \tfrac{27}{8}\,s^{10} + 1.3750\,s^9 + 9.0\,s^8 + 9.0\,s^7 - 0.500\,s^6 \\
&\quad +5.8750\,s^5 - 4.375\,s^4 - 9\,s^3 + \tfrac{9}{8}\,s^2 - 5.625\,s
\end{aligned}$$

$$
\begin{aligned}
p_8(s) &= -2\,s^{11} - \tfrac{31}{4}\,s^{10} - 4.250\,s^9 - 10.500\,x^8 + 2.2500\,s^7 - 4.2500\,s^6 \\
&\quad +5.500\,s^5 - \tfrac{43}{8}\,s^4 + 6.5000\,s^3 - 3.6250\,s^2 + 2.500\,s \\
p_9(s) &= 5\,s^{11} - \tfrac{13}{8}\,s^{10} - 1.7500\,s^9 + 8.0\,s^8 - 4.125\,s^7 + 5.0\,s^6 - 9\,s^5 \\
&\quad +\tfrac{9}{8}\,s^4 - 5.625\,s^3 - 4\,s^2 + \tfrac{1}{2}\,s - 2.500 \\
p_{10}(s) &= 2\,s^{11} - \tfrac{9}{4}\,s^{10} - 10.500\,s^9 + 9.250\,s^8 - 8.6250\,s^7 + 5.625\,s^6 \\
&\quad -7\,s^4 + \tfrac{7}{8}\,s^3 - 16.375\,s^2 + \tfrac{3}{2}\,s - 7.500
\end{aligned}
$$

The associated basis matrix $P_{10}$ has dimensions $10 \times 10$ and rank $\rho(P_{10}) = 10$. For this polynomial set, we studied the performance of the ERES method using SFP, VFP and HY types of arithmetic. The results are presented in Table 4.6.

| | |
|---|---|
| Type of data | float |
| Accuracies | SFP, Digits=15, $\epsilon_t = 10^{-15}$, $\epsilon_G = 10^{-15}$ |
| GCD | $1.0\,s^2 - 0.125000000000210\,s + 0.625000000001880$ |
| Relative error | $0.15952151\ 10^{-11}$ |
| Iterations | 6 |
| Type of data | float |
| Accuracies | VFP, Digits=21, $\epsilon_t = 10^{-15}$, $\epsilon_G = 10^{-15}$ |
| GCD | $1.0\,s^2 - 0.12500000000000004021\,s + 0.62499999999999997365$ |
| Relative error | $0.405400573\ 10^{-16}$ |
| Iterations | 6 |
| Type of data | float - rational |
| Accuracies | HY, Digits=21, $\epsilon_t = 10^{-15}$, $\epsilon_G = 10^{-15}$ |
| GCD | $1.0\,s^2 - 0.12500000000000000863\,s + 0.62500000000000018372$ |
| Relative error | $0.155097137\ 10^{-17}$ |
| Iterations | 3 |

Table 4.6: Numerical behaviour of the ERES algorithm for the set $\mathcal{P}_{10,9}$ in Example 4.4.

The results in Table 4.6 confirm that:

a) SFP operations produce less accurate results than VFP and HY operations.

b) The performed HY operations by the Hybrid algorithm of the ERES method, produced a more accurate result in less iterations than the numerical ERES algorithm using VFP operations. This was due to the triangularisation of the matrices without rounding errors.

□

Random polynomial sets have been selected for further testing of the behaviour of the ERES algorithm. The produced results are presented in Table 4.8. Regarding these tests, we have to note that all the polynomial sets have been constructed by random polynomials with integer coefficients ($\leq 9999$) so as their GCD is known. Furthermore, in order to focus on the different kinds of implementation, both accuracies $\varepsilon_t$ and $\varepsilon_G$ are set equal to $2.2\ 10^{-15}$, which is close enough to the limit of hardware accuracy. Thus, the accuracy of the results is based on the software accuracy, which in modern mathematical software programs can be determined by the user. When using ES operations, the accuracy of the system is set equal to the hardware precision (16 decimal digits) and the relative error is 0.

Using the experimental results in Table 4.8, we observe that:

a) The ERES algorithm is quite fast when using VFP or HY operations and becomes slow when ES operations are used.

b) The ERES algorithm performs many iterations of its main procedure when the maximum polynomial degree is much higher than the number of polynomials in the set.

c) When using HY operations the relative error is sufficiently small and more closer to the specified software accuracy.

d) The HY operations require less digits of software accuracy comparing to VFP operations in order to give a result of the same quality.

In Table 4.7 the results from a set $\mathcal{P}_{12,12}$ of 12 polynomials with randomly selected coefficients, maximum degree 11, and a 2-degree exact GCD, confirm that the ES operations require more computational time and consume at least 3 times more memory bytes than VFP and HY operations. However, in the case of VFP or HY operations, it was necessary to double the software accuracy (from 15 digits to 30 digits) in order to get a GCD with minimal as possible relative error. This indicates that SFP operations are not suitable here. Generally, the ERES algorithm gives more accurate and reliable results if we assign more than 20 digits to the system's software accuracy. In most cases of polynomials sets, 34 digits of precision (quadruple precision) are adequate for the ERES algorithm to produce very good results.

**Conclusions.** When working with the ERES method, the important factors that must to be taken into account are:

- The data type of the coefficients of the input polynomials.
- The dimensions and the structure of the original basis matrix $P_{h+1}$.

- The numerical accuracy of the original data (for floating-point numbers).
- Careful selection of the the tolerances $\varepsilon_t$ and $\varepsilon_G$ and the number of digits for the software accuracy.

The Hybrid ERES algorithm has the following advantages:

- It can handle large sets of polynomials more efficiently by using hybrid computations instead of numerical computations.
- It produces results with minimal rounding error in acceptable time limits and storage requirements.
- The produced results are less affected (or not affected) by the Gaussian accuracy $\varepsilon_G$ and thus its presence in the algorithm is not of great importance.
- The PSVD1 method for detecting a rank 1 matrix allows the selection of a proper value for the termination accuracy $\varepsilon_t$ which is based on the properties of the singular values of the processed matrix rather than in our intuition.
- It allows the computation of "meaningful" approximate solutions (which will be analysed later).

Therefore, we conclude that the computation of the GCD of a set of polynomials with real coefficients by the ERES method, is a process where numerical floating-point operations and exact symbolic operations must combined together for a better overall performance.

## 4.5 Discussion

In this chapter the hybrid implementation of the ERES method for computing approximate GCDs of a set of several real univariate polynomials has been developed. This implementation is based on the arithmetic properties of symbolic-numeric (hybrid) computations which enabled the formulation of a more efficient and numerically stable algorithm for the ERES method. The developed Hybrid ERES algorithm involves algebraic procedures which are implemented either symbolically or numerically to improve the overall performance of the ERES method and the quality of the given GCD. This algorithm was tested using different arithmetic systems and was compared with other algorithms which are also designed to compute the GCD of sets of polynomials by processing all the polynomials simultaneously. The obtained results show that the Hybrid ERES algorithm is faster and provides more accurate solutions compared with the other algorithms. The requirement for a quality solution has also led to the development of the PSVD1 algorithm for the efficient detection of an approximate rank-1 matrix.

The PSVD1 method is based on the partial singular value decomposition method and improves significantly the termination criterion of the ERES algorithm and the estimation of approximate GCDs. The ERES method and its hybrid form always produces estimates of the GCD as the result of the rank 1 approximation. Estimating how good such approximations are, is an issue related to the notion of the approximate GCD and the evaluation of its quality, which will be considered in the following.

| Type of data | Arithmetic | Digits | Memory (bytes) | Time (secs) |
|---|---|---|---|---|
| rational | ES | 15 | 7,735,542 | 0.547 |
| float | VFP | 30 | 2,868,462 | 0.172 |
| rational, float | HY | 30 | 2,853,434 | 0.203 |

Table 4.7: Storage and time requirements of the ERES algorithm for a random set of polynomials $\mathcal{P}_{12,12}$.

| | | | | | VFP | | | HY | | | ES |
|---|---|---|---|---|---|---|---|---|---|---|---|
| No. | $m$ | $d$ | $d_0$ | Iter | Dig | Rel | Time | Dig | Rel | Time | Time |
| i | 5 | 4 | 1 | 3 | 17 | $10^{-17}$ | 0.015 | 16 | $10^{-15}$ | 0.016 | 0.125 |
| ii | 8 | 7 | 2 | 5 | 21 | $10^{-18}$ | 0.032 | 17 | $10^{-16}$ | 0.032 | 0.110 |
| iii | 10 | 10 | 2 | 5 | 21 | $10^{-17}$ | 0.047 | 19 | $10^{-17}$ | 0.063 | 0.219 |
| iv | 5 | 10 | 2 | 6 | 20 | $10^{-18}$ | 0.078 | 17 | $10^{-14}$ | 0.109 | 0.250 |
| v | 15 | 15 | 2 | 7 | 22 | $10^{-15}$ | 0.172 | 20 | $10^{-15}$ | 0.187 | 0.656 |
| vi | 15 | 15 | 5 | 6 | 35 | $10^{-17}$ | 0.172 | 30 | $10^{-16}$ | 0.171 | 0.937 |
| vii | 20 | 15 | 5 | 5 | 21 | $10^{-18}$ | 0.203 | 18 | $10^{-16}$ | 0.219 | 1.156 |
| viii | 30 | 15 | 3 | 6 | 22 | $10^{-16}$ | 0.218 | 18 | $10^{-15}$ | 0.265 | 1.329 |
| ix | 2 | 20 | 4 | 33 | 32 | $10^{-15}$ | 0.266 | 32 | $10^{-15}$ | 0.234 | 15.031 |
| x | 15 | 20 | 4 | 7 | 23 | $10^{-17}$ | 0.234 | 22 | $10^{-17}$ | 0.265 | 38.391 |
| xi | 100 | 20 | 4 | 7 | 40 | $10^{-20}$ | 1.109 | 34 | $10^{-22}$ | 1.922 | 34.562 |

$m$ : the number of polynomials,
$d$ : the maximum degree of the polynomials,
$d_0$ : the degree of the GCD,
Iter : the number of iterations of the algorithm,
Dig : the number of Digits (software accuracy),
Rel : the relative error according to Frobenius norm,
Time : algorithm's estimated time of execution (sec).

Table 4.8: Behaviour of the ERES algorithm for random sets of polynomials.

# Chapter 5

# The strength of the approximate GCD and its computation

## 5.1 Introduction

The computation of the GCD of a set of polynomials is a problem representative of the class of nongeneric computations. The set of polynomials for which there exists a nontrivial GCD, different than 1, is a subvariety of the projective space with measure zero and this makes the computation of the GCD a hard problem. The subject of defining an *approximate GCD* goes back to the attempt of defining the notion of *almost zero* for a set of polynomials [43], where it has been shown that almost zeros behave in a similar way to exact zeros, as far as solutions of polynomial Diophantine equations. The issue of computing an approximate GCD has been considered before in [16, 20, 45, 52, 53, 57, 61, 67, 69, 70, 83, 85, 86] and references therein, but until recently the evaluation of the quality of the approximations from GCD computations was obscure and required special attention. The results presented in [21, 22] provided the fundamentals of a framework for the characterization of the *almost GCD* of a polynomial set and its *strength* which actually qualifies it. There, the notion of approximate GCD is defined as a distance problem between the given polynomial set and the given $d$ degree GCD variety. This approach is based on the representation of the greatest common divisor of many polynomials in terms of the factorisation of the generalised resultant into a reduced resultant and a Toeplitz matrix representing the GCD [21]. These results allow the parametrisation of all perturbations which are required to make a selected approximate GCD, an exact GCD of a perturbed set of polynomials.

However, the essence of the computation of approximate solutions is that they are based on the relaxation of exact conditions which characterise the greatest common divisor. Methods computing the GCD of a set of polynomials which deploy relaxation of the exact conditions for GCD evaluation, such as the ERES

method, lead to expressions for the approximate GCD. The quality, or strength of a given approximate GCD is then defined by the size of the minimal perturbation required to make the chosen approximate GCD, an exact GCD of the perturbed set. The solution of an optimisation problem then allows the evaluation of the quality of the given polynomial as an approximate solution [21, 42].

In this chapter we will be concerned with:

a) the computation of approximate GCDs of sets of several real polynomials in one variable using the developed ERES methodology and particularly the Hybrid ERES algorithm, and

b) the evaluation of the derived approximations using the developed methodology in [21, 22, 42] for computing the strength of a given approximate GCD.

In the following, a review of the representation of the GCD in terms of the resultant factorisation is given [22]. The established matrix-based representation of the GCD is equivalent to the standard algebraic factorisation of the GCD in the original set of polynomials and provides the means to define the notion of the approximate GCD in a formal way and then develop a computational procedure that allows the evaluation of its strength. This review continues with the definition and the computation of the distance of a given set of polynomials from a $d$-GCD variety which ultimately leads to an appropriate optimisation problem. The solution of this problem is the key for the evaluation of a given approximate GCD. However, in general this form of optimisation (minimisation) problem is actually non-convex in several cases of polynomial sets and a global solution is not always guaranteed [42].

The main objective in this chapter is to constrain this minimisation problem by finding some tight bounds which can be computed more easily than the actual problem and may work as indicators of the strength a given approximation. These indicators, defined as the *strength bounds*, can be found by exploiting the properties of the resultant GCD factorisation. The distance of these bounds is used to define the *average strength* of an approximate GCD. A simple algorithm is provided for the computation of the strength bounds and its computational complexity is analysed. Finally, the ability of the Hybrid ERES algorithm to produce approximate solutions and the evaluation of the quality of these approximations is discussed through various examples.

## 5.2 Representation of the GCD of polynomials

Consider the set of univariate polynomials

$$\mathcal{P}_{h+1,n} = \Big\{ \; a(s), b_i(s) \in \mathbb{R}[s],\ i = 1, 2, \ldots, h \text{ with } n = \deg\{a(s)\},\ p = \max_i\big(\deg\{b_i(s)\}\big) \leq n \text{ and } h, n \geq 1 \Big\} \quad (5.1)$$

We represent the polynomials $a(s),\ b_i(s)$ with respect to the highest degrees $(n, p)$ as

$$\begin{aligned} a(s) &= a_n s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0\ ,\ a_n \neq 0 \\ b_i(s) &= b_{i,p} s^p + \ldots + b_{i,1} s + b_{i,0}\ ,\ i = 1, 2, \ldots, h \end{aligned} \quad (5.2)$$

The set $\mathcal{P}_{h+1,n}$ is an $(n, p)$-*ordered polynomial set.*

*NOTATION* 5.1. Denote by $\Pi(n, p; h+1)$ the family of polynomial sets $\mathcal{P}_{h+1,n}$ having $h+1$ elements and highest degrees $(n, p),\ n \geq p$ ; i.e. if the degrees of the polynomials in the set are denoted by $d_i,\ i = 0, \ldots, h$, then $d_0 \geq d_1 \geq d_2 \geq \ldots \geq d_h$ and $d_0 = n,\ d_1 = p$.

The representation of the GCD relies on the square nonsingular Toeplitz matrices [21]. The following result provides a representation in matrix terms of the standard factorization of the GCD of a set of polynomials.

**Definition 5.1.** Let

$$v(s) = \lambda_r s^r + \cdots + \lambda_1 s + \lambda_0 \in \mathbb{R}[s] \text{ where } r \in \mathbb{Z}_+^*,\ \lambda_r, \lambda_0 \neq 0 \quad (5.3)$$

be a polynomial. A special Toeplitz matrix representation $\hat{\Phi}_v \in \mathbb{R}^{(n+p)\times(n+p)}$ of $v(s)$ can be defined by

$$\hat{\Phi}_v = \begin{bmatrix} \lambda_0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ \lambda_1 & \lambda_0 & \ddots & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & \vdots \\ \lambda_r & \vdots & \ddots & \ddots & \ddots & & \vdots \\ 0 & \lambda_r & & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & & \ddots & \lambda_0 & 0 \\ 0 & \cdots & 0 & \lambda_r & \cdots & \lambda_1 & \lambda_0 \end{bmatrix} \quad (5.4)$$

The matrix $\hat{\Phi}_v$ is lower triangular and unless its main diagonal is zero, it is always invertible. Its inverse $\hat{\Phi}_v^{-1} \in \mathbb{R}^{(n+p)\times(n+p)}$ is also lower triangular and can be found easily by computing its first column only [21].

**Definition 5.2.** We consider the set $\mathcal{P} = \mathcal{P}_{h+1,n}$ as defined in (5.1).

(i) Define a $p \times (n+p)$ matrix associated with $a(s)$ :

$$S_0 = \begin{bmatrix} a_n & a_{n-1} & a_{n-2} & \cdots & a_1 & a_0 & 0 & \cdots & 0 \\ 0 & a_n & a_{n-1} & \cdots & \cdots & a_1 & a_0 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_n & a_{n-1} & \cdots & \cdots & a_1 & a_0 \end{bmatrix}$$

and $n \times (n+p)$ matrices associated with each $b_i(s)$, $i = 1, 2, \ldots, h$ :

$$S_i = \begin{bmatrix} b_{i,p} & b_{i,p-1} & b_{i,p-2} & \cdots & b_{i,1} & b_{i,0} & 0 & \cdots & 0 \\ 0 & b_{i,p} & b_{i,p-1} & \cdots & \cdots & b_{i,1} & b_{i,0} & \ddots & \vdots \\ \vdots & & \ddots & \ddots & & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & b_{i,p} & b_{i,p-1} & \cdots & \cdots & b_{i,1} & b_{i,0} \end{bmatrix}$$

An *extended Sylvester matrix* or *generalized resultant* for the set $\mathcal{P}$ is defined by:

$$S_{\mathcal{P}} = \begin{bmatrix} S_0 \\ S_1 \\ \vdots \\ S_h \end{bmatrix} \in \mathbb{R}^{(p+hn)\times(n+p)} \tag{5.5}$$

(ii) The matrix $S_{\mathcal{P}}$ is the basis matrix of the set of polynomials :

$$S[\mathcal{P}] = \left\{ a(s), s\,a(s), \ldots, s^{p-1}\,a(s)\,;\; b_j(s), s\,b_j(s), \ldots, s^{n-1}\,b_j(s),\; j = 1, ..., h \right\}$$

which is also referred to as the *Sylvester resultant set* of the given set $\mathcal{P}$ [23, 82].

We can relate an $(n, p)$ extended Sylvester matrix to any polynomial set $\mathcal{P}_{h+1,n'}$ with two maximal degrees $n' = n - j$ and $p' = p - j$, $j > 0$ by assuming the first $j$ coefficients of the polynomials of $\mathcal{P}_{h+1,n'}$ to be zero. The new matrix will be called $(n, p)$-*expanded generalized resultant* of the set $\mathcal{P}_{h+1,n'}$.

*NOTATION* 5.2. The set of all generalized resultants corresponding to $h+1$ polynomials with maximal nominal degrees $(n, p)$ will be denoted by $\Psi(n, p; h+1)$.

*NOTATION* 5.3. In the following, we will denote by $m$ the row dimension of the above extended Sylvester matrix $S_{\mathcal{P}}$, where $m = p + hn$.

Toeplitz matrices and their properties are crucial elements in the representation of the GCD, which is defined by the following factorisation of resultants result [21, 22].

**Theorem 5.1.** *Let $\mathcal{P} \in \Pi(n, p; h+1)$ be a proper polynomial set (5.1). Let $S_{\mathcal{P}}$ be the respective extended Sylvester matrix (5.5), and $\phi(s)$ be the GCD of the set with degree $0 < d \leq p$. Then, there exists a transformation matrix $\hat{\Phi}_{\phi}$ (5.4), such that*

$$S_{\mathcal{P}} = \left[\mathbb{O}_{m,d} | \tilde{S}^{(d)}_{\mathcal{P}^*}\right] \cdot \hat{\Phi}_{\phi} \tag{5.6}$$

*where $\mathbb{O}_{m,d}$ is the $m \times d$ zero matrix, $m = p + hn$, $\mathcal{P}^* \in \Pi(n-d, p-d; h+1)$ is the set of coprime polynomials obtained from the original set $\mathcal{P}$ after dividing its elements by the GCD, $\phi(s)$, and $\tilde{S}^{(d)}_{\mathcal{P}^*}$ is the respective $(m, n+p-d)$ extended Sylvester matrix of $\mathcal{P}^*$.*

We will denote by $S^{(d)}_{\mathcal{P}^*} = [\mathbb{O}_{m,d} | \tilde{S}^{(d)}_{\mathcal{P}^*}]$ the corresponding $(n,p)$-*expanded generalized resultant* of the reduced coprime set $\mathcal{P}^* = \mathcal{P}^*_{h+1,n-d}$ . The following results give an important property of generalized resultants [22, 77].

**Theorem 5.2.** *Let $\mathcal{P} \in \Pi(n, p; h+1)$ be a polynomial set (5.1) and $S_{\mathcal{P}}$ the respective generalized resultant (5.5). Then*

$$\rho(S_{\mathcal{P}}) = n + p - d \;\Leftrightarrow\; \deg\{\gcd\{\mathcal{P}\}\} = d$$

**Proposition 5.1.** *The GCD of $\mathcal{P}$ is the same as the GCD of $S[\mathcal{P}]$, that is*

$$\gcd\{\mathcal{P}\} = \gcd\{S[\mathcal{P}]\}$$

## 5.3 The notion of the approximate GCD

We consider the notion of the *approximate GCD* and the development of a computational procedure that allows the evaluation of how good is the given approximate GCD. Defining approximate notions of GCD using the pairwise Euclidean approach has been an issue that has attracted a lot of attention recently [20, 61, 67]. It is well known that, when working with inexact data in a computational environment with limited numerical accuracy, the outcome of a numerical algorithm is usually an approximation of the expected exact solution due to the accumulation of numerical errors. Concerning the GCD algorithms, their solution can be considered either as an approximate solution of the original set of polynomials, within a tolerance $\varepsilon$, or as the exact solution of a perturbed set of polynomials. The following definition is typical for the approximate GCD [16, 20, 53, 61, 67].

**Definition 5.3.** Let $\mathcal{P}_{h+1,n} = \{a(s), b_i(s),\ i = 1, \ldots, h\}$ be a set of univariate polynomials as defined in (5.1) and $\varepsilon > 0$ a fixed numerical tolerance. An almost common divisor ($\varepsilon$-divisor) of the polynomials of the set $\mathcal{P}_{h+1,n}$ is an exact common

divisor of a perturbed set of polynomials

$$\mathcal{P}'_{h+1,n} \triangleq \{a(s) + \Delta a(s),\ b_i(s) + \Delta b_i(s),\ i = 1, \ldots, h\}$$

where the polynomial perturbations satisfy

$$\deg\{\Delta a(s)\} \leq \deg\{a(s)\}, \quad \deg\{\Delta b_i(s)\} \leq \deg\{b_i(s)\}$$

and

$$\|\Delta a(s)\|^2 + \sum_{i=1}^{h} \|\Delta b_i(s)\|^2 < \varepsilon$$

An approximate GCD (or $\varepsilon$-GCD) of the set $\mathcal{P}_{h+1,n}$ is an $\varepsilon$-divisor of maximum degree.

A different approach is presented in [21, 42] where the approximate GCD is defined as a distance problem in a projective space. The particular optimis ation problem is formulated by exploiting the resultant properties of the GCD and applies to any number of polynomials without resorting to the features of a particular algorithm. The essence of current methods for introduction of approximate GCD is the relaxation of conditions characterizing the exact notion. Furthermore, the quality or *strength* of a given approximate GCD is defined by the size of the minimal perturbation required to make a chosen approximate GCD an exact GCD of a perturbed set of polynomials [21, 42]. These significant results for the approximate GCD problem are summarized in the following.

Consider a set $\mathcal{P}_{h+1,n} \in \Pi(n, p; h+1)$ as defined in (5.1) and (5.2). Then

$$a(s) = \underline{a}^t\, \underline{e}'_n(s), \quad b_i(s) = \underline{b}_i^t\, \underline{e}'_p(s),\ i = 1, \ldots, h$$

with $\underline{e}'_j(s) = [s^j, s^{j-1}, \ldots, s, 1]^t$ for $j = n$ or $p$ respectively. We may associate with the set $\mathcal{P}_{h+1,n}$ the vector

$$\underline{p}_{h+1,n} = \left[\ \underline{a}^t,\ \ \underline{b}_1^t,\ \ \ldots,\ \ \underline{b}_h^t\ \right]^t \in \mathbb{R}^N \tag{5.7}$$

where $N = (n+1) + h\,(p+1)$, or alternatively a point $P_{h+1,n}$ in the projective space $P^{N-1}$. The set $\Pi(n, p; h+1)$ is clearly isomorphic with $\mathbb{R}^N$, or $P^{N-1}$. An important question relates to the characterisation of all points of $P^{N-1}$, which correspond to sets of polynomials with a given degree GCD. Such sets of polynomials correspond to certain varieties of $P^{N-1}$, which are defined below. We first note that an alternative representation of $\mathcal{P}_{h+1,n}$ is provided by the generalised resultant $S_{\mathcal{P}} \in \mathbb{R}^{(p+hn)\times(n+p)}$ which is a matrix defined by the vector of coefficients $\underline{p}_{h+1,n}$. If we denote by $C_k(\cdot)$ the $k^{th}$ compound of $S_{\mathcal{P}}$ [56], then the varieties characterising the sets having a given degree $d$ GCD, are defined below.

**Proposition 5.2** ([21, 22, 42])**.** *Let* $\Pi(n,p;h+1)$ *be the set of all polynomial sets* $\mathcal{P}_{h+1,n}$ *with* $h+1$ *elements and with the two higher degrees* $(n,p)$, $n \geq p$ *and let* $S_{\mathcal{P}}$ *be the extended Sylvester matrix of the general set* $\mathcal{P}_{h+1,n}$*. The variety of* $P^{N-1}$*, which characterise all sets* $\mathcal{P}_{h+1,n}$ *having a GCD with degree* $d$*,* $0 < d \leq p$ *is defined by the set of equations*

$$C_{n+p-d+1}\left(S_{\mathcal{P}}\right) = 0 \tag{5.8}$$

Conditions (5.8) define polynomial equations in the parameters of the vector $\underline{p}_{h+1,n}$, or the point $\mathcal{P}_{h+1,n}$ of $P^{N-1}$. The set of equations in (5.8) define a variety of $P^{N-1}$, which will be denoted by $\Delta_d(n,p;h+1)$ and referred to as the *d-GCD variety* of $P^{N-1}$. $\Delta_d(n,p;h+1)$ characterises all sets in $\Pi(n,p;h+1)$, which have a GCD with degree $d$.

*REMARK* 5.1. The sets $\Delta_d(n,p;h+1)$ have measure zero [33] and thus the existence of a nontrivial GCD of degree $d>0$ is a nongeneric property.

The important question now, is how close the given set $\mathcal{P}_{h+1,n}$ is to the given variety $\Delta_d(n,p;h+1)$. Defining the notion of the *approximate GCD* is linked to introducing an appropriate distance of $\mathcal{P}_{h+1,n}$ from $\Delta_d(n,p;h+1)$. In fact, if $\mathcal{Q}^i_{h+1,n}$ is some perturbation set (to be properly defined) and assuming that $\mathcal{P}'^{\,i}_{h+1,n} = \mathcal{P}_{h+1,n} + \mathcal{Q}^i_{h+1,n}$ such that $\mathcal{P}'^{\,i}_{h+1,n} \in \Delta_d(n,p;h+1)$, then the GCD of $\mathcal{P}'^{\,i}_{h+1,n}$, $\phi(s)$, with degree $d$ defines the notion of the approximate GCD and its strength is defined by the "size" of the perturbation $\mathcal{Q}^i_{h+1,n}$. Numerical procedures, such as ERES, produce estimates of an approximate GCD. Estimating the size of the corresponding perturbations provides the means to evaluate how good such approximations are. By letting the parameters of the GCD free (arbitrary) and searching for the minimal size of the corresponding perturbations a distance problem is formulated that is linked to the definition of the *optimal approximate GCD*. The key questions which have to be considered for such studies are:

i) Existence of perturbations of $\mathcal{P}_{h+1,n}$ yielding

$$\mathcal{P}'_{h+1,n} = \mathcal{P}_{h+1,n} + \mathcal{Q}_{h+1,n} \ \in \Delta_d(n,p;h+1)$$

ii) Parametrisations of all such perturbations.

iii) Determine the minimal distance of $\mathcal{P}_{h+1,n}$ from an element of $\Delta_d(n,p;h+1)$ with a given GCD $u(s)$, and thus evaluation of strength of $u(s)$.

iv) Determine the minimal distance of $\mathcal{P}_{h+1,n}$ from $\Delta_d(n,p;h+1)$ and thus compute the *optimal approximate GCD*.

Here we are concerned with the issues (i)-(iii) which relate to the evaluation of the strength of a given approximation that is not necessarily optimal.

## 5.4 Parametrisation of GCD varieties and definition of the Strength of the approximate GCD

The characterisation of the $\Delta_d(n,p;h+1)$ variety in a parametric form, as well as subvarieties of it, is a crucial issue for the further development of the topic. The subset of $\Delta_d(n,p;h+1)$, characterised by the property that all $\mathcal{P}_{h+1,n}$ in it have a given GCD $u(s)\in\mathbb{R}[s]$, $\deg\{u(s)\}=d$, can be shown to be a subvariety of $\Delta_d(n,p;h+1)$ and is denoted by $\Delta_d^u(n,p;h+1)$ [21, 42]. In fact $\Delta_d^u(n,p;h+1)$ is characterised by the equations of $\Delta_d(n,p;h+1)$ and a set of additional linear relations amongst the parameters of the vector $\underline{p}_{h+1,n}$.

**Proposition 5.3.** *Consider the set $\Pi(n,p;h+1)$, $P^{N-1}$ be the associated projective space, $\mathcal{P}_{h+1,n}\in\Pi(n,p;h+1)$ and let $S_{\mathcal{P}}$ be the associated resultant. Then,*

*i) The variety $\Delta_d(n,p;h+1)$ of $P^{N-1}$ is expressed parametrically by the generalized resultant:*

$$S_{\mathcal{P}}=\left[\mathbb{O}_{m,d}|\tilde{S}^{(d)}_{\mathcal{P}^*}\right]\cdot\hat{\Phi}_u \tag{5.9}$$

*where $\mathbb{O}_{m,d}$ is the $m\times d$ zero matrix, $m=p+hn$, $\hat{\Phi}_u$ is the $(n+p)\times(n+p)$ Toeplitz representation of an arbitrary $u(s)\in\mathbb{R}[s]$ with $\deg\{u(s)\}=d$ and $\tilde{S}^{(d)}_{\mathcal{P}^*}\in\mathbb{R}^{m\times(n+p-d)}$ is the $(n,d)$-expanded generalized resultant of an arbitrary set of polynomials $\mathcal{P}^*\in\Pi(n-d,p-d;h+1)$.*

*ii) The variety $\Delta_d^u(n,p;h+1)$ of $P^{N-1}$ is defined by (5.9) with the additional constraint that $u(s)\in\mathbb{R}[s]$ is given.*

Clearly, the free parameters in $\Delta_d(n,p;h+1)$ are the coefficients of the polynomials of $\Pi(n-d,p-d;h+1)$. Having defined the description of these varieties we consider next the perturbations that transfer a general set $\mathcal{P}_{h+1,n}$ on a set $\mathcal{P}'_{h+1,n}$ on them. If $\mathcal{P}_{h+1,n}\in\Pi(n,p;h+1)$ we can define an $(n,p)$-ordered perturbed set $\mathcal{P}'_{h+1,n}\in\Pi(n,p;h+1)$ by

$$\begin{aligned}\mathcal{P}'_{h+1,n} &\triangleq \mathcal{P}_{h+1,n}-\mathcal{Q}_{h+1,n} \\ &= \left\{p'_i(s)=p_i(s)-q_i(s):\deg\{q_i(s)\}\le\deg\{p_i(s)\},i=0,\ldots,h\right\}\end{aligned} \tag{5.10}$$

Using the set of perturbations defined above we may now show that any polynomial from a certain class may become an exact GCD of a perturbed set under a family of perturbations.

**Proposition 5.4** ([21, 42])**.** *Given a set $\mathcal{P}_{h+1,n}$ with maximal degrees $(n,p)$, $n\ge p$ and a polynomial $v(s)\in\mathbb{R}[s]$ with $\deg\{v(s)\}\le p$. There always exists a family of $(n,p)$-ordered perturbations $\mathcal{Q}_{h+1,n}$ such that for every element of this family $\mathcal{P}'_{h+1,n}=\mathcal{P}_{h+1,n}-\mathcal{Q}_{h+1,n}$ has a GCD which is divisible by $v(s)$.*

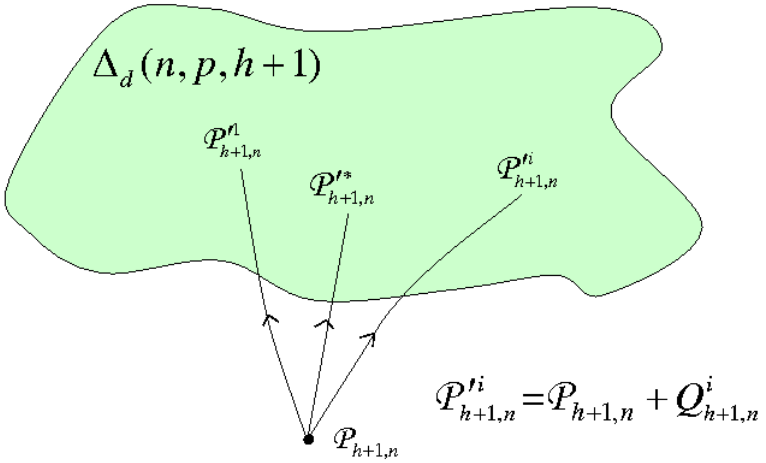


Figure 5.1: The notion of the "approximate GCD".

The above result establishes the existence of perturbations making $v(s)$ an exact GCD of the perturbed set and motivates the following definition, which defines $v(s)$ as an *approximate GCD* in an optimal sense [21, 42].

**Definition 5.4.** Let $\mathcal{P}_{h+1,n} \in \Pi(n,p;h+1)$ and $v(s) \in \mathbb{R}[s]$ be a given polynomial with $\deg\{v(s)\} = r \leq p$. Furthermore, let $\Sigma_v = \{\mathcal{Q}_{h+1,n}\}$ be the set of all $(n,p)$-order perturbations such that

$$\mathcal{P}'_{h+1,n} = \mathcal{P}_{h+1,n} - \mathcal{Q}_{h+1,n} \in \Pi(n,p;h+1) \tag{5.11}$$

with the property that $v(s)$ is a common factor of the elements of $\mathcal{P}'_{h+1,n}$. If $\mathcal{Q}^{\circ}_{h+1,n}$ is the minimal norm element of the set $\Sigma_v$, then $v(s)$ is referred to as an *r-order almost common factor* of $\mathcal{P}_{h+1,n}$, and the norm of $\mathcal{Q}^{\circ}_{h+1,n}$, denoted by $\|\mathcal{Q}^{\circ}\|$ is defined as the *strength* of $v(s)$. If $v(s)$ is the GCD of

$$\mathcal{P}^{\circ}_{h+1,n} = \mathcal{P}_{h+1,n} - \mathcal{Q}^{\circ}_{h+1,n} \tag{5.12}$$

then $v(s)$ will be called an *r-order almost GCD* of $\mathcal{P}_{h+1,n}$ with strength $\|\mathcal{Q}^{\circ}\|$.

Thus, any polynomial *v(s)* may be considered as an approximate GCD, provided $r = \deg\{v(s)\} \leq p$.

**Theorem 5.3** ([21, 42])**.** *For $\mathcal{P}_{h+1,n} \in \Pi(n,p;h+1)$, let $S_{\mathcal{P}} \in \Psi(n,p;h+1)$ be the corresponding generalized resultant and let $v(s) = \lambda_r\, s^r + \ldots + \lambda_1\, s + \lambda_0 \in \mathbb{R}[s]$, $\deg\{v(s)\} = r \leq p$. Then,*

*i) Any perturbation set $\mathcal{Q}_{h+1,n} \in \Pi(n,p;h+1)$ that leads to*

$$\mathcal{P}'_{h+1,n} = \mathcal{P}_{h+1,n} - \mathcal{Q}_{h+1,n}$$

*which has the polynomial $v(s)$ as common divisor, has a generalized resultant $S_{\mathcal{Q}} \in \Psi(n, p; h+1)$ that is expressed as shown below:*

*a) If $v(0) \neq 0$ then*

$$S_{\mathcal{Q}} = S_{\mathcal{P}} - S^{(r)}_{\mathcal{P}^*} \cdot \hat{\Phi}_v = S_{\mathcal{P}} - \left[\mathbb{O}_{m,r} | \tilde{S}^{(r)}_{\mathcal{P}^*}\right] \cdot \hat{\Phi}_v \tag{5.13}$$

*where $\mathbb{O}_{m,r}$ is the $m \times r$ zero matrix, $\hat{\Phi}_v$ is the $(n+p) \times (n+p)$ Toeplitz representation of $v(s)$ as defined in (5.4) and $S^{(r)}_{\mathcal{P}^*} \in \mathbb{R}^{m \times (n+p)}$ is the $(n, p)$-expanded generalized resultant of an arbitrary set of polynomials $\mathcal{P}^* \in \Pi(n-r, p-r; h+1)$.*

*b) If $v(s)$ has $k$ zeros at $s = 0$, then*

$$S_{\mathcal{Q}} = S_{\mathcal{P}} - \tilde{S}^{(r)}_{\mathcal{P}^*} \cdot \Theta_v \tag{5.14}$$

*where $\tilde{S}^{(r)}_{\mathcal{P}^*}$ is again the $(n, p)$-expanded generalized resultant of an arbitrary set of polynomials $\mathcal{P}^* \in \Pi(n-r, p-r; h+1)$ and $\Theta_v$ is the $(n+p-k) \times (n+p)$ representation of $v(s)$ defined by*

$$\Theta_v = \begin{bmatrix} \lambda_k & \lambda_{k-1} & \lambda_{k-2} & \cdots & \cdots & \lambda_0 & 0 & \cdots & \cdots & 0 \\ 0 & \lambda_k & \lambda_{k-1} & \lambda_{k-2} & \cdots & \cdots & \lambda_0 & 0 & & 0 \\ \vdots & \ddots & \ddots & & & & & \ddots & & \\ \vdots & & & \ddots & & & & & \ddots & \\ 0 & \cdots & \cdots & 0 & \lambda_k & \lambda_{k-1} & \lambda_{k-2} & \cdots & \cdots & \lambda_0 \end{bmatrix} \tag{5.15}$$

*ii) If the parameters of $\mathcal{P}^*$ are constrained such that $\tilde{S}^{(r)}_{\mathcal{P}^*}$ has full rank, then $v(s)$ is a GCD of the perturbed set $\mathcal{P}'_{h+1,n}$.*

**Corollary 5.1** ([21, 42])**.** *Let $\mathcal{P}_{h+1,n} \in \Pi(n, p; h+1)$ and $v(s) \in \mathbb{R}[s]$, $\deg\{v(s)\} = r \leq p$. The polynomial $v(s)$ is an $r$-order almost common divisor of $\mathcal{P}_{h+1,n}$ and its strength is defined as a solution of the following minimization problems:*

*a) If $v(0) \neq 0$, then its strength is defined by the global minimum of*

$$f(\mathcal{P}, \mathcal{P}^*) = \min_{\forall \mathcal{P}^*} \left\| S_{\mathcal{P}} - \left[\mathbb{O}_{m,r} | \tilde{S}^{(r)}_{\mathcal{P}^*}\right] \cdot \hat{\Phi}_v \right\|_F \tag{5.16}$$

*b) If $v(s)$ has $k$ zeros at $s = 0$, then its strength is defined by the global minimum of*

$$f(P, \mathcal{P}^*) = \min_{\forall \mathcal{P}^*} \|S_{\mathcal{P}} - \tilde{S}^{(r)}_{\mathcal{P}^*} \cdot \Theta_v\|_F \tag{5.17}$$

*where $\mathcal{P}^*$ takes values from the set $\Pi(n, p; h+1)$.*

*Furthermore, $v(s)$ is an $r$-order almost GCD of $\mathcal{P}_{h+1,n}$, if the minimal corresponds to a coprime set $\mathcal{P}^*$ or to full rank $S_{\mathcal{P}^*}$.*

## 5.5 The numerical computation of the strength of an approximate GCD

For the computation of the minimization problems in (5.16) or (5.17) we need an appropriate numerical procedure. However, the successful computation of such a global minimum is not always guaranteed. The minimization problem in (5.16) or (5.17) is actually non-convex and in cases of sets of many polynomials, where the number of arbitrary parameters is usually large, it is very likely to lead to unsatisfactory results. Conversely, it is easier to find some bounds for the main function in (5.16) which is

$$\|S_{\mathcal{Q}}\| = \left\| S_{\mathcal{P}} - \left[\mathbb{O}_{m,r} | \tilde{S}^{(r)}_{\mathcal{P}^*}\right] \cdot \hat{\Phi}_v \right\|$$

We analyse how the norm $\|S_{\mathcal{Q}}\|$ is bounded and what information we can get from these bounds. Without loss of generality, we assume a given polynomial $v(s)$ with no zero roots. Combining the relations (5.4) and (5.13), gives the following equation:

$$S_{\mathcal{Q}} \cdot \hat{\Phi}_v^{-1} = S_{\mathcal{P}} \cdot \hat{\Phi}_v^{-1} - \left[\mathbb{O}_{m,r} | \tilde{S}^{(r)}_{\mathcal{P}^*}\right] \tag{5.18}$$

Let $\widehat{S}_{\mathcal{P}} = S_{\mathcal{P}} \cdot \hat{\Phi}_v^{-1}$ and split $\widehat{S}_{\mathcal{P}}$ such that

$$\widehat{S}_{\mathcal{P}} = \hat{S}'_{\mathcal{P}} + \hat{S}''_{\mathcal{P}} \tag{5.19}$$

where $\hat{S}''_{\mathcal{P}}$ has the same structure as $S^{(r)}_{\mathcal{P}^*} = \left[\mathbb{O}_{m,r} | \tilde{S}^{(r)}_{\mathcal{P}^*}\right]$. Specifically, if we denote by $A[i,j]$ the $(i,j)$ element of a matrix $A$, the partitioning of $\widehat{S}_{\mathcal{P}}$ is based on the next rule:

$$\hat{S}'_{\mathcal{P}}[i,j] = \begin{cases} \widehat{S}_{\mathcal{P}}[i,j], & \text{if} \quad S^{(r)}_{\mathcal{P}^*}[i,j] = 0 \\ 0, & \text{if} \quad S^{(r)}_{\mathcal{P}^*}[i,j] \neq 0 \end{cases} \quad \forall\, i,j \tag{5.20}$$

Therefore, $\hat{S}''_{\mathcal{P}}$ can be presented as $\hat{S}''_{\mathcal{P}} = \left[\mathbb{O}_{m,r} | \bar{S}\right]$, where $\bar{S}$ is an $m \times (n+p-r)$ matrix. From (5.18) and (5.19) we get the following relations:

$$\begin{aligned} S_{\mathcal{Q}} \cdot \hat{\Phi}_v^{-1} &= \hat{S}'_{\mathcal{P}} + \left[\mathbb{O}_{m,r} | \bar{S}\right] - \left[\mathbb{O}_{m,r} | \tilde{S}^{(r)}_{\mathcal{P}^*}\right] = \\ &= \hat{S}'_{\mathcal{P}} + \left[\mathbb{O}_{m,r} | \bar{S} - \tilde{S}^{(r)}_{\mathcal{P}^*}\right] \end{aligned} \tag{5.21}$$

It readily follows that:

$$S_{\mathcal{Q}} = \hat{S}'_{\mathcal{P}} \cdot \hat{\Phi}_v + \left[\mathbb{O}_{m,r} | \bar{S} - \tilde{S}^{(r)}_{\mathcal{P}^*}\right] \cdot \hat{\Phi}_v \tag{5.22}$$

and if we use the Frobenius norm, which relates to the set of polynomials in a

direct way, we get:

$$\|S_{\mathcal{Q}}\| \leq \|\hat{S}'_{\mathcal{P}} \cdot \hat{\Phi}_v\| + \| \left[\mathbb{O}_{m,r} | \bar{S} - \tilde{S}^{(r)}_{\mathcal{P}^*}\right] \cdot \hat{\Phi}_v\| \quad \text{or} \tag{5.23}$$

$$\|S_{\mathcal{Q}}\| \leq \|\hat{S}'_{\mathcal{P}}\| \, \|\hat{\Phi}_v\| + \| \left[\mathbb{O}_{m,r} | \bar{S} - \tilde{S}^{(r)}_{\mathcal{P}^*}\right] \| \, \|\hat{\Phi}_v\| \tag{5.24}$$

Furthermore, from the equation (5.21) we will have:

$$\|S_{\mathcal{Q}}\| \, \|\hat{\Phi}_v^{-1}\| \geq \left\| \hat{S}'_{\mathcal{P}} + \left[\mathbb{O}_{m,r} | \bar{S} - \tilde{S}^{(r)}_{\mathcal{P}^*}\right] \right\| \tag{5.25}$$

It is clear that any exact common factor of the polynomials of the set is expected to give $\|S_{\mathcal{Q}}\| = 0$. Therefore, we may consider a polynomial as a good approximation of an exact common divisor or the exact GCD of a given set, if $\|S_{\mathcal{Q}}\|$ is close enough to zero.

The structure of the matrices here allows us to select the arbitrary parameters of the set $\mathcal{P}^*$ such that

$$\bar{S} = \tilde{S}^{(r)}_{\mathcal{P}^*} \tag{5.26}$$

Then, if we apply the above result (5.26) to the inequalities (5.23) and (5.25) and since the condition number of $\hat{\Phi}_v$ according to the Frobenius norm is

$$Cond(\hat{\Phi}_v) = \|\hat{\Phi}_v\| \, \|\hat{\Phi}_v^{-1}\| \geq n + p > 1 \tag{5.27}$$

the following important inequality will be obtained:

$$\frac{\|\hat{S}'_{\mathcal{P}}\|}{\|\hat{\Phi}_v^{-1}\|} \leq \|S_{\mathcal{Q}}\| \leq \|\hat{S}'_{\mathcal{P}} \cdot \hat{\Phi}_v\| \tag{5.28}$$

Obviously, if $\|\hat{S}'_{\mathcal{P}}\| = 0$, then $\|S_{\mathcal{Q}}\| = 0$ and therefore the given polynomial $v(s)$ can be considered as an exact common divisor of degree $r$ of the original set. Otherwise, the inequality (5.28) gives a lower bound of $\|S_{\mathcal{Q}}\|$, which indicates the minimum distance towards $\|S_{\mathcal{Q}}\| = 0$.

**Definition 5.5.** Given a polynomial $v(s)$ with no zero roots, we shall define as:

i) $\mathcal{S}(v)$, the *strength* of $v(s)$ given by the minimization problems (5.16), (5.17).

ii) $\underline{\mathcal{S}}(v) \triangleq \|\hat{S}'_{\mathcal{P}}\| \left(\|\hat{\Phi}_v^{-1}\|\right)^{-1}$, the *lower strength bound* of $v(s)$.

iii) $\overline{\mathcal{S}}(v) \triangleq \|\hat{S}'_{\mathcal{P}} \cdot \hat{\Phi}_v\|$, the *upper strength bound* of $v(s)$.

iv) $\mathcal{S}^a(v) \triangleq \dfrac{\underline{\mathcal{S}}(v) + \overline{\mathcal{S}}(v)}{2}$, the *average strength* of $v(s)$.

The computation of the strength bounds $\underline{\mathcal{S}}(v)$ and $\overline{\mathcal{S}}(v)$ is straightforward and the results can be used as an indicator of the strength of the given approximation.

For example, if $\underline{\mathcal{S}}(v) >> 1$, then the given approximation has very poor quality and the opposite holds if $\overline{\mathcal{S}}(v) << 1$. The strength bounds are very reliable indicators of the strength of a given GCD approximation $v(s)$ provided that the respective matrix $\hat{\Phi}_v$ is well-conditioned ( $Cond(\hat{\Phi}_v) \approx O(n+p)$ ).

The following algorithm establishes a method for the evaluation of the strength bounds and hence the average strength $\mathcal{S}^a(v)$ of a given approximation $v(s)$.

---

**ALGORITHM 5.1. The algorithm of Average Strength.**

Input : Give a set $\mathcal{P} \in \Pi(n, p; h+1)$ of univariate polynomials.
Give a univariate polynomial $v(s)$ of degree $r \leq p$ with no zero roots.

Step 1 : Construct the $(n,p)$-expanded Sylvester matrix $S_{\mathcal{P}}$ of $\mathcal{P}$.
Construct the special Toeplitz representation $\hat{\Phi}_v$ of $v(s)$.
Compute the first column of the inverse of $\hat{\Phi}_v$ and construct the matrix $\hat{\Phi}_v^{-1}$.

Step 2 : Compute the matrix $\widehat{S}_{\mathcal{P}}$ by solving the linear system : $\hat{\Phi}_v^{\,t} \cdot \widehat{S}_{\mathcal{P}}^{\,t} = S_{\mathcal{P}}^{\,t}$

Step 3 : Split $\widehat{S}_{\mathcal{P}}$ such that $\widehat{S}_{\mathcal{P}} = \hat{S}'_{\mathcal{P}} + \hat{S}''_{\mathcal{P}}$ using (5.20) .
Compute the Frobenius norms $\|\hat{S}'_{\mathcal{P}}\|$, $\|\hat{\Phi}_v^{-1}\|$ and $\|\hat{S}'_{\mathcal{P}} \cdot \hat{\Phi}_v\|$.

Output : $\underline{\mathcal{S}}(v) = \dfrac{\|\hat{S}'_{\mathcal{P}}\|}{\|\hat{\Phi}_v^{-1}\|}$ , $\overline{\mathcal{S}}(v) = \|\hat{S}'_{\mathcal{P}} \cdot \hat{\Phi}_v\|$, $\mathcal{S}^a(v) = \dfrac{\underline{\mathcal{S}}(v) + \overline{\mathcal{S}}(v)}{2}$

---

### ► Computational complexity

Due to the special structure of the matrices $S_{\mathcal{P}}$ and $\hat{\Phi}_v$, it is possible to avoid the matrix operations and compute the norms explicitly and more efficiently. The inverse of the lower triangular matrix $\hat{\Phi}_v$ is computed by solving a simple linear system of the form $\hat{\Phi}_v \, \underline{x} = \underline{e}_{n+p}^1$, where $\underline{x}$ represents the first column of the matrix $\hat{\Phi}_v^{-1}$ and $\underline{e}_{n+p}^1 = [1, 0, \ldots, 0]^t \in \mathbb{R}^{n+p}$. The multiple linear system $\hat{\Phi}_v^{\,t} \cdot \widehat{S}_{\mathcal{P}}^{\,t} = S_{\mathcal{P}}^{\,t}$ can be solved by using only backward substitution since $\hat{\Phi}_v^{\,t}$ is an upper triangular matrix. The multiplication $\hat{S}'_{\mathcal{P}} \cdot \hat{\Phi}_v$ actually zeros specific entries of $\hat{S}'_{\mathcal{P}}$ and produces an extended-resultant-like matrix. Thus it is not required to perform it. The number of operations required by the above algorithm is given in Table 5.1. The total amount of operations is $O\left(\frac{3n^2h+10n^2+2nr-r^2}{2}\right)$ for $n = p$. For an effective computation of the GCD, the respective matrix $\hat{S}'_{\mathcal{P}}$ is quite sparse and the required operations are less than $O(2hn^2)$.

| $\hat{\Phi}_v^{-1}$ | $\widehat{S}_{\mathcal{P}}$ | $\|\hat{S}'_{\mathcal{P}} \cdot \hat{\Phi}_v\|$ | $\|\hat{\Phi}_v^{-1}\|$ | $\|\hat{\Phi}_v\|$ |
|---|---|---|---|---|
| $O\left(\frac{(n+p)^2}{2}\right)$ | $O\left(\frac{n^2+p^2h}{2}\right)$ | $O\big(hnp - hr(n+p)\big)$ | $O\left(\frac{(n+p)^2}{2}\right)$ | $O\left(\frac{(n+p)r-r^2}{2}\right)$ |

Table 5.1: Required operations for the computation of the strength bounds.

### ▶ Computational examples

In the following, we will demonstrate the steps of the previous Algorithm 5.1 for computing the average strength of a given approximation by considering the next example.

**Example 5.1.** Consider a polynomial set $\mathcal{P}_{3,3} \in \Pi(3,2;3)$ with arbitrary coefficients. The set $\mathcal{P}_{3,3}$ will be a $\{3,2\}$-ordered polynomial defined as:

$$\mathcal{P}_{3,3} = \left\{ \begin{array}{lcl} a(s) & = & s^3 + a_2 s^2 + a_1 s + a_0 \\ b_1(s) & = & s^2 + b_{1,1} s + b_{1,0} \\ b_2(s) & = & s^2 + b_{2,1} s + b_{2,0} \end{array} \right\} \tag{5.29}$$

According to the representation (5.5), the generalised resultant matrix of the set $\mathcal{P}_{3,3}$ has the form:

$$S_{\mathcal{P}} = \left[ \begin{array}{ccccc} 1 & a_2 & a_1 & a_0 & 0 \\ 0 & 1 & a_2 & a_1 & a_0 \\ 1 & b_{1,1} & b_{1,0} & 0 & 0 \\ 0 & 1 & b_{1,1} & b_{1,0} & 0 \\ 0 & 0 & 1 & b_{1,1} & b_{1,0} \\ 1 & b_{2,1} & b_{2,0} & 0 & 0 \\ 0 & 1 & b_{2,1} & b_{2,0} & 0 \\ 0 & 0 & 1 & b_{2,1} & b_{2,0} \end{array} \right] \tag{5.30}$$

For simplicity reasons and without loss of generality we will consider a $1^{st}$ degree GCD approximation for the set $\mathcal{P}_{3,3}$ such that

$$v(s) = s + c \quad , \; c \neq 0$$

As it was described in Definition 5.3, a Toeplitz-like matrix can represent the polynomial $v(s)$ in the form:

$$\hat{\Phi}_v = \begin{bmatrix} c & 0 & 0 & 0 & 0 \\ 1 & c & 0 & 0 & 0 \\ 0 & 1 & c & 0 & 0 \\ 0 & 0 & 1 & c & 0 \\ 0 & 0 & 0 & 1 & c \end{bmatrix} \tag{5.31}$$

Then, the respective inverse of $\hat{\Phi}_v$ is a lower triangular matrix of the form:

$$\hat{\Phi}_v^{-1} = \begin{bmatrix} c^{-1} & 0 & 0 & 0 & 0 \\ -c^{-2} & c^{-1} & 0 & 0 & 0 \\ c^{-3} & -c^{-2} & c^{-1} & 0 & 0 \\ -c^{-4} & c^{-3} & -c^{-2} & c^{-1} & 0 \\ c^{-5} & -c^{-4} & c^{-3} & -c^{-2} & c^{-1} \end{bmatrix} \tag{5.32}$$

The next computations are made by following the steps 2 and 3 of the Algorithm 5.1 and lead to the evaluation of the strength bounds and the average strength of the approximation $v(s)$.

$$\widehat{S}_{\mathcal{P}} = S_{\mathcal{P}} \cdot \hat{\Phi}_v^{-1} = \tag{5.33}$$

$$\left[\begin{array}{c|c|c|c|c} \frac{1}{c} - \frac{a_2}{c^2} + \frac{a_1}{c^3} - \frac{a_0}{c^4} & \frac{a_2}{c} - \frac{a_1}{c^2} + \frac{a_0}{c^3} & \frac{a_1}{c} - \frac{a_0}{c^2} & \frac{a_0}{c} & 0 \\ -\frac{1}{c^2} + \frac{a_2}{c^3} - \frac{a_1}{c^4} + \frac{a_0}{c^5} & \frac{1}{c} - \frac{a_2}{c^2} + \frac{a_1}{c^3} - \frac{a_0}{c^4} & \frac{a_2}{c} - \frac{a_1}{c^2} + \frac{a_0}{c^3} & \frac{a_1}{c} - \frac{a_0}{c^2} & \frac{a_0}{c} \\ \frac{1}{c} - \frac{b_{1,1}}{c^2} + \frac{b_{1,0}}{c^3} & \frac{b_{1,1}}{c} - \frac{b_{1,0}}{c^2} & \frac{b_{1,0}}{c} & 0 & 0 \\ -\frac{1}{c^2} + \frac{b_{1,1}}{c^3} - \frac{b_{1,0}}{c^4} & \frac{1}{c} - \frac{b_{1,1}}{c^2} + \frac{b_{1,0}}{c^3} & \frac{b_{1,1}}{c} - \frac{b_{1,0}}{c^2} & \frac{b_{1,0}}{c} & 0 \\ \frac{1}{c^3} - \frac{b_{1,1}}{c^4} + \frac{b_{1,0}}{c^5} & -\frac{1}{c^2} + \frac{b_{1,1}}{c^3} - \frac{b_{1,0}}{c^4} & \frac{1}{c} - \frac{b_{1,1}}{c^2} + \frac{b_{1,0}}{c^3} & \frac{b_{1,1}}{c} - \frac{b_{1,0}}{c^2} & \frac{b_{1,0}}{c} \\ \frac{1}{c} - \frac{b_{2,1}}{c^2} + \frac{b_{2,0}}{c^3} & \frac{b_{2,1}}{c} - \frac{b_{2,0}}{c^2} & \frac{b_{2,0}}{c} & 0 & 0 \\ -\frac{1}{c^2} + \frac{b_{2,1}}{c^3} - \frac{b_{2,0}}{c^4} & \frac{1}{c} - \frac{b_{2,1}}{c^2} + \frac{b_{2,0}}{c^3} & \frac{b_{2,1}}{c} - \frac{b_{2,0}}{c^2} & \frac{b_{2,0}}{c} & 0 \\ \frac{1}{c^3} - \frac{b_{2,1}}{c^4} + \frac{b_{2,0}}{c^5} & -\frac{1}{c^2} + \frac{b_{2,1}}{c^3} - \frac{b_{2,0}}{c^4} & \frac{1}{c} - \frac{b_{2,1}}{c^2} + \frac{b_{2,0}}{c^3} & \frac{b_{2,1}}{c} - \frac{b_{2,0}}{c^2} & \frac{b_{2,0}}{c} \end{array}\right]$$

$$\hat{S}'_{\mathcal{P}} = \left[\begin{array}{c|c|c|c|c} \frac{1}{c} - \frac{a_2}{c^2} + \frac{a_1}{c^3} - \frac{a_0}{c^4} & 0 & 0 & 0 & 0 \\ -\frac{1}{c^2} + \frac{a_2}{c^3} - \frac{a_1}{c^4} + \frac{a_0}{c^5} & \frac{1}{c} - \frac{a_2}{c^2} + \frac{a_1}{c^3} - \frac{a_0}{c^4} & 0 & 0 & 0 \\ \frac{1}{c} - \frac{b_{1,1}}{c^2} + \frac{b_{1,0}}{c^3} & 0 & 0 & 0 & 0 \\ -\frac{1}{c^2} + \frac{b_{1,1}}{c^3} - \frac{b_{1,0}}{c^4} & \frac{1}{c} - \frac{b_{1,1}}{c^2} + \frac{b_{1,0}}{c^3} & 0 & 0 & 0 \\ \frac{1}{c^3} - \frac{b_{1,1}}{c^4} + \frac{b_{1,0}}{c^5} & -\frac{1}{c^2} + \frac{b_{1,1}}{c^3} - \frac{b_{1,0}}{c^4} & \frac{1}{c} - \frac{b_{1,1}}{c^2} + \frac{b_{1,0}}{c^3} & 0 & 0 \\ \frac{1}{c} - \frac{b_{2,1}}{c^2} + \frac{b_{2,0}}{c^3} & 0 & 0 & 0 & 0 \\ -\frac{1}{c^2} + \frac{b_{2,1}}{c^3} - \frac{b_{2,0}}{c^4} & \frac{1}{c} - \frac{b_{2,1}}{c^2} + \frac{b_{2,0}}{c^3} & 0 & 0 & 0 \\ \frac{1}{c^3} - \frac{b_{2,1}}{c^4} + \frac{b_{2,0}}{c^5} & -\frac{1}{c^2} + \frac{b_{2,1}}{c^3} - \frac{b_{2,0}}{c^4} & \frac{1}{c} - \frac{b_{2,1}}{c^2} + \frac{b_{2,0}}{c^3} & 0 & 0 \end{array}\right] \tag{5.34}$$

$$\hat{S}'_{\mathcal{P}} \cdot \hat{\Phi}_v = \left[\begin{array}{c|c|c|c|c} \frac{c^3 - a_2c^2 + a_1c - a_0}{c^3} & 0 & 0 & 0 & 0 \\ 0 & \frac{c^3 - a_2c^2 + a_1c - a_0}{c^3} & 0 & 0 & 0 \\ \frac{c^2 - b_{1,1}c + b_{1,0}}{c^2} & 0 & 0 & 0 & 0 \\ 0 & \frac{c^2 - b_{1,1}c + b_{1,0}}{c^2} & 0 & 0 & 0 \\ 0 & 0 & \frac{c^2 - b_{1,1}c + b_{1,0}}{c^2} & 0 & 0 \\ \frac{c^2 - b_{2,1}c + b_{2,0}}{c^2} & 0 & 0 & 0 & 0 \\ 0 & \frac{c^2 - b_{2,1}c + b_{2,0}}{c^2} & 0 & 0 & 0 \\ 0 & 0 & \frac{c^2 - b_{2,1}c + b_{2,0}}{c^2} & 0 & 0 \end{array}\right] \tag{5.35}$$

Now, we can give the algebraic expression of the strength bounds using the Frobenius matrix norm. The upper strength bound for $v(s)$ is:

$$\overline{\mathcal{S}}(v) = \|\hat{S}'_{\mathcal{P}} \cdot \hat{\Phi}_v\|_F = \tag{5.36}$$

$$= \left(2\left(\frac{c^3 - a_2c^2 + a_1c - a_0}{c^3}\right)^2 + 3\left(\frac{c^2 - b_{2,1}c + b_{2,0}}{c^2}\right)^2 + 3\left(\frac{c^2 - b_{1,1}c + b_{1,0}}{c^2}\right)^2\right)^{\frac{1}{2}}$$

and the lower strength bound of $v(s)$ is:

$$
\begin{aligned}
\underline{\mathcal{S}}(v) &= \|\hat{S}'_{\mathcal{P}}\|_F \cdot \left(\|\hat{\Phi}_v^{-1}\|_F\right)^{-1} = \qquad (5.37)\\
&= \left(2\left(\frac{1}{c}-\frac{a_2}{c^2}+\frac{a_1}{c^3}-\frac{a_0}{c^4}\right)^2+\left(-\frac{1}{c^2}+\frac{a_2}{c^3}-\frac{a_1}{c^4}+\frac{a_0}{c^5}\right)^2\right.\\
&\quad +3\left(\frac{1}{c}-\frac{b_{2,1}}{c^2}+\frac{b_{2,0}}{c^3}\right)^2-2\left(\frac{1}{c^2}-\frac{b_{2,1}}{c^3}+\frac{b_{2,0}}{c^4}\right)^2\\
&\quad +\left(\frac{1}{c^3}-\frac{b_{2,1}}{c^4}+\frac{b_{2,0}}{c^5}\right)^2+3\left(\frac{1}{c}-\frac{b_{1,1}}{c^2}+\frac{b_{1,0}}{c^3}\right)^2\\
&\quad \left.-2\left(\frac{1}{c^2}-\frac{b_{1,1}}{c^3}+\frac{b_{1,0}}{c^4}\right)^2+\left(\frac{1}{c^3}-\frac{b_{1,1}}{c^4}+\frac{b_{1,0}}{c^5}\right)^2\right)^{\frac{1}{2}}\cdot\\
&\quad \cdot\left(\frac{5\,c^8+4\,c^6+3\,c^4+2\,c^2+1}{c^{10}}\right)^{-\frac{1}{2}}
\end{aligned}
$$

The condition number of the matrix $\hat{\Phi}_v$ characterises the stability of the computation of the inverse of $\hat{\Phi}_v$ [18, 81] which also affects the computation of the matrix $\hat{S}'_{\mathcal{P}}$ and hence the strength bounds. Since we use the Frobenius matrix norm, our computations can be considered sufficiently numerical stable if $Cond(\hat{\Phi}_v)$ is about $O(n+p)$. Otherwise, it is very likely to have an unreliable result. Therefore, when working in a variable precision computational environment it is more preferable to perform the computation of the strength bounds using enough digits of precision, for example quadruple precision (34 digits). In the present case, the condition number of $\hat{\Phi}_v$ is given by

$$
Cond(\hat{\Phi}_v) = \|\hat{\Phi}_v\|_F\,\|\hat{\Phi}_v^{-1}\|_F = \sqrt{25+\frac{40}{c^2}+\frac{31}{c^4}+\frac{22}{c^6}+\frac{13}{c^8}+\frac{4}{c^{10}}} \qquad (5.38)
$$

and clearly the minimum value that we can get from it is $n+p=3+2=5$. When solving the linear system $\hat{\Phi}_v^{\,t}\cdot\widehat{S}_{\mathcal{P}}^{\,t}=S_{\mathcal{P}}^{\,t}$ the condition number of $\hat{\Phi}_v$ may act as an indicator for using scaling techniques [18] in order to prevent unreliable results. □

**Example 5.2.** Consider a set of polynomials $\mathcal{P}\in\Pi\{3,2;3\}$ with numeric floating-point coefficients:

$$
\mathcal{P}=\left\{\begin{array}{lcll} a(s) &=& (s-1)(s+1)(s+3) &= s^3+3\,s^2-s-3\\ b_1(s) &=& (s-2)(s+0.9995) &= s^2-1.0005\,s-1.9990\\ b_2(s) &=& (s-3)(s+1.0050) &= s^2-0.9950\,s-3.0150 \end{array}\right\} \qquad (5.39)
$$

which obviously has the same structure as (5.29). Then, the respective generalised resultant matrix of $\mathcal{P}$ has the form:

$$S_{\mathcal{P}} = \begin{bmatrix} 1 & 3 & -1 & -3 & 0 \\ 0 & 1 & 3 & -1 & -3 \\ 1 & -1.0005 & -1.9990 & 0 & 0 \\ 0 & 1 & -1.0005 & -1.9990 & 0 \\ 0 & 0 & 1 & -1.0005 & -1.9990 \\ 1 & -0.9950 & -3.0150 & 0 & 0 \\ 0 & 1 & -0.9950 & -3.0150 & 0 \\ 0 & 0 & 1 & -0.9950 & -3.0150 \end{bmatrix} \in \mathbb{R}^{8\times 5} \tag{5.40}$$

Clearly, for the typical 16-digit numerical precision the three polynomials of the above set $\mathcal{P}$ are considered coprime. The three polynomials of the set have been constructed so as to have a root around -1. The data are given in 4-digit numerical precision and thus, for 2-digit numerical precision the polynomials should have approximately the same root. However, it is interesting and perhaps more appropriate to search for a 4-digit approximate root and, consequently, a $1^{st}$ degree approximate common factor $v(s) = s + c$, which could also be considered here as the approximate $\varepsilon$-GCD of the set $\mathcal{P}$ for $\varepsilon = 10^{-4}$.

Using the results from the previous theoretical example 5.1 for the set $\mathcal{P}_{3,3}$, we tested and computed the strength bounds and the average strength of 150 approximations of the form $v(s) = s + c$ for the set $\mathcal{P}$. The constant $c$ ranged from 0.995 to 1.010 with increment 0.0001. The values of the strength bounds and the average strength of these approximations are presented in the graph of Figure 5.2.

This particular graph shows that when the constant $c$ increases, the average strength $\mathcal{S}^a(v)$ of $v(s)$ decreases, following a nearly parabolic line until it reaches a minimum at $c = 1.0022$. Then it increases, following again a nearly parabolic line in a symmetrical way. The results presented in Table 5.2 show that, for a specified numerical accuracy $\varepsilon = 10^{-4}$, the approximation

$$v(s) = s + 1.0022$$

has a minimum average strength equal to

$$\mathcal{S}^a(v) = 0.01821270609$$

and thus we may consider it as a "good" approximate $\varepsilon$-GCD of the set $\mathcal{P}$.

However, in Figure 5.2 we notice that the strength bounds $\underline{\mathcal{S}}(v)$ and $\overline{\mathcal{S}}(v)$ have

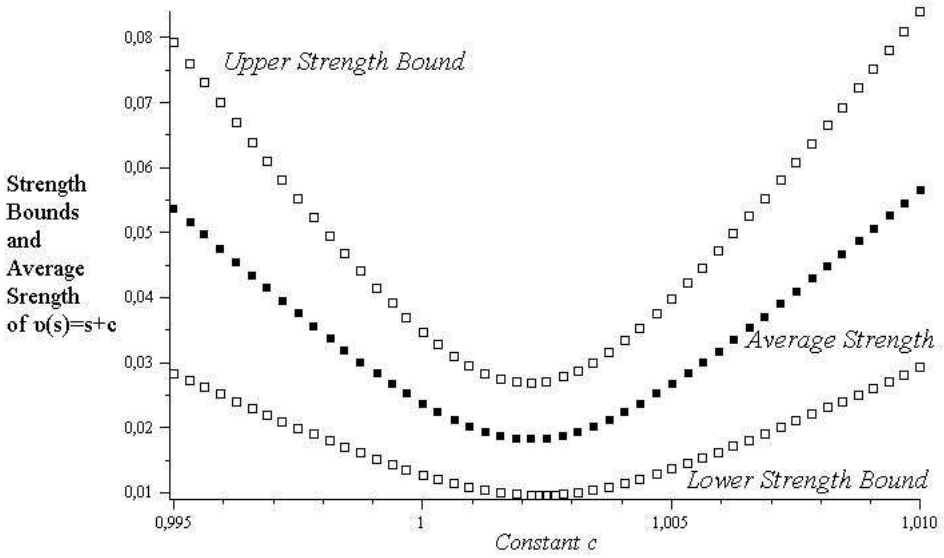


Figure 5.2: Measuring the strength of a $1^{st}$ degree approximate GCD of the set $\mathcal{P} \in \Pi(3, 2; 3)$ in Example 5.2.

similar graphs with $\mathcal{S}^a(v)$ and their minimum values are obtained for different approximations $v(s) = s + 1.0023$ and $v(s) = s + 1.0021$, respectively. These minimum values are highlighted in Table 5.2.

| $v(s)$ | $\underline{\mathcal{S}}(v)$ | $\overline{\mathcal{S}}(v)$ | $\mathcal{S}^a(v)$ | $Cond(\hat{\Phi}_v)$ |
|---|---|---|---|---|
| s+ 1.0019 | 0.009662968139 | 0.02696250452 | 0.01831273633 | 11.57988227 |
| s+ 1.0020 | 0.009615551663 | 0.02689041331 | 0.01825298248 | 11.57783592 |
| s+ 1.0021 | 0.009581612104 | **0.02685759025** | 0.01821960118 | 11.57579055 |
| s+ 1.0022 | 0.009561282419 | 0.02686412976 | **0.01821270609** | 11.57374615 |
| s+ 1.0023 | **0.009554638914** | 0.02690995372 | 0.01823229632 | 11.57170274 |
| s+ 1.0024 | 0.009561699564 | 0.02699481292 | 0.01827825624 | 11.56966031 |
| s+ 1.0025 | 0.009582423551 | 0.02711829199 | 0.01835035777 | 11.56761885 |

Table 5.2: The strength of the approximate GCD $v(s) = s + c$ of the set $\mathcal{P} \in \Pi(3, 2; 3)$ in Example 5.2.

Therefore, it is reasonable to search for approximations which have minimum strength bounds. According to (5.36), the upper strength bound for an arbitrary approximation $v(s) = s + c$, $c \in \mathbb{R} \setminus \{0\}$, will be given as a real function of $c$ such that

$$\overline{\mathcal{S}}_v(c) = \left(8 + \frac{5.9730}{c} - \frac{1.14092425}{c^2} - \frac{24.0895470}{c^3} + \frac{5.2586780}{c^4} - \frac{12}{c^5} + \frac{18}{c^6}\right)^{\frac{1}{2}} \quad (5.41)$$

We will attempt to minimize this function and see if it is possible to obtain

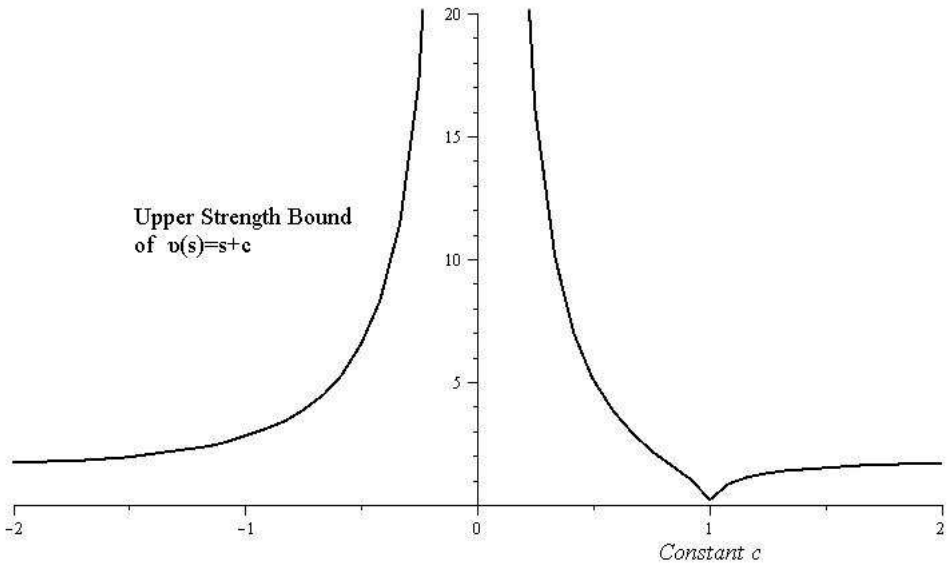


Figure 5.3: Graphical representation of the upper strength bound of $v(s)$ in Example 5.2.

an approximation with lower upper strength bound and without the restriction of 4-digit numerical precision. Using standard methods of calculus (first derivative test) and by observing the graph of the real function $\overline{\mathcal{S}}_v(c)$ in Figure 5.3, we can prove that $\overline{\mathcal{S}}_v(c)$ is minimized for $c_0 = 1.00213337932$.[1] Therefore, the upper strength bound $\overline{\mathcal{S}}(v)$ is minimized by the approximation:

$$v(s) = s + 1.00213337932$$

with minimum value:

$$\min_c \overline{\mathcal{S}}(v) = \overline{\mathcal{S}}_v(c_0) = 0.02685539677$$

Similarly, considering (5.37), the lower strength bound for an arbitrary approximation $v(s) = s + c$, $c \in \mathbb{R} \setminus \{0\}$, will be given as a real function of $c$ such that

$$\begin{aligned} \underline{\mathcal{S}}_v(c) \;=\; & \Big( \big(88344904 - 88119396\,c + 88501909\,c^2 - 104274792\,c^3 + 16658914\,c^4 - \\ & -72430188\,c^5 + 15436303\,c^6 + 23892000\,c^7 + 32000000\,c^8\big) \cdot \\ & \cdot \frac{2.5 \cdot 10^{-7}}{5\,c^8 + 4\,c^6 + 3\,c^4 + 2\,c^2 + 1} \Big)^{\frac{1}{2}} \end{aligned} \tag{5.42}$$

Using again standard methods of calculus (first derivative test) and by observing the graph of the real function $\underline{\mathcal{S}}_v(c)$ in Figure 5.4, we can prove that $\underline{\mathcal{S}}_v(c)$ is minimized for $c_0' = 1.002298464659$.

[1] The results are provided from the built-in routine of Maple `minimize`.

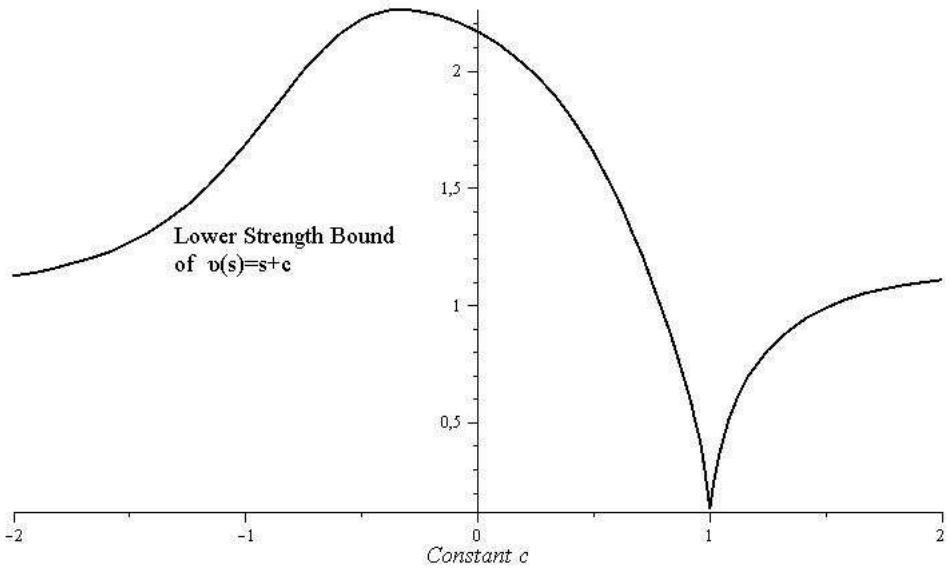


Figure 5.4: Graphical representation of the lower strength bound of $v(s)$ in Example 5.2.

Therefore, the lower strength bound $\underline{\mathcal{S}}(v)$ is minimized by the approximation:

$$v(s) = s + 1.002298464659$$

with minimum value:

$$\min_{c} \underline{\mathcal{S}}(v) = \overline{\mathcal{S}}_v(c_0') = 0.009554637298$$

Finally, we conclude that the "best" approximate GCD of the form $v(s) = s + c$ for the polynomial set $\mathcal{P}$ is expected to have a strength $\mathcal{S}(v)$ :

$$0.009554637298 \leq \mathcal{S}(v) \leq 0.02685539677 \tag{5.43}$$

Nevertheless, it remains important to solve the extended minimisation problem:

$$\min_{c} \mathcal{S}(v) = \min_{c,\mathcal{P}^*} \left\| S_{\mathcal{P}} - \left[ \mathbb{O}_{m,r} | \tilde{S}^{(r)}_{\mathcal{P}^*} \right] \cdot \hat{\Phi}_v \right\|_F \tag{5.44}$$

which derives from the original problem (5.16) for $v(s) = s + c$, $c \neq 0$ and $r = 1$. Then, it is required to study the following objective function:

$$\mathcal{S}_v(c, a_{i,j}) = \left\| S_{\mathcal{P}} - \left[ \mathbb{O}_{m,r} | \tilde{S}^{(r)}_{\mathcal{P}^*} \right] \cdot \hat{\Phi}_v \right\|^2_F \tag{5.45}$$

The matrix $\left[ \mathbb{O}_{m,r} | \tilde{S}^{(r)}_{\mathcal{P}^*} \right]$ with arbitrary parameters has the form:

$$\left[\mathbb{O}_{m,r}|\tilde{S}^{(r)}_{\mathcal{P}^*}\right] = \left[\begin{array}{c|cccc} 0 & a_{1,2} & a_{1,3} & a_{1,4} & 0 \\ 0 & 0 & a_{1,2} & a_{1,3} & a_{1,4} \\ 0 & a_{2,2} & a_{2,3} & 0 & 0 \\ 0 & 0 & a_{2,2} & a_{2,3} & 0 \\ 0 & 0 & 0 & a_{2,2} & a_{2,3} \\ 0 & a_{3,2} & a_{3,3} & 0 & 0 \\ 0 & 0 & a_{3,2} & a_{3,3} & 0 \\ 0 & 0 & 0 & a_{3,2} & a_{3,3} \end{array}\right] \tag{5.46}$$

Then, the objective function $\mathcal{S}_v(c, a_{i,j})$ in (5.45) has the form:

$$\begin{aligned} \mathcal{S}_v(c, a_{i,j}) &= 2\,(1 - a_{1,2})^2 + 2\,(3 - a_{1,2}c - a_{1,3})^2 - 2\,(1 + a_{1,3}c + a_{1,4})^2 \\ &\quad -2\,(3 + a_{1,4}c)^2 + 3\,(1 - a_{3,2})^2 - 3\,(1.995 + a_{3,2}c + a_{3,3})^2 \\ &\quad -3\,(3.015 + a_{3,3}c)^2 + 3\,(1 - a_{2,2})^2 - 3\,(1.0005 + a_{2,2}c + a_{2,3})^2 \\ &\quad -3\,(1.9990 + a_{2,3}c)^2 \end{aligned} \tag{5.47}$$

The minimisation of the above function $\mathcal{S}_v(c, a_{i,j})$ is basically a non-linear least-squares problem. Furthermore, it is necessary here to compute the global minimum of this function rather than a local minimum. For the purpose of this study, we used the built-in `Optimization[Minimize]` routine of Maple, which is based on the Gauss-Newton and modified Newton iterative methods [25]. Unfortunately, the results showed that the minimum value of $\mathcal{S}_v(c, a_{i,j})$ is 6.115444877153 for $c = -2.703773531865$. This result is not acceptable and obviously we deal with an ill-conditioned optimisation problem.

However, if we apply scaling such as $v(s) = \dfrac{s}{c} + 1$ and set $w := \dfrac{1}{c}$, then the objective function will be given by

$$\begin{aligned} \mathcal{S}_v(w, a_{i,j}) &= 2\,(1 - a_{1,2}w)^2 + 2\,(3 - a_{1,2} - a_{1,3}w)^2 - 2\,(1 + a_{1,3} + a_{1,4}w)^2 \\ &\quad -2\,(3 + a_{1,4})^2 + 3\,(1 - a_{3,2}w)^2 - 3\,(1.995 + a_{3,2} + a_{3,3}w)^2 \\ &\quad -3\,(3.015 + a_{3,3})^2 + 3\,(1 - a_{2,2}w)^2 - 3\,(1.0005 + a_{2,2} + a_{2,3}w)^2 \\ &\quad -3\,(1.9990 + a_{2,3})^2 \end{aligned} \tag{5.48}$$

Using again the `Optimization[Minimize]` routine of Maple, we will notice that the function $\mathcal{S}_v(w, a_{i,j})$ has a minimum value equal to $2.2860982654 \cdot 10^{-4}$ at $w_0 = 0.99770975355$. This immediately shows that the solution of the problem (5.44) is $\min_c \mathcal{S}(v) = \sqrt{2.2860982654 \cdot 10^{-4}} = 0.0151198487605$, which is obtained for $c = \frac{1}{w_0} = 1.0022955037196$.

Unless we have not found the global minimum of the minimisation problem (5.44), the "best" approximate CCD of the set $\mathcal{P}$ is

$$v(s) = s + 1.0022955037196 \tag{5.49}$$

with strength

$$\mathcal{S}(v) = 0.0151198487605 \tag{5.50}$$

and, of course, the inequality (5.43) is satisfied. □

**Example 5.3.** Consider a larger set of polynomials $\mathcal{P} \in \Pi\{5, 4; 12\}$ with numeric floating-point coefficients:

$$\mathcal{P} = \left\{ \begin{array}{rcl} a(s) & = & s^5 + 4.60010\,s^4 + 5.90026\,s^3 + 2.60007\,s^2 + 3.30012\,s + 1.80009 \\ b_1(s) & = & -20\,s^4 - 77.00400\,s^3 - 54.00740\,s^2 + 61.00400\,s + 42.00420 \\ b_2(s) & = & -10\,s^4 - 51.00300\,s^3 - 117.00930\,s^2 - 134.01650\,s - 48.00720 \\ b_3(s) & = & 40\,s^4 + 109.00400\,s^3 + 106.00290\,s^2 + 123.00480\,s + 54.00270 \\ b_4(s) & = & 20\,s^4 + 92.00400\,s^3 + 153.01040\,s^2 + 113.00980\,s + 30.00300 \\ b_5(s) & = & -10\,s^4 - 26.00300\,s^3 + 7.99820\,s^2 + 52.00600\,s + 24.00360 \\ b_6(s) & = & 10\,s^4 + 21.00100\,s^3 + 14.00010\,s^2 + 33.00120\,s + 18.00090 \\ b_7(s) & = & 30\,s^4 + 33.00600\,s^3 - 101.00540\,s^2 - 106.00940\,s - 24.00240 \\ b_8(s) & = & 15\,s^4 + 59.00450\,s^3 + 55.00870\,s^2 - 15.00090\,s - 18.00270 \\ b_9(s) & = & -15\,s^4 - 74.00150\,s^3 - 89.00440\,s^2 + 9.99990\,s + 24.00120 \\ b_{10}(s) & = & 25\,s^4 + 90.00500\,s^3 + 140.00800\,s^2 + 147.01200\,s + 54.00540 \\ b_{11}(s) & = & 10\,s^4 + 71.00300\,s^3 + 119.01530\,s^2 + 28.00510\,s - 12.00180 \end{array} \right\} \tag{5.51}$$

We search for an approximate GCD of the above set $\mathcal{P}$ and we will use the Hybrid ERES algorithm in order to find one. For the typical 16-digits numerical precision and tolerances $\varepsilon_t = 10^{-15}$, $\varepsilon_G = 10^{-15}$, the given solution is

$$\gcd\{\mathcal{P}\}: \quad g(s) = s + 0.6 \tag{5.52}$$

with average strength $\mathcal{S}^a(g) = 4.467947303 \cdot 10^{-28}$. This solution can be characterised as an exact GCD or an excellent approximate GCD of the set $\mathcal{P}$, because the value of the average strength is practically zero for the 16-digits accuracy.

The Hybrid ERES algorithm does not provide any other approximate solutions for the given set $\mathcal{P}$. However, if we compute the average strength of a simple polynomial factor of the general form:

$$v(s) = s + c, \; c \in \mathbb{R} \setminus \{0\}$$

for the set $\mathcal{P}$ and by using the Frobenius norm, then we actually obtain the average strength $\mathcal{S}^a(v)$ as a real function of $c$, denoted by $\mathcal{S}^a_v(c)$, with its graph presented in Figure 5.5.

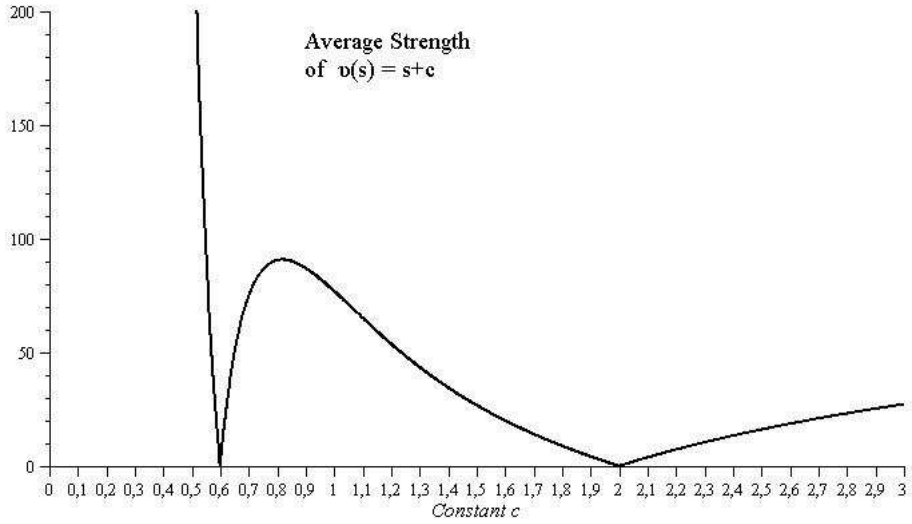


Figure 5.5: Graphical representation of the average strength $\mathcal{S}_v^a(c)$ of the common factor $v(s) = s + c$ of the set $\mathcal{P}$ in Example 5.3.

If we carefully observe the graph of the average strength $\mathcal{S}_v^a(c)$ in Figure 5.5, we will notice that there are two local minima. The first local minimum is at $c = 0.6$, which is very normal, since it verifies the already computed GCD in (5.52). However, there is another local minimum near $c = 2$ which reveals the presence of another simple common factor with weaker strength. We may compute this minimum by using the minimisation routine `Optimization[Minimize]` of Maple with objective function $\mathcal{S}_v^a(c)$ and initial point $\hat{c} = 2.1$ (or $\hat{c} = 1.9$). Then, we get

$$\min_c \mathcal{S}^a(v) = 0.002342396475056914 \tag{5.53}$$

for $c = 2.00015927122308631$. Therefore, there exists another approximate common factor of the form:

$$\hat{v}(s) = s + 2.00015927122308631 \tag{5.54}$$

with average strength $\mathcal{S}^a(\hat{v}) = 2.342396475056914 \cdot 10^{-3}$. Obviously, $\hat{v}(s)$ is a "weaker" common factor of $\mathcal{P}$ in comparison with $g(s)$, but we may infer that there is an approximate GCD of degree 2, given by the product of these common factors. Thus, we can have an approximate GCD of the form:

$$\begin{aligned} \hat{g}(s) &= \hat{v}(s) \cdot g(s) = \\ &= s^2 + 2.60015927122308631\, s + 1.20009556273385179 \end{aligned} \tag{5.55}$$

with average strength $\mathcal{S}^a(\hat{g}) = 2.951102916 \cdot 10^{-3}$. However, if we evaluate the

strength of $\hat{g}(s)$ by solving the problem:

$$\mathcal{S}(\hat{g}) = \min_{\mathcal{P}^*} \left\| S_{\mathcal{P}} - \left[ \mathbb{O}_{m,2} | \tilde{S}^{(2)}_{\mathcal{P}^*} \right] \cdot \hat{\Phi}_{\hat{g}} \right\|_F \tag{5.56}$$

we get $\mathcal{S}(\hat{g}) = 3.2297259946 \cdot 10^{-3}$.

Surprisingly, if we solve the extended minimisation problem (5.44) for $v(s) = s^2 + c_1\, s + c_2$ with $\deg\{v(s)\} = r = 2$, where $c_1$, $c_2$ are nonzero constants in $\mathbb{R}$, we get

$$\min_{c_1, c_2} \mathcal{S}(v) = 0.00322861972251372628$$

for $c_1 = 2.60015931739949568$ and $c_2 = 1.20009515827349452$. This implies that there exists a $2^{nd}$ degree approximate GCD of the form:

$$\tilde{g}(s) = s^2 + 2.600159317399495681\, s + 1.20009515827349452 \tag{5.57}$$

with strength $\mathcal{S}(\tilde{g}) = 3.2286197225 \cdot 10^{-3}$, which is nearly identical with $\hat{g}(s)$ in (5.55).

Furthermore, if we solve again the extended minimisation problem (5.44) for a simple common factor $v(s) = s + c_1$, $r = 1$, where $c_1$ is a nonzero constant in $\mathbb{R}$, and initial point $\hat{c} = 2.1$, we get:

$$\min_{c_1} \mathcal{S}(v) = 0.00305624668250932152 \tag{5.58}$$

for $c_1 = 2.00015926974934885$. Therefore, we have computed again a $1^{st}$ degree approximate common factor of the form:

$$\tilde{v}(s) = s + 2.00015926974934885 \tag{5.59}$$

with strength $\mathcal{S}(\tilde{v}) = 3.0562466825 \cdot 10^{-3}$, which is nearly identical with $\hat{v}(s)$ in (5.54).

Conclusively, we used here the strength minimisation problem $\min_c \mathcal{S}(v)$ and the average strength minimisation problem $\min_c \mathcal{S}^a(v)$ in order to compute a simple approximate common factor $v(s) = s + c$ for a set $\mathcal{P}$ of 12 real polynomials of maximum degree 5. The given results in (5.54) and (5.59) are nearly identical. But the major difference is that the former problem involves 50 arbitrary parameters (including $c$) to be solved, whereas the latter involves only one arbitrary parameter, $c$, which results in significantly less computational complexity and improved efficiency. □

## 5.6 Computational results

In this section we shall present a number of results given by the previous algorithms for different sets of several polynomials. More specifically, we use the ERES algorithm to find the $\varepsilon_t$-GCD of a set of polynomials $\mathcal{P}_{h+1,n}$ for $\varepsilon_t$ accuracy and we evaluate the given GCD approximation $v$ by computing the strength bounds $\underline{\mathcal{S}}(v)$ and $\overline{\mathcal{S}}(v)$ and, of course, its strength $\mathcal{S}(v)$. Our intention is to show the advantages of the hybrid implementation of the ERES algorithm in contrast with its fully numerical implementation. Thus, we shall denote by *H-ERES* the Hybrid ERES algorithm as described in Chapter 4 and presented in Figure 4.1, and *N-ERES* the fully numerical ERES algorithm as presented in [57, 58]. The algorithms were implemented in the software programming environment of Maple using both rational and variable floating-point arithmetic. For the computation of the GCD in Example 5.5, the software numerical accuracy of Maple was set to 16 digits in order to simulate the common hardware accuracy. Conversely, for the computation of the strength bounds and the strength of the given approximation, the software numerical accuracy was set to 64 digits for more reliable results. The strength of the given approximation was computed in Maple by using the built-in numerical minimization routine `Optimization[Minimize]`.

**Example 5.4.** Consider the following set $\mathcal{P}_{3,2} \in \Pi(2,2;3)$, $h = n = p = 2$ with polynomials:

$$\begin{aligned} a(s) &= 2.0\,s^2 + 2.380952380952381\,s - 0.3809523809523810 \\ b_1(s) &= s^2 - 3.642857142857143\,s + 0.5 \\ b_2(s) &= 1.5\,s^2 - 7.214285714285714\,s + 1.0 \end{aligned}$$

The coefficients of the set $\mathcal{P}_{3,2}$ are given in 16-digits floating-point format. The basis matrix of the set has the form:

$$P_3 = \begin{bmatrix} -0.3809523809523810 & 2.380952380952381 & 2.0 \\ 0.5 & -3.642857142857143 & 1.0 \\ 1.0 & -7.214285714285714 & 1.5 \end{bmatrix}$$

The data of $P_3$ are converted to an approximate[2] rational format and the matrix has the following form:

$$P'_3 = \begin{bmatrix} -\frac{8}{21} & \frac{50}{21} & 2 \\ \frac{1}{2} & -\frac{51}{14} & 1 \\ 1 & -\frac{101}{14} & \frac{3}{2} \end{bmatrix}$$

[2]The conversion is done according to Maple's arithmetic by using the directive `convert`.

The error from this conversion is

$$\|P'_3 - P_3\|_\infty = 2.8571428571428571 \cdot 10^{-16}$$

This is our initial error in 16-digits floating-point arithmetic, where the respective machine's epsilon is $\varepsilon_m = 2.22044604925031308 \cdot 10^{-16}$. This error does not grow during the iterations of the algorithm.

The final matrix $P_3^{(N)}$ has rank 1 for a selected tolerance $\varepsilon_t = 10^{-15}$ and the respective right singular vector is

$$\underline{w}^t = [-0.141421356237309, 0.9899494936611661]^t$$

The solution is given either from the rows of the final matrix of the main iterative procedure:

$$v(s) = s - \frac{1}{7}$$

or from the vector $\underline{w}$ :

$$v'(s) = s - 0.1428571428571427$$

The polynomial $v'(s)$ is an $\varepsilon_t$-GCD and the distance between these two solutions is

$$\|\underline{v} - \underline{v}'\|_\infty \approx 1.57 \cdot 10^{-16}$$

Now, we shall evaluate the strength bounds of the polynomial $v'(s)$ by using the Algorithm 5.1.

- Computation of the Frobenius norms $\|\hat{S}'_{\mathcal{P}}\|$, $\|\hat{\Phi}_v^{-1}\|$ and $\|\hat{\Phi}_v\|$ :

$$\begin{aligned} \|\hat{S}'_{\mathcal{P}}\| &= 4.6128999736247051 \cdot 10^{-13} \\ \|\hat{\Phi}_{v'}\| &= 12.288205727444521 \\ \|\hat{\Phi}_{v'}^{-1}\| &= 350.14568396597551 \\ Cond(\hat{\Phi}_{v'}) &= 4302.6621991506793 \end{aligned}$$

- Computation of the strength bounds:

$$\begin{aligned} \underline{\mathcal{S}}(v') = \frac{\|\hat{S}'_{\mathcal{P}}\|}{\|\hat{\Phi}_{v'}^{-1}\|} &= 1.317423056990459 \cdot 10^{-15} \\ \overline{\mathcal{S}}(v') = \|\hat{S}'_{\mathcal{P}} \cdot \hat{\Phi}_{v'}\| &= 9.134903149763325 \cdot 10^{-14} \end{aligned}$$

The strength of the polynomial $v'(s) = s - 0.1428571428571424$ is evaluated

by the minimization problem (5.16) :

$$f(\mathcal{P},\mathcal{P}^*)=\min_{\forall\mathcal{P}^*}\left\|S_{\mathcal{P}}-\left[\mathbb{O}_{m,r}|\tilde{S}^{(r)}_{\mathcal{P}^*}\right]\cdot\hat{\Phi}_{v'}\right\|_F$$

where

$$\left[\mathbb{O}_{m,r}|\tilde{S}^{(r)}_{\mathcal{P}^*}\right]=\left[\begin{array}{c|ccc}0&a_1&a_2&0\\0&0&a_1&a_2\\0&a_3&a_4&0\\0&0&a_3&a_4\\0&a_5&a_6&0\\0&0&a_5&a_6\end{array}\right]$$

and $a_i,\ i=1,\ldots,6$ are arbitrary parameters. The built-in minimization routine of Maple gives the next result:

$$\mathcal{S}(v')=f(\mathcal{P},\mathcal{P}^*)=1.8451526088542018\cdot10^{-15}$$

for

$$\{a_1,a_2,a_3,a_4,a_5,a_6\}=\{2.0,2.666666666666668,1.0,-3.5,1.5,-7.0\}$$

in 16-digits software accuracy. Obviously, the computed value of the strength is within the values of its bounds and extremely close to them. □

**Example 5.5.** We consider the set $\mathcal{P}_{3,11}\in\Pi(11,11;3)$, $h=2,\ n=p=11$ with polynomials :

$$\begin{aligned}a(s)&=-16.316\,s^{11}+182.73\,s^{10}-185.83\,s^9+106.68\,s^8\\&\quad-266.22\,s^7+125.80\,s^6-195.53\,s^5+243.81\,s^4\\&\quad+23.013\,s^3+64.186\,s^2-24.300\,s-43.810\\b_1(s)&=4.6618\,s^{11}-52.209\,s^{10}+53.094\,s^9-30.481\,s^8\\&\quad+76.064\,s^7-35.944\,s^6+55.866\,s^5-69.659\,s^4\\&\quad-6.5751\,s^3-18.339\,s^2+6.9428\,s+12.517\\b_2(s)&=-4.1155\,s^{11}+47.507\,s^{10}-59.034\,s^9+2.2157\,s^8\\&\quad-45.276\,s^7+83.932\,s^6-34.013\,s^5+15.007\,s^4\\&\quad+4.3083\,s^3-9.0031\,s^2+14.297\,s-14.783\end{aligned}$$

The exact GCD of the set $\mathcal{P}_{3,11}$ is $g(s)=1$ and the tolerance $\varepsilon_G$ is set equal to the machine's epsilon in 16-digits accuracy $\varepsilon_m\approx2.2204\cdot10^{-16}$. The H-ERES algorithm gave us five possible approximate solutions for different values of the tolerance $\varepsilon_t$. In Table 5.3 we denote these solutions by $v_i(s),\ i=1,2,\ldots,5$.

| H-ERES | $v_1(s)$ | $v_2(s)$ | $v_3(s)$ | $v_4(s)$ | $v_5(s)$ |
|---|---|---|---|---|---|
| Degree | 1 | 8 | 4 | 5 | 2 |
| $\varepsilon_t$ | $9.5 \cdot 10^{-2}$ | $6 \cdot 10^{-2}$ | $3 \cdot 10^{-3}$ | $6 \cdot 10^{-4}$ | $5 \cdot 10^{-5}$ |
| $\mathcal{S}(v)$ | 14.1684562 | 144.132727 | 0.72610271 | 4.00725481 | 0.02571431 |
| $\underline{\mathcal{S}}(v)$ | 3.07315597 | 0.53653745 | 0.22191334 | 0.06286887 | 0.00319468 |
| $\overline{\mathcal{S}}(v)$ | 69.9370280 | 3563.87058 | 930.143927 | 322.951121 | 0.19880416 |
| $Cond(\hat{\Phi}_v)$ | 22.7573962 | 6642.35195 | 4191.47371 | 5136.90032 | 62.2296806 |
| Main Iter. | 20 | 6 | 14 | 12 | 18 |
| Rank-1 Iter. | 11 | 4 | 8 | 7 | 10 |

Table 5.3: Results for the $\varepsilon_t$-GCD of the set $\mathcal{P}_{3,11}$ in Example 5.5.

From the presented results in Table 5.3 we can conclude that:

i) The "best" approximate solution is the polynomial

$$v_5(s) = s^2 - 11.28371806974011\, s + 11.64469379842480$$

which has the lowest strength $\mathcal{S}(v) = 0.02571431$ and it is given for $\varepsilon_t = 5 \cdot 10^{-5}$.

ii) The strength $\mathcal{S}(v)$ is well bounded when $Cond(\hat{\Phi}_v) < (h+1)(n+p) = 66$.

iii) The number of iterations of the main procedure is greater than the respective number of iterations of the rank-1 procedure.

The N-ERES algorithm gave approximately the same results with a restriction to 8 digits of accuracy and $\varepsilon_G > 10^{-6}$. □

**► Comparison of the Hybrid ERES algorithm with other GCD algorithms**

We tested and compared the algorithms H-ERES and N-ERES for various random sets of polynomials and some representative results are presented in Table 5.4. More specifically, we examined random sets of polynomials with integer coefficients between $-10^7$ and $10^7$, which have an exact GCD with rational coefficients. The N-ERES algorithm uses the data as floating-point numbers and in every polynomial set $\mathcal{P}_{h+1,n}$ in Table 5.4 the selected software floating-point accuracy of the system, denoted as *Dig*, is selected as the minimum accuracy that N-ERES requires to produce a polynomial, which has at least the same degree as the respective exact GCD. The H-ERES algorithm uses the data as symbolic-rational numbers and produces the GCD of the set accurately. However, we can also have a numerical solution given by the Rank-1 procedure (PSVD1 algorithm) according to Proposition 4.1. For the numerical part of the H-ERES algorithm, the software

floating-point accuracy remained the same with the selected accuracy for the N-ERES algorithm in order to compare the results. The internal accuracies of the algorithm are $\varepsilon_t = \varepsilon_G \approx 10^{-Digits}$.

We compared the two algorithms in respect of the numerical relative error between the exact GCD and the given solution, the number of main iterations and the required time of execution. The results in Table 5.4 show that the H-ERES algorithm produces numerical solutions of better quality than the N-ERES algorithm.

The relative error is given by $Rel = \frac{\|\underline{v}-\underline{g}\|_2}{\|\underline{g}\|_2}$, where $\underline{v}$, $\underline{g}$ are the coefficient vectors of the provided solution $v(s)$ and the exact GCD $g(s)$ respectively. $\|\cdot\|_2$ denotes the Euclidean norm. In the case of large sets of polynomials, the N-ERES fails to produce accurate results in the standard floating-point precision of 16-digits of accuracy. Conversely, the H-ERES algorithm works with this accuracy and gives very good results, which are presented in Table 5.5 [3].

Furthermore, a comparison of the Hybrid ERES method with other existing matrix-based methods developed for the computation of the GCD of a set of several polynomials by processing all the polynomials of the set *simultaneously*, has been made [87, 91] and the results are given in Tables 5.6 and 5.7. This comparison includes the standard matrix pencil (MP) [45], the modified resultant matrix pencil (MRMP) [72, 73], the resultant extended-row-equivalence (RERE) [72, 73], the modified resultant extended-row-equivalence (MRERE) [72, 73], and the subspace (SS) [64] method. A description of the MP, MRMP and SS methods has been given in Chapter 2. The RERE and MRERE methods are based on the application of elementary row transformations to a Sylvester-type matrix (resultants), formed from the coefficients of the polynomials, using either QR or LU factorisation [18]. The results show that the Hybrid ERES algorithm combines speed and accuracy and thus it has better overall performance compared with the other algorithms, especially in the case of large sets of polynomials.

## 5.7 Discussion

The notion of the approximate GCD of sets of several polynomials and the related developed framework for the evaluation of its quality was considered. The results presented in [21, 22, 42] provided the means to define the strength of a given approximation as a solution of an appropriate optimisation problem and thus reduced the approximate GCD problem to an equivalent distance problem. This allows the evaluation of the quality of an approximation and it is totally independent from any GCD computational method. Unfortunately, such an optimization problem is non-convex and therefore it requires special methods

[3] The sets of polynomials in Tables 5.4 and 5.5 are the same.

to be solved. However, by exploiting the structural properties of the resultant matrices, the algebraic analysis of the objective function of the derived optimisation problem led to the establishment of an upper and lower numerical bound for the strength of a given approximate GCD. These numerical bounds can easily be computed via a simple algorithm and act as indicators of the strength of an approximation. Depending on the numerical condition of the problem these indicators can give us reliable information about the quality of a given approximate GCD without the need to compute the actual strength by solving the related minimisation problem.

In this context, the Hybrid ERES algorithm was tested regarding its effectiveness in computing meaningful and acceptable approximate solutions when the input polynomials suffer from numerical inaccuracies, or if they are considered to be non-coprime within a numerical accuracy $\varepsilon_t$. The produced results have showed that the Hybrid ERES algorithm has the remarkable advantage to produce multiple approximate GCDs within a specified numerical accuracy. These approximations can be considered of good quality, if their strength bounds and consequently their actual strength are close enough to zero. The approach however, provides the means for defining the explicit form of a reduced optimisation problem for the computation of the strength, based only on the free parameters (i.e. the coefficients) of the approximate GCD. This reduction will simplify a lot the optimization process and will put the study of computing the *optimal approximate GCD* into new perspective.

| Set | $h, n, p, d$ | Dig. | Alg. | Main | SVD | Time | Rel. Err. |
|---|---|---|---|---|---|---|---|
| $\mathcal{P}_{11,10}$ | $10, 10, 10, 1$ | 16 | N-ERES | 5 | 2 | 0.203 | $5.28309 \cdot 10^{-16}$ |
| | | | H-ERES | 3 | 3 | 0.266 | $6.44538 \cdot 10^{-16}$ |
| $\mathcal{P}_{21,20}$ | $20, 20, 20, 2$ | 55 | N-ERES | 5 | 2 | 0.688 | $1.02570 \cdot 10^{-19}$ |
| | | | H-ERES | 3 | 3 | 0.797 | $1.00764 \cdot 10^{-53}$ |
| $\mathcal{P}_{31,30}$ | $30, 30, 30, 3$ | 34 | N-ERES | 6 | 2 | 1.749 | $3.38425 \cdot 10^{-20}$ |
| | | | H-ERES | 2 | 2 | 2.156 | $2.53046 \cdot 10^{-33}$ |
| $\mathcal{P}_{31,40}$ | $30, 40, 40, 4$ | 45 | N-ERES | 10 | 2 | 3.375 | $3.45159 \cdot 10^{-21}$ |
| | | | H-ERES | 4 | 3 | 14.250 | $1.14197 \cdot 10^{-44}$ |
| $\mathcal{P}_{51,30}$ | $50, 30, 30, 5$ | 58 | N-ERES | 2 | 2 | 3.812 | $1.27734 \cdot 10^{-19}$ |
| | | | H-ERES | 3 | 3 | 3.703 | $4.14280 \cdot 10^{-56}$ |

Table 5.4: Results from H-ERES and N-ERES algorithms for randomly selected sets of polynomials.

| Set | $h, n, p, d$ | Dig. | Main | PSVD1 | Time | Rel. Err. |
|---|---|---|---|---|---|---|
| $\mathcal{P}_{11,10}$ | $10, 10, 10, 1$ | 16 | 3 | 3 | 0.266 | $6.44538 \cdot 10^{-16}$ |
| $\mathcal{P}_{21,20}$ | $20, 20, 20, 2$ | 16 | 3 | 3 | 0.704 | $2.65026 \cdot 10^{-16}$ |
| $\mathcal{P}_{31,30}$ | $30, 30, 30, 3$ | 16 | 3 | 3 | 2.171 | $4.78899 \cdot 10^{-16}$ |
| $\mathcal{P}_{31,40}$ | $30, 40, 40, 4$ | 16 | 5 | 4 | 14.156 | $1.65847 \cdot 10^{-16}$ |
| $\mathcal{P}_{51,30}$ | $50, 30, 30, 5$ | 16 | 3 | 3 | 3.094 | $5.44165 \cdot 10^{-16}$ |

Table 5.5: Results from the H-ERES algorithm in 16 digits of accuracy.

Notation in Tables 5.4 and 5.5 :

| | | |
|---|---|---|
| Set | : | Set $\mathcal{P}_{h+1,n}$ of random polynomials. |
| Main | : | Number of main iterations. |
| h,n,p | : | Parameters of the set as defined in (3.1). |
| SVD | : | Number of SVD or PSVD1 calls. |
| d | : | Degree of the exact GCD of the set $\mathcal{P}_{h+1,n}$. |
| Time | : | Time of execution in seconds. |
| Dig. | : | Number of digits of software accuracy. |
| Rel. Err. | : | Relative error. |
| Alg. | : | Type of algorithm. |

| | | ERES | RERE (LU) | MRERE (LU) | RERE (QR) | MRERE (QR) | MRMP | MP | SS |
|---|---|---|---|---|---|---|---|---|---|
| | | Hybrid | Numerical | Numerical | Numerical | Numerical | Hybrid | Hybrid | Numerical |
| $m = 10$ | Rel | $3.31 \cdot 10^{-31}$ | $3.82 \cdot 10^{-30}$ | $2.94 \cdot 10^{-31}$ | $1.35 \cdot 10^{-29}$ | $3.36 \cdot 10^{-30}$ | $2.08 \cdot 10^{-31}$ | $2.49 \cdot 10^{-31}$ | $2.15 \cdot 10^{-31}$ |
| $n = 5$ | Strength | $1.25 \cdot 10^{-27}$ | $6.11 \cdot 10^{-26}$ | $7.70 \cdot 10^{-27}$ | $3.30 \cdot 10^{-25}$ | $5.56 \cdot 10^{-26}$ | $2.60 \cdot 10^{-26}$ | $3.28 \cdot 10^{-26}$ | $8.95 \cdot 10^{-27}$ |
| $p = 5$ | Time | 0.094 | 0.016 | 0.016 | 0.031 | 0.015 | 0.125 | 0.063 | 0.203 |
| $d = 2$ | Flops | 471 | 2584 | 1447 | 5167 | 2097 | 9167 | 2480 | 5445000 |
| $m = 10$ | Rel | $1.30 \cdot 10^{-31}$ | $4.56 \cdot 10^{-26}$ | $2.01 \cdot 10^{-28}$ | $5.56 \cdot 10^{-27}$ | $3.80 \cdot 10^{-26}$ | $1.82 \cdot 10^{-26}$ | $1.53 \cdot 10^{-27}$ | $6.12 \cdot 10^{-31}$ |
| $n = 10$ | Strength | $4.26 \cdot 10^{-27}$ | $1.49 \cdot 10^{-23}$ | $6.42 \cdot 10^{-24}$ | $1.16 \cdot 10^{-21}$ | $1.44 \cdot 10^{-21}$ | $1.66 \cdot 10^{-21}$ | $1.01 \cdot 10^{-22}$ | $4.37 \cdot 10^{-26}$ |
| $p = 10$ | Time | 0.141 | 0.047 | 0.031 | 0.172 | 0.063 | 0.937 | 0.156 | 1.0 |
| $d = 2$ | Flops | 900 | 20667 | 11894 | 41334 | 20603 | 73334 | 2480 | $1.2 \cdot 10^{9}$ |
| $m = 10$ | Rel | $3.82 \cdot 10^{-32}$ | $2.43 \cdot 10^{-28}$ | $1.78 \cdot 10^{-29}$ | $1.98 \cdot 10^{-23}$ | $5.11 \cdot 10^{-23}$ | $3.16 \cdot 10^{-24}$ | $2.14 \cdot 10^{-25}$ | $2.02 \cdot 10^{-31}$ |
| $n = 15$ | Strength | $3.03 \cdot 10^{-26}$ | $1.99 \cdot 10^{-23}$ | $9.96 \cdot 10^{-25}$ | $1.47 \cdot 10^{-18}$ | $3.34 \cdot 10^{-18}$ | $5.59 \cdot 10^{-19}$ | $1.85 \cdot 10^{-20}$ | $7.07 \cdot 10^{-26}$ |
| $p = 10$ | Time | 0.110 | 0.094 | 0.032 | 0.375 | 0.172 | 1.516 | 0.203 | 2.375 |
| $d = 3$ | Flops | 1476 | 40896 | 19355 | 81792 | 30334 | 126720 | 4575 | $5.7 \cdot 10^{9}$ |

m = number of polynomials, n = the maximum degree as in (3.1), p = the second maximum degree as in (3.1), d = the degree of the GCD

Table 5.6: Comparison of GCD algorithms with randomly selected sets of 10 polynomials: $\varepsilon_t = 10^{-16}$, Dig = 32

| | | ERES | RERE (LU) | MRERE (LU) | RERE (QR) | MRERE (QR) | MRMP | MP | SS |
|---|---|---|---|---|---|---|---|---|---|
| | | Hybrid | Numerical | Numerical | Numerical | Numerical | Hybrid | Hybrid | Numerical |
| $m = 15$ | Rel | $4.37 \cdot 10^{-32}$ | 0.927 | 0.928 | 0.927 | 0.928 | $3.84 \cdot 10^{-27}$ | $1.51 \cdot 10^{-18}$ | $1.01 \cdot 10^{-30}$ |
| $n = 25$ | Strength | $6.10 \cdot 10^{-17}$ | $4.53 \cdot 10^{-10}$ | $1.71 \cdot 10^{-9}$ | $2.93 \cdot 10^{-9}$ | $3.79 \cdot 10^{-10}$ | $5.026 \cdot 10^{-12}$ | $3.36 \cdot 10^{-12}$ | $1.48 \cdot 10^{-16}$ |
| $p = 25$ | Time | 5.203 | 1.328 | 0.562 | 2.719 | 1.578 | 18.048 | 4.078 | 18.063 |
| $d = 5$ | Flops | 3993 | 479167 | 204917 | 958334 | 372389 | 1145834 | 73684 | $3.1 \cdot 10^{12}$ |
| $m = 50$ | Rel | $4.31 \cdot 10^{-31}$ | $0.14 \cdot 10^{-26}$ | $0.96 \cdot 10^{-30}$ | $0.13 \cdot 10^{-28}$ | $0.29 \cdot 10^{-28}$ | $0.95 \cdot 10^{-28}$ | $0.23 \cdot 10^{-28}$ | × |
| $n = 40$ | Strength | $1.51 \cdot 10^{-27}$ | $0.13 \cdot 10^{-26}$ | $0.16 \cdot 10^{-26}$ | $0.41 \cdot 10^{-26}$ | $0.42 \cdot 10^{-27}$ | $0.55 \cdot 10^{-23}$ | $0.33 \cdot 10^{-25}$ | × |
| $p = 40$ | Time | 4.390 | 39.81 | 2.75 | 190.19 | 11.64 | 32.92 | 12.62 | × |
| $d = 5$ | Flops | 37266 | 10125000 | 1125291 | 20331000 | 2250582 | 8933088 | 19551 | × |
| $m = 50$ | Rel | $4.39 \cdot 10^{-32}$ | $0.14 \cdot 10^{-26}$ | $0.96 \cdot 10^{-30}$ | $0.13 \cdot 10^{-28}$ | $0.29 \cdot 10^{-28}$ | $0.93 \cdot 10^{-25}$ | $0.55 \cdot 10^{-25}$ | × |
| $n = 40$ | Strength | $4.67 \cdot 10^{-27}$ | $0.13 \cdot 10^{-26}$ | $0.16 \cdot 10^{-26}$ | $0.41 \cdot 10^{-26}$ | $0.42 \cdot 10^{-27}$ | $0.72 \cdot 10^{-22}$ | $0.48 \cdot 10^{-21}$ | × |
| $p = 40$ | Time | 4.225 | 96.20 | 3.22 | 198.64 | 11.39 | 51.35 | 24.91 | × |
| $d = 30$ | Flops | 358875 | 9703225 | 485534 | 19488301 | 971068 | 7433284 | 1125376 | × |

m = number of polynomials, n = the maximum degree as in (3.1), p = the second maximum degree as in (3.1), d = the degree of the GCD

Table 5.7: Comparison of GCD algorithms with randomly selected sets of many polynomials: $\varepsilon_t = 10^{-16}$, Dig = 32

# Chapter 6

# Computation of the LCM of several polynomials using the ERES method

## 6.1 Introduction

Another key problem in the area of algebraic computations is the computation of the Least Common Multiple (LCM) of polynomials. In this chapter we review the LCM problem and summarise the results of an algebraic characterisation of the LCM of sets of several polynomials [46, 47], based on the classical identity (2.3) satisfied by the LCM and GCD of two polynomials. This characterisation enabled the development of procedures for computing the LCM of a set of polynomials based on numerical GCD and approximate factorisation of polynomial ideas. The use of GCD algorithms is central to these methods and in the present study an algorithm for the computation of the LCM that uses the ERES method as an integral part for computing the GCD is presented and analysed. The use of the ERES method and its ability to produce approximate GCDs is considered for defining *approximate LCMs*.

We also present here an alternative way to compute the LCM of a set of several polynomials with inexact coefficients which avoids the computation of the associated GCD. The current approach is based on the direct use of Euclid's division algorithm through ERES operations, which is represented by the ERES Division algorithm (Algorithm 3.1). The LCM is computed by solving an appropriate linear system which is transformed to a least-squares optimisation problem in the approximate case. The properties of the linear least-squares problems are briefly discussed and a new method for the computation of approximate LCMs of sets of polynomials is developed and analysed thoroughly. Finally, various examples are given for the demonstration of the developed procedures.

## 6.2 Computation of the LCM using the GCD

The GCD and LCM problems are naturally interlinked, but they are different. For the simple case of two polynomials $p_1(s)$ and $p_2(s)$ with GCD denoted by $g(s)$ and LCM denoted by $l(s)$ we have the standard identity that $p_1(s) \cdot p_2(s) = g(s) \cdot l(s)$, which indicates the coupling of the two problems. For randomly selected polynomials, the existence of a non trivial GCD is a nongeneric property, but the corresponding LCM always exists. In the generic case the LCM is a polynomial equal to the product of all the individual polynomials of the given set. This suggests that there are fundamental differences between the two problems. We mainly focus on the following problem:

*Given a set of polynomials* $\mathcal{P} = \{p_i(s) \in \mathbb{R}[s],\ i = 1, 2, \ldots, h\}$ *define a numerical procedure for the computation of their LCM and associated factorisation by avoiding root finding. Furthermore, this numerical procedure must have the ability to work on data with numerical inaccuracies and thus lead to "approximate LCM computation".*

The above problem has been addressed in [47] and the approach followed, was based on the reduction of the computation of the LCM to an equivalent problem where the computation of GCD is an integral part. The developed methodologies depend on the proper transformation of the LCM computations to real matrix computations and thus also introduce a notion of *almost LCM*. The latter problem is of special interest when the initial data are characterised by parameter uncertainty.

**Proposition 6.1** ([46])**.** *Let* $\mathcal{P} = \{p_i(s) \in \mathbb{R}[s],\ i = 1, 2, \ldots, h\}$ *and the subsets:*

$$\mathcal{P}_\mu = \{p_1(s), p_2(s), \ldots, p_{\mu-1}(s), p_\mu(s)\}$$

*for every* $\mu = 2, \ldots, h$ *and* $\mathcal{P}_h = \mathcal{P}$*. If we denote by* $[\cdot]$ *the procedure of computing the LCM and* $[\mathcal{P}_\mu] \triangleq l_\mu(s)$ *then, the following recursive process establishes the associativity property for the LCM :*

$$\begin{array}{lllllll} [\mathcal{P}_h] & = & l_h(s) & = & [[\mathcal{P}_{h-1}], p_h(s)] & = & [l_{h-1}(s), p_h(s)] \\ [\mathcal{P}_{h-1}] & = & l_{h-1}(s) & = & [[\mathcal{P}_{h-2}], p_{h-1}(s)] & = & [l_{h-2}(s), p_{h-1}(s)] \\ & \vdots & & & & & \\ [\mathcal{P}_3] & = & l_3(s) & = & [[\mathcal{P}_2], p_3(s)] & = & [l_2(s), p_3(s)] \\ [\mathcal{P}_2] & = & l_2(s) & = & [p_1(s), p_2(s)] & & \end{array} \tag{6.1}$$

The associativity property is fundamental in the computation of the LCM. The derived computational method involves:

a) the computation of the GCD, and

b) the factorisation into two factors when one is given.

The most important characteristic of this method is that it does not require root finding procedures. The next results will be stated here without proof and provide a basis for the study of the LCM problem by following the above method.

**Theorem 6.1** ([47]). *Let $\mathcal{P} = \{p_i(s) \in \mathbb{R}[s],\ i = 1, 2, \ldots, h\}$. The polynomial $l(s)$ is an LCM of $\mathcal{P}$, if and only if there exist $g_i(s) \in \mathbb{R}[s]$, $i = 1, \ldots, h$ such that the vector*

$$\underline{g}(s) = \left[l(s), g_1(s), \ldots, g_h(s)\right]^t \tag{6.2}$$

*is the least degree solution of the equation (modulo $c \in \mathbb{R}$) :*

$$Q_{\mathcal{P}}(s) \cdot \underline{g}(s) = \underline{0} \tag{6.3}$$

The identity (6.3) has the following form:

$$\underbrace{\begin{bmatrix} 1 & -p_1(s) & 0 & 0 & \ldots & 0 & 0 \\ 0 & p_1(s) & -p_2(s) & 0 & \ldots & 0 & 0 \\ 0 & 0 & p_2(s) & -p_3(s) & \ldots & 0 & 0 \\ \vdots & & & \ddots & \ddots & \vdots & \vdots \\ 0 & \ldots & \ldots & \ldots & 0 & p_{h-1}(s) & -p_h(s) \end{bmatrix}}_{\triangleq Q_{\mathcal{P}}(s)} \underbrace{\begin{bmatrix} l(s) \\ g_1(s) \\ g_2(s) \\ \vdots \\ g_h(s) \end{bmatrix}}_{\triangleq \underline{g}(s)} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{\triangleq \underline{0}}$$

Solution of condition (6.3) gives rise to an one dimensional rational vector space since $Q_{\mathcal{P}}(s)$ is of $h \times h + 1$ and its rank is $h$. Solvability of LCM is thus equivalent to defining a least degree basis and then selecting its first coordinate. The rest of the coordinates are the factors in the minimal factorisation of $l(s)$ described by

$$l(s) = p_1(s) \cdot g_1(s) = \cdots = p_h(s) \cdot g_h(s)$$

and thus crucial in turning fractions $\dfrac{1}{p_i(s)},\ i = 1, \ldots, h$ into equivalent with common denominator i.e.

$$\left\{ \frac{1}{p_i(s)} = \frac{g_i(s)}{l(s)},\ \forall\, i = 1, \ldots, h \right\}$$

The above results suggest that the LCM computation amounts to extracting a minimal degree polynomial vector from the nullspace of a polynomial matrix [47]. The next theorem gives an important algebraic relation for the computation of the LCM of a set of polynomials using GCD computations.

*NOTATION* 6.1. Let $Q_{\mu,\nu}$ be the ordered set of lexicographically ordered sequences of $\mu$ integers from $\nu$. We shall denote by $\omega = (i_1, i_2, \ldots, i_{h-1}) \in Q_{h-1,h}$ and $\widehat{\omega} = (j)$ is the index from $(1, 2, \ldots, h)$ which is complementary to the set of indices in $\omega$.

**Theorem 6.2** ([47]). *Let* $\mathcal{P} = \left\{p_i(s) \in \mathbb{R}[s],\ i = 1, 2, \ldots, h\right\}$ *be a set of real polynomials and* $l(s) = \mathrm{lcm}\{\mathcal{P}\}$. *If we denote by*

$$\mathcal{T} = \left\{ p_\omega(s) :\ p_\omega(s) = \prod_{k=1}^{h-1} p_{i_k}(s), \forall\ \omega = (i_1, i_2, \ldots, i_{h-1}) \in Q_{h-1,h} \right\},$$

$p(s) = \prod_{i=1}^{h} p_i(s)$ *and* $g(s) = \gcd\{\mathcal{T}\}$, *then* $l(s)$ *satisfies the following properties:*

*i) For any* $\omega \in Q_{h-1,h}$, *if we can write*

$$p_\omega(s) = g(s) \cdot r_\omega(s)$$

*then*

$$l(s) = p_{\widehat{\omega}}(s) \cdot g_{\widehat{\omega}}(s)$$

*where* $g_{\widehat{\omega}}(s) = r_\omega(s)$.

*ii) The polynomials* $l(s)$, $g(s)$ *and* $p(s)$ *satisfy the identity :*

$$l(s) \cdot g(s) = p(s) \tag{6.4}$$

At this point, we need a procedure that can handle the factorisation of a polynomial into two factors, when the one is given. This factorisation can be implemented either as a polynomial division or a system of linear equations. These two approaches will be presented in the following.

### 6.2.1 Factorisation of polynomials using ERES Division

The identity (6.4) implies that the LCM $l(s)$, is actually the quotient of the division

$$\frac{p(s)}{g(s)} = l(s) \tag{6.5}$$

which suggests an algorithmic procedure for the computation of the LCM based on polynomial division. The development of such an LCM algorithm requires:

a) the GCD of the given set of polynomials, and

b) a computational procedure for the division of two polynomials.

Both of the above requirements can be handled effectively in the context of the ERES methodology. As described in Chapters 3 and 4, the ERES method with the developed Hybrid ERES algorithm is an efficient numerical tool for the computation of the GCD of polynomials. Furthermore, the developed ERES Division algorithm in Chapter 3 can also be used to calculate the result of

the division (6.5). Specifically, the vector of polynomials $p(s)$ and $g(s)$ can be transformed through ERES operations into a new vector of polynomials such that

$$\begin{bmatrix} p(s) \\ g(s) \end{bmatrix} \; - - \overset{ERES}{- - - - -} > \; \begin{bmatrix} r(s) \\ g(s) \end{bmatrix}$$

The coefficients of the LCM $l(s)$, are obtained implicitly during the ERES Division process and the polynomial $r(s)$ represents the remainder of the division $\frac{p(s)}{g(s)}$.

Moreover, in the case of sets of polynomials with numerical inaccuracies it has been proved in chapters 4 and 5 that the Hybrid ERES algorithm is capable of producing approximate GCDs of good quality within a specified range of numerical accuracy. The use of an approximate GCD in the identity (6.5) naturally leads to an approximate solution for the LCM. However, the final result may also be affected by additional numerical errors introduced by the ERES operations. But this complication can be avoided, if symbolic-rational operations are used during the application of the ERES Division algorithm. Under this assumption, the result obtained from the polynomial division (6.5), when an approximate GCD is present, will be considered as an *approximate LCM*. However, we will not proceed further our analysis for the approximate LCM problem using this methodology. A different approach will be presented in the following.

Consequently, the ERES method appears to be a significant part in the process of computing the LCM of a set of several polynomials. However, other methods, such as the Matrix Pencil method [45], can also be used for the computation of the GCD of polynomials. The following algorithm is developed in the context of symbolic-rational computations for the computation of the LCM of a set $\mathcal{P}$ of several polynomials and it is based on the results derived from the previous Theorem 6.2. The next *Symbolic-Rational (S-R) LCM algorithm* is actually a variation of the numerical LCM algorithm, which is described in [47].

---

**ALGORITHM 6.1. The S-R LCM Algorithm**

Input : $\mathcal{P} = \left\{ p_i(s) \in \mathbb{R}[s],\ i = 1, 2, \ldots, h \right\}$

Step 1 : Compute $p(s) = p_1(s)\, p_2(s) \cdots p_h(s)$.

Step 2 : Find the set $\mathcal{T} = \left\{ p_{\omega_i}(s) :\ p_{\omega_i}(s) = \prod_{k=1}^{h-1} p_{i_k}(s),\ i = 1, \ldots, h \right\}$
for all $\omega_i = (i_1, i_2, \ldots, i_{h-1}) \in Q_{h-1,h}$.

Step 3 : Compute $g(s) = \gcd\{\mathcal{T}\}$.

Step 4 : Compute $l(s) = \dfrac{p(s)}{g(s)}$ by applying the
ERES Division Algorithm 3.1 to the pair $(p(s), g(s))$.

Output : $l(s) = \mathrm{lcm}\{\mathcal{P}\}$

---

**► Computational complexity and numerical behaviour of the S-R LCM algorithm**

The previous Algorithm 6.1 behaves very well when the polynomials have integer coefficients and they are processed by using exact rational operations. However, the amount of operations (addition or multiplication) required for the computation of the initial polynomial $p(s)$ and the polynomials of the set $\mathcal{T}$ can be prohibitively high. Specifically, for $h$ polynomials with average degree $\bar{d} \geq 2$, the algorithm must perform:

$$\begin{aligned} fl(\bar{d}, h) &= (\bar{d}+1)^h + h\,(\bar{d}+1)^{h-1} \\ &= (\bar{d}+1)^h \left(1 + \frac{h}{\bar{d}+1}\right) \end{aligned} \tag{6.6}$$

operations. If we use the Hybrid ERES algorithm to compute the GCD of the set $\mathcal{T}$, then the dimensions of the initial matrix will be equal to $h \times (\bar{d}\,h+1)$ and the total number of the performed operations is about $O\left(\frac{1}{3}\,h^3 + 2\,\bar{d}^2 h^3 - \frac{2}{3}\,\bar{d}^3 h^3\right)$. Finally, the ERES Division algorithm requires about $O\left(\bar{d}^h(k+2) - k^2\right)$ operations, where $k$ denotes here the degree of the GCD. Therefore, we conclude that the S-R LCM Algorithm 6.1 can be computationally efficient only for moderate sets of polynomials.

Regarding its numerical efficiency, if the original data are given inexactly in floating-point format, it is obvious that it is not wise to perform numerical floating-point operations to compute the polynomial $p(s)$ and the polynomials of the set $\mathcal{T}$, because it is very likely to have many unnecessary numerical errors during the process. Thus, the construction of the initial polynomials of the set $\mathcal{T}$ and also the ERES Division algorithm is better to be implemented by using symbolic-rational operations in order to minimize the risk of getting erroneous results. Therefore, considering the S-R LCM algorithm, the computation of an approximate LCM of the set $\mathcal{P}$, relies on the computation of an approximate GCD given by the Hybrid ERES algorithm. In the following, we shall introduce an alternative method for the computation of an approximate LCM without computing the GCD and we will compare the results of the two methods.

**Example 6.1.** Demonstrate now the steps of the S-R LCM Algorithm 6.1, consider the polynomial set $\mathcal{P}_4 = \left\{p_i(s) \in \mathbb{R}[s],\ i = 1, 2, 3, 4\right\}$ with

$$\begin{aligned} p_1(s) &= (s-1)(s+2)^2 &&= s^3 + 3\,s^2 - 4 \\ p_2(s) &= (s+2)(s-3)^2 &&= s^3 - 4\,s^2 - 3\,s + 18 \\ p_3(s) &= (s-1)(s-3) &&= s^2 - 4\,s + 3 \\ p_4(s) &= (s+2)(s-4) &&= s^2 - 2\,s - 8 \end{aligned} \tag{6.7}$$

*Step 1:* We compute the polynomial

$$p(s) = p_1(s) \cdot p_2(s) \cdot p_3(s) \cdot p_4(s) = \tag{6.8}$$

$$= s^{10} - 7\,s^9 - 6\,s^8 + 118\,s^7 - 55\,s^6 - 759\,s^5 + 556\,s^4 + 2168\,s^3 - 1584\,s^2 - 2160\,s + 1728$$

In the generic case, this polynomial would be the actual LCM of the original set $\mathcal{P}$, unless the polynomials have common factors.

*Step 2:* Obviously, $h = 4$ and hence, we get 4 sequences $\omega_i \in Q_{3,4}$, which will be used in order to formulate the new set

$$\mathcal{T} = \left\{ p_{\omega_i}(s) :\ p_\omega(s) = \prod_{k=1}^{3} p_{i_k}(s),\ i = 1, \ldots, 4 \right\}$$

Therefore,

$$\begin{array}{rcrcl} \omega_1 = (1,2,3) & : & p_{\omega_1}(s) & = & p_1(s) \cdot p_2(s) \cdot p_3(s) \\ \omega_2 = (1,2,4) & : & p_{\omega_2}(s) & = & p_1(s) \cdot p_2(s) \cdot p_4(s) \\ \omega_3 = (1,3,4) & : & p_{\omega_3}(s) & = & p_1(s) \cdot p_3(s) \cdot p_4(s) \\ \omega_4 = (2,3,4) & : & p_{\omega_4}(s) & = & p_2(s) \cdot p_3(s) \cdot p_4(s) \end{array}$$

and

$$\begin{aligned} p_{\omega_1}(s) &= s^8 - 5\,s^7 - 8\,s^6 + 62\,s^5 + 5\,s^4 - 253\,s^3 + 90\,s^2 + 324\,s - 216 \\ p_{\omega_2}(s) &= s^8 - 3\,s^7 - 21\,s^6 + 43\,s^5 + 180\,s^4 - 168\,s^3 - 656\,s^2 + 48\,s + 576 \\ p_{\omega_3}(s) &= s^7 - 3\,s^6 - 15\,s^5 + 31\,s^4 + 78\,s^3 - 84\,s^2 - 104\,s + 96 \\ p_{\omega_4}(s) &= s^7 - 10\,s^6 + 24\,s^5 + 50\,s^4 - 245\,s^3 + 72\,s^2 + 540\,s - 432 \end{aligned}$$

*Step 3:* The basis matrix of the polynomial set $\mathcal{T}$ is

$$T = \begin{bmatrix} -216 & 324 & 90 & -253 & 5 & 62 & -8 & -5 & 1 \\ 576 & 48 & -656 & -168 & 180 & 43 & -21 & -3 & 1 \\ 96 & -104 & -84 & 78 & 31 & -15 & -3 & 1 & 0 \\ -432 & 540 & 72 & -245 & 50 & 24 & -10 & 1 & 0 \end{bmatrix} \in \mathbb{R}^{4\times 9} \tag{6.9}$$

and, using the Hybrid ERES algorithm, we obtain the GCD of $\mathcal{T}$ :

$$g(s) = s^4 - 9\,s^2 - 4\,s + 12 \tag{6.10}$$

*Step 4:* The ERES Division Algorithm 3.1 applied to the pair $(p(s), g(s))$ gives the final solution:

$$l(s) = s^6 - 7\,s^5 + 3\,s^4 + 59\,s^3 - 68\,s^2 - 132\,s + 144 \tag{6.11}$$

which is a $6^{th}$ degree polynomial representing the exact LCM of $\mathcal{P}_4$ in (6.7). □

### 6.2.2 Factorisation of polynomials using a system of linear equations

We will describe now the algebraic properties of the identity (6.4) more generally. Let $\alpha(s), \beta(s), \gamma(s) \in \mathbb{R}[s]$ and assume that

$$\alpha(s) = \beta(s) \cdot \gamma(s) \tag{6.12}$$

where

$$\begin{array}{rcll} \alpha(s) & = & a_\kappa\, s^\kappa + \ldots + a_1\, s + a_0, & a_\kappa \neq 0 \\ \beta(s) & = & b_\lambda\, s^\lambda + \ldots + b_1\, s + b_0, & b_\lambda \neq 0 \\ \gamma(s) & = & c_\mu\, s^\mu + \ldots + c_1\, s + c_0, & a_\mu \neq 0 \end{array} \tag{6.13}$$

and $\kappa, \lambda, \mu \in \mathbb{N} \setminus \{0\}$. Furthermore, define:

$$\underline{a} = [a_0, a_1, \ldots, a_\kappa]^t, \quad \underline{b} = [b_0, b_1, \ldots, b_\lambda]^t, \quad \underline{c} = [c_0, c_1, \ldots, c_\mu]^t$$

$$T_{\underline{b}} \triangleq \begin{bmatrix} b_0 & 0 & \ldots & 0 \\ b_1 & b_0 & \ldots & 0 \\ \vdots & b_1 & \ddots & b_0 \\ \vdots & \vdots & \ddots & b_1 \\ b_\lambda & \vdots & & \vdots \\ 0 & b_\lambda & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & b_\lambda \end{bmatrix} \in \mathbb{R}^{(\lambda+1+\mu)\times(\mu+1)} \tag{6.14}$$

If the equation (6.12) holds true and assuming that $\alpha(s)$ and $\beta(s)$ are given, then we have the following result.

**Proposition 6.2** ([47]). *If $\alpha(s), \beta(s), \gamma(s) \in \mathbb{R}[s]$ and satisfy (6.12) with $\kappa = \lambda+\mu$, then the following conditions are equivalent:*

*i)* $\alpha(s) = \beta(s) \cdot \gamma(s)$

*ii)* $T_{\underline{b}} \cdot \underline{c} = \underline{a}$

*iii)* $\underline{a}$ *is a member of the column space of* $T_{\underline{b}}$.

In general, the vector $\underline{c}$ of the coefficients of the polynomial $\gamma(s)$ is the solution of the overdetermined system of linear equations:

$$T_{\underline{b}} \cdot \underline{c} = \underline{a} \tag{6.15}$$

which can be obtained using standard linear algebra methods, such as Householder-QR factorisation [18, 27]. Moreover, the linear algebra formulation (6.15) of

the factorisation problem allows the introduction of the notion of *approximate factorisation* of polynomials, which has been analysed in [47].

**Definition 6.1.** Let $\alpha(s), \beta(s) \in \mathbb{R}[s]$ as defined in (6.12) – (6.15), with $\lambda \leq \kappa$. Then, $\beta(s)$ is defined as an *almost factor* of $\alpha(s)$ of order

$$\tau = \min_{\underline{c}} \|T_{\underline{b}} \cdot \underline{c} - \underline{a}\|_2 \tag{6.16}$$

for every $\underline{c} \in \mathbb{R}^{\mu+1}$.

*REMARK* 6.1. When $\tau = 0$ the polynomial $\beta(s)$ is characterised as an exact factor of $\alpha(s)$. Otherwise, it is an almost factor and the vector $\underline{c}$ defines the *complement* to the $\beta(s)$ almost factor, $\gamma(s)$.

The problem of finding the minimum solution of (6.16), with given $\underline{a}$, $\underline{b}$, is referred to as *approximate factorisation problem* (AFP) [47] and it can be solved using standard linear least-squares methods [18, 27]. The AFP problem involves two orders $(\phi, \psi)$ characterising each almost factor. The order $\phi$ is defined by

$$\phi = \left\| T_{\underline{b}}^{\perp} \, \underline{a} \right\|_2 \tag{6.17}$$

where $T_{\underline{b}}^{\perp}$ is the left annihilator of $T_{\underline{b}}$. The value of $\phi$ is a measure of proximity of $\beta(s)$ to be a true factor of $\alpha(s)$, and it is also referred to as *order of approximation* [47]. Conversely, the order $\psi$ is defined by

$$\psi = \min_{\underline{c}} \left\| T_{\underline{a}}^{\perp} \, T_{\underline{b}}^{\perp} \, \underline{c} \right\|_2 \tag{6.18}$$

where $T_{\underline{a}}^{\perp}$ is the left annihilator of $T_{\underline{a}}$. The value of $\psi$ indicates the "best" selection of $\gamma(s)$, when $\alpha(s)$ and $\beta(s)$ are given, and it is also referred to as *order of optimal completion* [47]. Both orders $\phi$ and $\psi$ are very important indicators for specifying the nature of the approximation and for evaluating the quality of the obtained numerical results.

The above theoretical results for arbitrary polynomials $\alpha(s), \beta(s), \gamma(s) \in \mathbb{R}[s]$ have a direct application to the LCM problem as stated in Theorem 6.2, if we set $\alpha(s) := p(s)$, $\beta(s) := g(s)$, and $\gamma(s) := l(s)$.

**Conclusions.** In this section, the problem of computing the LCM of sets of several polynomials has been considered using a method that involves the computation of GCD of an associated set and a factorisation procedure. This approach avoids the computation of roots and may also produce estimates of approximate LCM, when the GCD algorithm used yields approximate solutions. However, the fact that we rely on the computation of the GCD may be considered as a disadvantage for this method. Another disadvantage is that this method requires

the creation of products of many polynomials with implications on numerical complexity and stability, especially when dealing with high order polynomials. Alternative methods, which deal with the LCM and approximate LCM problems without relying on GCD procedures, will be discussed in the sequel.

## 6.3 Computation of the LCM without using the GCD

In this section we aim to develop an efficient method for computing the LCM of a set of several univariate polynomials $\mathcal{P} = \mathcal{P}_{m,n}$, without root finding procedures or GCD computation, which will eventually provide us with results of better numerical quality. A very interesting approach to the computation of LCM of real univariate polynomials that avoids root finding, as well as use of algebraic procedures of GCD computation, has been presented in [48] and it is based on standard system theory concepts. In [48], the provided system theoretic characterisation of the LCM of a given set $\mathcal{P}$ has led to an efficient numerical procedure for the computation of the LCM, and the associated set of multipliers of $\mathcal{P}$ with respect to LCM, which implies a polynomial factorisation procedure. Specifically, for a given set of polynomials $\mathcal{P}$ a natural realization $S(A, \underline{b}, C)$ is defined by inspection of the elements of the set $\mathcal{P}$. It is shown that the degree $\ell$ of the LCM is equal to the dimension of the controllable subspace of the pair $(A, \underline{b})$, whereas the coefficients of the LCM express the relation of $A^{\ell}\,\underline{b}$ with respect to the basis of the controllable space. The main advantage of this procedure is that the system is defined from the original data without involving transformations and thus, the risk of adding undesired numerical rounding errors is reduced. The vital part of this numerical procedure is the determination of the successive ranks of parts of the controllability matrix, which due to the special structure of the system may be computed using stable numerical procedures. Moreover, the companion form structure of $A$ simplifies the computation of controllability properties and leads to a simple procedure for defining the associated set of polynomial multipliers of $\mathcal{P}$ with respect to LCM. The developed results in [48] provide a robust procedure for the computation of the LCM and enable the computation of approximate values, when the original data have some numerical inaccuracies. In such cases the method computes an approximate LCM with degree smaller than the generic degree, which is equal to the sum of the degrees of all the polynomials of $\mathcal{P}$.

However, in the current study we will follow a different approach for the computation of the LCM of a set $\mathcal{P}$ without computing the associated GCD, based on the properties of polynomial division and using algebraic procedures in the context of the ERES methodology. This study will be focused on the formulation of an appropriate algebraic system of linear equations which actually provides

the coefficients of the LCM when it is solved. The solution of this system may be sought either using direct algebraic methods, such as LU factorisation, or optimisation methods in case of approximate solutions. Therefore, the current LCM method involves:

a) the formulation of a linear system through a transformation process of the original polynomials by using the ERES Division algorithm, and

b) a procedure to solve the formulated system of linear equations.

The basic idea for the development of this particular method derives from the following lemma:

**Lemma 6.1.** *Given a set of real polynomials* $\mathcal{P} = \{p_i(s) \in \mathbb{R}[s],\ i = 1, 2, \ldots, h\}$ *the LCM of* $\mathcal{P}$ *is a real polynomial with degree* $\ell \leq \sum_{i=1}^{h} \deg\{p_i(s)\}$ *and every polynomial* $p_i(s)$ *divides evenly into LCM.*

*Proof.* Denote by $l(s)$ the LCM of $\mathcal{P}$, $\ell = \deg\{l(s)\}$ and $d_i = \deg\{p_i(s)\}$ for every $i = 1, \ldots, h$. We will consider two cases:

a) If the polynomials $p_i(s) \in \mathcal{P}$ cannot be factorized into polynomials in $\mathbb{R}[s]$, the LCM is given by the product of all $p_i(s) \in \mathcal{P}$ :

$$l(s) = \prod_{i=1}^{h} p_i(s) \quad \text{and} \quad \ell = \sum_{i=1}^{h} d_i$$

Obviously, in this case every $p_i(s)$ divides evenly into LCM, $l(s)$.

b) We may assume now that the polynomials $p_i(s) \in \mathcal{P}$ can be factorized into polynomials $t_{i,j_i}(s) \in \mathbb{R}[s],\ j_i \leq d_i,\ i = 1, 2, \ldots, h$ that are irreducible over $\mathbb{R}$, i.e. $t_{i,j_i}(s)$ is non-constant and cannot be represented as the product of two or more non-constant polynomials from $\mathbb{R}[s]$. Then,

$$p_i(s) = t_{i,1}^{k_{i,1}}(s) \cdot t_{i,2}^{k_{i,3}}(s) \cdots t_{i,j_i}^{k_{i,j_i}}(s) \tag{6.19}$$

with

$$k_{i,1} + \ldots + k_{i,j_i} = d_i$$

for $j_i \in \mathbb{N}$ and $i = 1, 2, \ldots, h$.

Define the set containing the factors of every polynomial $p_i(s)$ as

$$\mathcal{T}_f = \left\{ t_{i,j}^{k_{i,j}}(s) \in \mathbb{R}[s], j = 1, 2, \ldots, j_i \text{ for } j_i \in \mathbb{N} \text{ and } i = 1, 2, \ldots, h \right\}$$

and consider the next subsets of $\mathcal{T}_f$ :

$$\begin{aligned}\mathcal{T}_{cf} &= \left\{t_{i,j}^{k_{i,j}}(s):\ t_{i,j}(s) \text{ is a common factor and } k_{i,j} \text{ is its highest power}\right\}\\ \mathcal{T}_{ncf} &= \left\{t_{i,j}^{k_{i,j}}(s):\ t_{i,j}(s) \text{ is a non-common factor}\right\}\end{aligned}$$

Then, the LCM is given as

$$l(s) = \prod_{i,j} t_{i,j}^{k_{i,j}}(s), \quad \text{where} \quad t_{i,j}^{k_{i,j}}(s) \in \mathcal{T}_{cf} \cup \mathcal{T}_{ncf}$$

and hence, considering (6.19), every $p_i(s)$ divides evenly into $l(s)$. Moreover, since $\mathcal{T}_{cf} \cup \mathcal{T}_{ncf} \subseteq \mathcal{T}_f$, it follows that

$$\ell = \sum_{i,j} k_{i,j} \leq \sum_{i=1}^{h} d_i$$

which implies that the LCM is a real polynomial with maximum degree equal to the sum of the degrees of $p_i(s) \in \mathcal{P}$.

□

Before we proceed with the analysis of this new LCM method, we will describe the concept and its basic steps through the following example.

**Example 6.2.** Consider a pair $\mathcal{P} = \{p_1(s), p_2(s)\}$ of real polynomials with arbitrary coefficients $c_i \in \mathbb{R} \setminus \{0\}$ for every $i = 1, \ldots 4$ such that

$$p_1(s) = (s + c_1)(s + c_2), \quad \deg\{p_1(s)\} = 2 \tag{6.20}$$
$$p_2(s) = (s + c_3)(s + c_4), \quad \deg\{p_2(s)\} = 2 \tag{6.21}$$

1. *Formulation of the LCM.*

   In the generic case, the LCM of $\mathcal{P}$ is a polynomial $l(s)$ with degree $\ell = 2 + 2 = 4$. Denote the LCM by

   $$l(s) = a_4\, s^4 + a_3\, s^3 + a_2\, s^2 + a_1\, s + a_0 \tag{6.22}$$

2. *Construction of the remainder vectors by using the ERES Division algorithm.*

   According to Lemma 6.1, both $p_1(s)$ and $p_2(s)$ evenly divide $l(s)$, leaving no remainder. If we denote by

   $$r_i(s) = r_1^{(i)}\, s + r_0^{(i)}\ , \quad \text{for} \quad i = 1, 2$$

   the remainders from the divisions $\frac{l(s)}{p_1(s)}$ and $\frac{l(s)}{p_2(s)}$ respectively, then it is

required that

$$r_1^{(i)} = r_0^{(i)} = 0\ , \quad \text{for} \ \ i = 1, 2 \tag{6.23}$$

and if the remainder vectors are denoted by

$$\underline{r}_1 = \begin{bmatrix} r_0^{(1)} \\ r_1^{(1)} \end{bmatrix} \quad , \quad \underline{r}_2 = \begin{bmatrix} r_0^{(2)} \\ r_1^{(2)} \end{bmatrix}$$

then (6.23) can be written as:

$$\begin{bmatrix} r_0^{(1)} \\ r_1^{(1)} \\ r_0^{(2)} \\ r_1^{(2)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{6.24}$$

The vectors of the remainders $r_i(s)$ can be obtained directly by using the ERES Division algorithm and they have the following similar symbolic forms:

$$\underline{r}_1 = \begin{bmatrix} a_0 - c_1c_2a_2 + ({c_1}^2c_2 + c_1{c_2}^2)a_3 - ({c_1}^3c_2 + {c_1}^2{c_2}^2 + c_1{c_2}^3)a_4 \\ a_1 - (c_1 + c_2)a_2 + ({c_1}^2 + c_1c_2 + {c_2}^2)a_3 - ({c_1}^3 + {c_1}^2c_2 + c_1{c_2}^2 + {c_2}^3)a_4 \end{bmatrix}$$

$$\underline{r}_2 = \begin{bmatrix} a_0 - c_3c_4a_2 + ({c_3}^2c_4 + c_3{c_4}^2)a_3 - ({c_3}^3c_4 + {c_1}^2{c_4}^2 + c_3{c_4}^3)a_4 \\ a_1 - (c_3 + c_4)a_2 + ({c_3}^2 + c_3c_4 + {c_4}^2)a_3 - ({c_3}^3 + {c_3}^2c_4 + c_3{c_4}^2 + {c_2}^3)a_4 \end{bmatrix}$$

3. *Formation of an appropriate linear system for the computation of the LCM.*

   Combining the above equations with (6.24), gives the linear system

$$F_{\mathcal{P}} \cdot \underline{a} = \underline{0} \tag{6.25}$$

where the unknowns are the coefficients of the LCM $l(s)$. Hence, we have

$$F_{\mathcal{P}} \triangleq \begin{bmatrix} 1 & 0 & -c_1c_2 & ({c_1}^2c_2 + c_1{c_2}^2) & -({c_1}^3c_2 + {c_1}^2{c_2}^2 + c_1{c_2}^3) \\ 0 & 1 & -(c_1 + c_2) & ({c_1}^2 + c_1c_2 + {c_2}^2) & -({c_1}^3 + {c_1}^2c_2 + c_1{c_2}^2 + {c_2}^3) \\ 1 & 0 & -c_3c_4 & ({c_3}^2c_4 + c_3{c_4}^2) & -({c_3}^3c_4 + {c_3}^2{c_4}^2 + c_3{c_4}^3) \\ 0 & 1 & -(c_3 + c_4) & ({c_3}^2 + c_3c_4 + {c_4}^2) & -({c_3}^3 + {c_3}^2c_4 + c_3{c_4}^2 + {c_2}^3) \end{bmatrix}$$

and

$$\underline{a} \triangleq \begin{bmatrix} a_0, & a_1, & a_2, & a_3, & a_4 \end{bmatrix}^t \ , \quad \underline{0} \triangleq \begin{bmatrix} 0, & 0, & 0, & 0, & 0 \end{bmatrix}^t$$

The matrix $F_{\mathcal{P}}$ has dimensions $4 \times 5$, which implies that (6.25) represents an underdetermined homogeneous linear system which has at least one solution. However, the linear system (6.25) has two very important characteristics:

i) the rank of $F_{\mathcal{P}}$ is equal to the actual degree of the LCM, $l(s)$ and

ii) if the LCM is a monic polynomial, then the linear system (6.25) has a unique solution.

4. *Computation of the LCM.*

- If the polynomials of the set $\mathcal{P}$ do not share a common factor, then the LCM will have the generic form:

$$l(s) = (s + c_1)(s + c_2)(s + c_3)(s + c_4)$$

and the matrix $F_{\mathcal{P}}$ has full rank. By setting $a_4 := 1$, the linear system (6.25) provides the generic LCM.

- Let $c_1 \neq c_2 \neq c_3$ and $c_4 = c_2$. Then, the LCM will have the form:

$$l(s) = (s + c_1)(s + c_2)(s + c_3)$$

In this case, we notice that the matrix $F_{\mathcal{P}}$ has rank $\rho(F_{\mathcal{P}}) = 3$. Indeed, after its triangularisation we obtain the matrix:

$$\widetilde{F}_{\mathcal{P}} \triangleq \begin{bmatrix} 1 & 0 & -c_1c_2 & c_1c_2\,(c_1+c_2) & -{c_1}^3c_2 - {c_1}^2{c_2}^2 - c_1{c_2}^3 \\ 0 & 1 & -c_1 - c_2 & {c_1}^2 + c_1c_2 + {c_2}^2 & -{c_1}^3 - {c_1}^2c_2 - c_1{c_2}^2 - {c_2}^3 \\ 0 & 0 & b_1 & b_2 & b_3 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where

$$\begin{aligned} b_1 &= -c_3c_2 + c_1c_2 \\ b_2 &= {c_3}^2c_2 + c_3{c_2}^2 - {c_1}^2c_2 - c_1{c_2}^2 \\ b_3 &= -{c_3}^3c_2 - {c_3}^2{c_2}^2 - c_3{c_2}^3 + {c_1}^3c_2 + {c_1}^2{c_2}^2 + c_1{c_2}^3 \end{aligned}$$

and therefore, if we set $a_4 = 0$ and $a_3 = 1$, the solution of the linear system

$$\widetilde{F}_{\mathcal{P}} \cdot \underline{a} = \underline{0}$$

is the desired LCM of degree 3.

- Similar results are obtained when $c_i = c_j$ for any $i, j = 1, 2, 3, 4$. Furthermore, provided that the polynomials $p_1(s)$ and $p_2(s)$ are not identical, there is not an LCM of degree less than 3.

□

**► The computation of the LCM of sets of several polynomials by using a system of linear equations**

Considering the process of computing the LCM of two polynomials as described in the previous Example 6.2, we will go further by extending this particular process in sets of several polynomials

$$\mathcal{P} = \left\{ p_i(s) \in \mathbb{R}[s],\ i = 1, 2, \ldots, h \right\}$$

The new computational method will use again the ERES Division algorithm in order to form a system of linear equations , which provides the coefficients of the LCM of a given set of polynomials. A key feature that we will prove in the following is that the rank of the matrix of the linear system denotes the actual degree of the LCM of the polynomials. These features will be analysed next.

*REMARK* 6.2. In general, the leading coefficient of the LCM is the least common multiple of the leading coefficients of all the polynomials of the set $\mathcal{P}$, but this does not hold for numerical floating-point data. In this case, the leading coefficient of the LCM is given by the product of the leading coefficients of all the polynomials of the set $\mathcal{P}$. In the following, we can assume without loss of generality that the polynomials of the set $\mathcal{P}$ are monic. Therefore, for $i = 1, 2, \ldots, h$ let

$$p_i(s) = \sum_{k=0}^{d_i} c_{i,k}\, s^k, \quad c_{i,k} \in \mathbb{R}, \ \ c_{i,d_i} = 1, \ \ d_i = \deg\{p_i(s)\} \tag{6.26}$$

Then the LCM will also be a monic polynomial of maximum degree:

$$d = \sum_{i=1}^{h} d_i \tag{6.27}$$

represented as

$$l(s) = \sum_{k=0}^{d} a_k\, s^k, \quad a_k \in \mathbb{R} \tag{6.28}$$

If $\ell \triangleq \deg\{l(s)\}$ then $a_d = a_{d-1} = \ldots = a_{\ell+1} = 0$ and $a_\ell = 1$.

We focus now on the division $\frac{l(s)}{p_i(s)}$ and, according to the Euclidean algorithm, there exist real polynomials $q_i(s)$ and $r_i(s)$ such that

$$\frac{l(s)}{p_i(s)} = q_i(s) + \frac{r_i(s)}{p_i(s)} \tag{6.29}$$

for every $i = 1, 2, \ldots, h$. The remainder polynomials $r_i(s)$ are real polynomials with degree $\deg\{r_i(s)\} < \deg\{p_i(s)\}$ and if take into account the result of Lemma 6.1, then,

$$r_i(s) = 0, \quad \forall\, i = 1, 2, \ldots, h \tag{6.30}$$

The ERES Division algorithm can be used for the computation of the remainder polynomials $r_i(s)$ so that

$$\begin{bmatrix} l(s) \\ p_i(s) \end{bmatrix} - - \overset{ERES}{- - -} > \begin{bmatrix} r_i(s) \\ p_i(s) \end{bmatrix} \tag{6.31}$$

where the coefficient vector of the polynomial $r_i(s)$ is given by

$$\underline{r}_i = \left[ r^{(i)}_{d_i-1}, \ldots, r^{(i)}_1, r^{(i)}_0 \right] \in \mathbb{R}^{d_i} \tag{6.32}$$

where $r^{(i)}_j$ denotes the $j^{th}$ element of the vector $\underline{r}_i$ and implies that

$$r_i(s) = r^{(i)}_0 + r^{(i)}_1 \, s + \ldots + r^{(i)}_{d_i-1} \, s^{d_i-1}$$

The study of the results from the application of the ERES Division algorithm to polynomials with symbolic form, show that if the polynomial $l(s)$ is written in the form (6.28) with arbitrary coefficients $a_k \in \mathbb{R}$, $k = 0, 1, 2, \ldots, d$, then every $r^{(i)}_j$ in (6.32) is a linear function of all $a_k$. Specifically, we have:

$$\widehat{\underline{r}}_i = \begin{bmatrix} r^{(i)}_0 \\ r^{(i)}_1 \\ \vdots \\ r^{(i)}_{d_i-1} \end{bmatrix} = \begin{bmatrix} f^i_{0,0} \, a_0 + f^i_{0,1} \, a_1 + \ldots + f^i_{0,d} \, a_d \\ f^i_{1,0} \, a_0 + f^i_{1,1} \, a_1 + \ldots + f^i_{1,d} \, a_d \\ \vdots \\ f^i_{d_i-1,0} \, a_0 + f^i_{d_i-1,1} \, a_1 + \ldots + f^i_{d_i-1,d} \, a_d \end{bmatrix} \tag{6.33}$$

where $f^i_{x,y}, \; x = 0, 1, \ldots, d_i - 1, \; y = 0, 1, \ldots, d$ are functions of $c_{i,k}$ from (6.26). Actually, $f^i_{x,y}$ are real numbers. Therefore, by using the ERES Division algorithm we may associate every polynomial $r_i(s)$ with a matrix $F_i$ such that

$$\widehat{\underline{r}}_i = F_i \cdot \underline{a} \tag{6.34}$$

where $\underline{a} = [a_0, a_1, \ldots, a_d]^t$ is the vector of arbitrary coefficients of the LCM, $l(s)$. If the polynomials $p_i(s)$ are written in the form (6.26), then every matrix $F_i$ appears to have the next form:

$$F_i = \left[ I_{d_i} | \widetilde{F}_i \right] = \left[\begin{array}{ccc|ccc} a_0 & \ldots & a_{d_i-1} & a_{d_i} & \ldots & a_d \\ \downarrow & & \downarrow & \downarrow & & \downarrow \\ 1 & \ldots & 0 & f^i_{0,d_i} & \ldots & f^i_{0,d} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 1 & f^i_{d_i,d_i} & \ldots & f^i_{d_i,d} \end{array}\right] \in \mathbb{R}^{d_i \times (d+1)} \tag{6.35}$$

where $I_{d_i}$ is the $d_i \times d_i$ identity matrix. Therefore, the equation (6.30) can be

written equivalently as

$$F_i \cdot \underline{a} = \underline{0}, \quad \forall\, i = 1, 2, \ldots, h \tag{6.36}$$

A bigger matrix

$$F_{\mathcal{P}} = \begin{bmatrix} F_1 \\ \vdots \\ F_h \end{bmatrix} = \left[\begin{array}{c|c} I_{d_1} & \widetilde{F}_1 \\ \vdots & \vdots \\ I_{d_h} & \widetilde{F}_h \end{array}\right] \tag{6.37}$$

with dimensions $d \times (d+1)$ can be formed such that :

$$F_{\mathcal{P}} \cdot \underline{a} = \underline{0} \tag{6.38}$$

The above linear system (6.38) is very important. The process of solving this system introduces a new algorithmic procedure for the numerical computation of the LCM. The following results provide the technical details for the development of such an algorithmic procedure.

**Proposition 6.3.** *The rank of $F_{\mathcal{P}}$ is equal to the degree of the LCM of the set $\mathcal{P}$.*

$$\rho(F_{\mathcal{P}}) = \deg\{l(s)\} \tag{6.39}$$

*Proof.* Consider the underdetermined homogeneous linear system (6.38) and denote by $n(F_{\mathcal{P}})$ the nullity of $F_{\mathcal{P}}$ and by $\rho(F_{\mathcal{P}})$ its rank. The system has $d$ equations and $d+1$ variables and therefore it is $n(F_{\mathcal{P}}) \geq 1$, which implies that there are infinite solutions.

a) If $n(F_{\mathcal{P}}) = 1$ and assuming that the LCM is a monic polynomial ($a_d = 1$), we obtain only one solution, which is actually the LCM given by:

$$l(s) = p_1(s) \cdot p_2(s) \cdots p_h(s)$$

Then,

$$\left.\begin{array}{l} \deg\{l(s)\} = \sum_{i=1}^{h} d_i = d \\ \rho(F_{\mathcal{P}}) = d + 1 - n(F_{\mathcal{P}}) = d \end{array}\right\} \Rightarrow \rho(F_{\mathcal{P}}) = \deg\{l(s)\} = d$$

b) If $n(F_{\mathcal{P}}) = n > 1$, we can set exactly $n$ free variables. However, if we observe carefully the structure of $F_{\mathcal{P}}$ in (6.35), we can see that every variable $a_k$ has a fixed position which corresponds to the term $s^k$ of the polynomial $l(s)$ and thus the first $d+1-n$ columns, which correspond to $a_k,\ k = 0, 1, \ldots, d-n$, should lead to the trivial solution. Assuming that the LCM is a monic

polynomial, the least degree solution will be obtained if we set :

$$a_{d-n+1} = 1 \;\text{ and }\; a_{d-n+2} = \ldots = a_d = 0$$

Then, we will have :

$$\left.\begin{array}{l} \deg\{l(s)\} = d - n + 1 \\ \rho(F_{\mathcal{P}}) = d + 1 - n(F_{\mathcal{P}}) = d + 1 - n \end{array}\right\} \Rightarrow \rho(F_{\mathcal{P}}) = \deg\{l(s)\} = d + 1 - n$$

□

The previous analysis leads to the following significant result.

**Theorem 6.3.** *Given a set* $\mathcal{P} = \{p_i(s) \in \mathbb{R}[s],\ i = 1, 2, \ldots, h\}$ *of monic polynomials, the corresponding LCM denoted by* $l(s) = \sum_{k=0}^{d} a_k\, s^k$ *with* $d = \sum_{i=1}^{h} \deg\{p_i(s)\}$ *is given by the least degree solution of the underdetermined linear system*

$$F_{\mathcal{P}} \cdot \underline{a} = \underline{0} \tag{6.40}$$

*where* $F_{\mathcal{P}} \in \mathbb{R}^{d\times(d+1)}$ *is associated with the remainder sequence* $(r_i(s),\ i = 1, \ldots, h)$ *which derives from the application of the ERES Division algorithm on the pairs* $(l(s),\, p_i(s))$ *for all* $i = 1, 2, \ldots, h$.

*Proof.* Let $r = \rho(F_{\mathcal{P}})$. From Proposition 6.3, we have that the degree of the LCM is equal to the rank of $F_{\mathcal{P}}$,

$$\ell \triangleq \deg\{l(s)\} = r$$

Since, the rank of $F_{\mathcal{P}}$ determines the degree of $l(s)$ and the columns of $F_{\mathcal{P}}$ correspond to the coefficients of $l(s)$ in a fixed order, the $d \times (d+1)$ linear system (6.40) can be reduced to a $d \times r$ linear system, such that

$$\widehat{F}_{\mathcal{P}} \cdot \underline{\hat{a}} + \underline{f}_{r+1}\, a_\ell + \widetilde{F}_{\mathcal{P}} \cdot \underline{\tilde{a}} = \underline{0} \quad \Rightarrow$$

$$\widehat{F}_{\mathcal{P}} \cdot \underline{\hat{a}} = -\underline{f}_{r+1} \tag{6.41}$$

where the matrix $\widehat{F}_{\mathcal{P}}$ is constructed from the first $r$ columns of $F_{\mathcal{P}}$, $\underline{\hat{a}} = [a_0, \ldots, a_{r-1}]^t$ is the vector of the first $r$ coefficients of $l(s)$ and $\underline{f}_{r+1}$ is the $r+1$ column of $F_{\mathcal{P}}$, which corresponds to the leading coefficient $a_\ell = 1$. The matrix $\widetilde{F}_{\mathcal{P}}$ is constructed from the last $d - r$ columns of $F_{\mathcal{P}}$ and $\underline{\tilde{a}} = [a_{r+1}, \ldots, a_d]^t$ is the vector of the last $d - r$ coefficients of $l(s)$, which are set equal to 0. The full-rank linear system (6.41) has a unique solution, which provides the LCM of the set of polynomials $\mathcal{P}$. □

The following example demonstrates the theoretical result from Theorem 6.3.

**Example 6.3.** Consider the same set $\mathcal{P}_4 = \{p_i(s) \in \mathbb{R}[s],\ i = 1, \ldots, 4\}$ as in Example 6.1.

$$\begin{array}{rcccl} p_1(s) & = & (s-1)(s+2)^2 & = & s^3 + 3\,s^2 - 4 \\ p_2(s) & = & (s+2)(s-3)^2 & = & s^3 - 4\,s^2 - 3\,s + 18 \\ p_3(s) & = & (s-1)(s-3) & = & s^2 - 4\,s + 3 \\ p_4(s) & = & (s+2)(s-4) & = & s^2 - 2\,s - 8 \end{array} \tag{6.42}$$

We aim to find the LCM of the set $\mathcal{P}_4$ using the new LCM approach of Theorem 6.3.

Suppose that we do not know in advance the actual degree of the LCM. Therefore, we have to represent the LCM in terms of its maximum theoretical degree $d = 3 + 3 + 2 + 2 = 10$ with arbitrary coefficients in the form:

$$l(s) = a_0 + a_1 s^1 + a_2 s^2 + a_3 s^3 + a_4 s^4 + a_5 s^5 + a_6 s^6 + a_7 s^7 + a_8 s^8 + a_9 s^9 + a_{10} s^{10}$$

Then, we can apply the ERES Division Algorithm 3.1 on every pair $(l(s), p_i(s))$, for $i = 1, 2, 3, 4$ and form the remainder sequence $(r_i(s),\ i = 1, \ldots, 4)$. Using the obtained remainder sequence, we form the matrices $F_i$ as described in (6.35) and we finally get the $10 \times 11$ matrix

$$F_{\mathcal{P}} = \left[\begin{array}{rrrrrrrrrrr} 1 & 0 & 0 & 4 & -12 & 36 & -92 & 228 & -540 & 1252 & -2844 \\ 0 & 1 & 0 & 0 & 4 & -12 & 36 & -92 & 228 & -540 & 1252 \\ 0 & 0 & 1 & -3 & 9 & -23 & 57 & -135 & 313 & -711 & 1593 \\ \hline 1 & 0 & 0 & -18 & -72 & -342 & -1260 & -4770 & -16704 & -58446 & -198036 \\ 0 & 1 & 0 & 3 & -6 & -15 & -132 & -465 & -1986 & -6963 & -25440 \\ 0 & 0 & 1 & 4 & 19 & 70 & 265 & 928 & 3247 & 11002 & 37045 \\ \hline 1 & 0 & -3 & -12 & -39 & -120 & -363 & -1092 & -3279 & -9840 & -29523 \\ 0 & 1 & 4 & 13 & 40 & 121 & 364 & 1093 & 3280 & 9841 & 29524 \\ \hline 1 & 0 & 8 & 16 & 96 & 320 & 1408 & 5376 & 22016 & 87040 & 350208 \\ 0 & 1 & 2 & 12 & 40 & 176 & 672 & 2752 & 10880 & 43776 & 174592 \end{array}\right]$$

which forms the homogeneous linear system (6.40). The rank of $F_{\mathcal{P}}$ is $\rho(F_{\mathcal{P}}) = 6$. Therefore, the degree of the LCM will be $\ell = 6$. As it has been analysed in the proof of Theorem 6.3, the linear system (6.40) can be reduced to the form:

$$\widehat{F}_{\mathcal{P}} \cdot \underline{\hat{a}} = -\underline{f}_7$$

where the matrix $\widehat{F}_{\mathcal{P}} \in \mathbb{R}^{10\times 6}$ consists of the first 6 columns of $F_{\mathcal{P}}$, $\underline{f}_7$ is the $7^{th}$

column of $F_{\mathcal{P}}$ and $\underline{\hat{a}} = [a_0, a_1, \ldots, a_5]^t$. Since the polynomials of the original set $\mathcal{P}_4$ are monic and the LCM has degree equal to 6, then $a_6 = 1$ and $a_k = 0$ for all $k = 7, \ldots, 10$. Finally, we end up with the overdetermined linear system:

$$\begin{bmatrix} 1 & 0 & 0 & 4 & -12 & 36 \\ 0 & 1 & 0 & 0 & 4 & -12 \\ 0 & 0 & 1 & -3 & 9 & -23 \\ 1 & 0 & 0 & -18 & -72 & -342 \\ 0 & 1 & 0 & 3 & -6 & -15 \\ 0 & 0 & 1 & 4 & 19 & 70 \\ 1 & 0 & -3 & -12 & -39 & -120 \\ 0 & 1 & 4 & 13 & 40 & 121 \\ 1 & 0 & 8 & 16 & 96 & 320 \\ 0 & 1 & 2 & 12 & 40 & 176 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} 92 \\ -36 \\ -57 \\ 1260 \\ 132 \\ -265 \\ 363 \\ -364 \\ -1408 \\ -672 \end{bmatrix}$$

Since the matrix $\widehat{F}_{\mathcal{P}}$ has full rank, the above linear system has a unique solution, which can be computed by using standard algebraic methods, such as LU decomposition or Gaussian elimination [18, 27]. The obtained solution is

$$\underline{\hat{a}} = [144, -132, -68, 59, 3, -7]^t$$

and corresponds to the exact LCM of the set $\mathcal{P}_4$ :

$$l(s) = 144 - 132\,s - 68\,s^2 + 59\,s^3 + 3\,s^4 - 7\,s^5 + s^6$$

□

## 6.4 The Hybrid LCM method and its computational properties

In this section we will consider the formulation of a computational method for the LCM of sets of polynomials based on the result provided from Theorem 6.2. Furthermore, we will discuss its proper implementation, especially in the case where numerical inaccuracies are present and an approximate solution is sought.

### ▶ Description of the Hybrid LCM method

Consider a set of real monic polynomials $\mathcal{P} = \{p_i(s) \in \mathbb{R}[s],\ i = 1, 2, \ldots, h\}$. Since we do not know the actual degree of the LCM in advance, we can represent

in terms of its maximum theoretical degree $d = \sum_{i=1}^{h} \deg\{p_i(s)\}$ with arbitrary coefficients $a_k$ in the form:

$$l(s) \triangleq a_d\, s^d + a_{d-1}\, s^{d-1} + \ldots + a_1\, s + a_0 \tag{6.43}$$

We now follow the procedures:

1. Apply the ERES Division algorithm to the pair of polynomials $(l(s), p_i(s))$ for $i = 1, 2, \ldots, h$ to obtain a vector $\underline{r}_i$, which contains the coefficients of the remainder $r_i(s)$ in symbolic form.

2. From the vectors $\underline{r}_i$ for all $i = 1, 2, \ldots, h$, form the matrix $F_{\mathcal{P}}$ as described in (6.32) – (6.38) and compute its rank.

3. If $r = \rho(F_{\mathcal{P}})$, we solve the $d \times r$ linear system $\widehat{F}_{\mathcal{P}} \cdot \underline{\hat{a}} = -\underline{f}_{r+1}$ where the matrix $\widehat{F}_{\mathcal{P}}$ is constructed from the first $r$ columns of $F_{\mathcal{P}}$, $\underline{\hat{a}} = [a_0, \ldots, a_{r-1}]^t$ is the vector of the first $r$ coefficients of $l(s)$ and $\underline{f}_{r+1}$ is the $r+1$ column of $F_{\mathcal{P}}$, which corresponds to the leading coefficient $a_r = 1$.

This process formulates a new method for the numerical computation of the LCM of a set of several polynomials based on the ERES methodology. Due to the nature of the method, the derived algorithm has to be implemented in a programming environment that has the ability to manipulate the data both symbolically and numerically using hybrid computations. This is a basic requirement, since the LCM is given as input to the ERES Division algorithm in pure symbolic form. Therefore, the procedures involved must be implemented very carefully using either symbolic or numerical computations. In the following, we shall refer to the above method of computing the LCM of a set of several polynomials as the *Hybrid LCM method* and we will analyse and discuss its computational properties.

### ▶ Computational properties of the Hybrid LCM method

The Hybrid LCM method has two main parts:

I. The computation of the remainder sequence $\{\underline{r}_i,\ i = 1, 2, \ldots, h\}$ by using the ERES Division algorithm.

II. The computation of the coefficients of the LCM by solving the linear system (6.41).

If the polynomials $p_i(s) \in \mathcal{P}$ have integer or rational coefficients, the solution of the linear system (6.41) can be computed exactly using symbolic-rational operations. However, it is important here to investigate how this method behaves when there are numerical inaccuracies in the input data. Once again, we are

interested in reducing the accumulation of rounding errors and at the same time we aim to reveal the presence of approximate solutions.

Since the LCM is represented with arbitrary coefficients, the first part of the method, which involves the ERES Division algorithm, must be treated symbolically performing symbolic-rational computations. The ERES Division is applied to the pairs $(l(s), p_i(s))$ for every $i = 1, 2, \ldots, h$. If $d_i = \deg\{p_i(s)\}$, the polynomials $p_i(s)$ can be arranged according to their degree and then the ERES Division algorithm is capable of providing a symbolic uniform algebraic formula for every $d_i$. The remainder vector $\underline{r}_i$ is obtained from the substitution of the coefficients of $p_i(s)$ in the corresponding symbolic algebraic formula. Thus, the ERES Division algorithm will be "called" $h$ times at the most.

The second part of the method involves:

- the construction of the matrix $F_{\mathcal{P}}$ from the remainder vectors $\underline{r}_i$,
- the computation of the rank of $F_{\mathcal{P}}$ and
- the solution of the linear system (6.41).

More important is the computation of the rank of $F_{\mathcal{P}}$, which determines the degree of the LCM as described in Proposition 6.3. Therefore, when numerical data are used, we can specify a small numerical accuracy $\varepsilon_t$ and the result of Proposition 6.3 can be stated as follows.

**Proposition 6.4.** *The numerical $\varepsilon_t$-rank of $F_{\mathcal{P}}$ determines the degree of the LCM of the set $\mathcal{P}$.*

The numerical accuracy $\varepsilon_t$ should be consistent with the machine precision, for example $\varepsilon_t = \mathbf{u}\,\|F_{\mathcal{P}}\|_\infty$, where $\mathbf{u} = 2^{-52}$ in 16-digits arithmetic precision. Therefore, by setting a numerical accuracy $\varepsilon_t$, we may have various degrees for the LCM which suggest the computation of an *approximate $\varepsilon_t$-LCM*. However, for a given $\varepsilon_t$-degree we need a reliable numerical procedure for the computation of a quality approximate LCM.

The numerical rank of the matrix $F_{\mathcal{P}}$ can be computed by using singular value decomposition [27, 76], which is the most reliable numerical method for the estimation of the rank of a matrix in finite precision arithmetic. When dealing with inexactly known data the linear system (6.41) must be written as:

$$F_{\mathcal{P}} \cdot \underline{a} \approx \underline{0} \tag{6.44}$$

The following proposition leads to the computation of an approximate $\varepsilon_t$-LCM.

**Proposition 6.5.** *If $\varepsilon_t$ is a specified small numerical accuracy and $r$ is the numerical $\varepsilon_t$-rank of $F_{\mathcal{P}}$, then an $\varepsilon_t$-LCM can be computed by solving the system:*

$$\widehat{F}_{\mathcal{P}} \cdot \hat{\underline{a}} = -\underline{f}_{r+1} + \underline{\varepsilon} \tag{6.45}$$

*where the matrix $\widehat{F}_{\mathcal{P}}$ is constructed from the first $r$ columns of $F_{\mathcal{P}}$, $\underline{\hat{a}} = [a_0, \ldots, a_{r-1}]^t$ is the vector of the first $r$ coefficients of $l(s)$, $\underline{f}_{r+1}$ is the $r+1$ column of $F_{\mathcal{P}}$, which corresponds to the leading coefficient $a_r = 1$ and $\underline{\varepsilon}$ is a $d$-vector of small numbers of magnitude $O(\varepsilon_t)$.*

*Proof.* This result is straightforward if we combine the results from Theorem 6.40 and Proposition 6.4. □

It is clear that an exact LCM is computed by the linear system (6.45) when $\underline{\varepsilon} = \underline{0}$. Therefore, we actually seek a solution so that the norm:

$$\mathcal{L} = \left\| \widehat{F}_{\mathcal{P}} \cdot \underline{\hat{a}} + \underline{f}_{r+1} \right\|$$

is minimized. If we use the Euclidean norm $\| \cdot \|_2$ the latter implies a *least-squares* solution for the linear system:

$$\widehat{F}_{\mathcal{P}} \cdot \underline{\hat{a}} \approx -\underline{f}_{r+1} \tag{6.46}$$

### ▶ The linear Least-Squares problem and its numerical computation

Among all the calculations studied in numerical analysis, the most frequently performed in engineering and mathematical sciences is *least-squares* estimation. The literature on least-squares solutions is quite rich and mainly refers to the sensitivity of the least-squares problem to perturbations [9, 25, 28, 29, 31, 54, 74, 78].

**Existence of least-squares solutions and sensitivity analysis.** The next theorems, stated here without proof, provide the theoretical background for computing the solution of the linear least-squares problem

$$\mathcal{L}(A, \underline{b}) \triangleq \min_{\underline{x}} \|A\,\underline{x} - \underline{b}\|_2$$

associated with the linear system $A\,\underline{x} = \underline{b}$.

The first theorem guarantees the existence and uniqueness of a least-squares solution.

**Theorem 6.4** (Least-Squares existence and uniqueness theorem [18])**.** *There always exists a solution to the linear least-squares problem*

$$\mathcal{L}(A, \underline{b}) \triangleq \min_{\underline{x}} \|A\,\underline{x} - \underline{b}\|_2 \tag{6.47}$$

*where $A$ is a $m \times n$ matrix with $m \geq n$, $\underline{b}$ is a real $m$-vector and $\underline{x}$ is a $n$-vector representing the solution. This solution is unique if and only if $A$ has full rank, that is $\rho(A) = n$.*

The next theorem relates the unique least-squares solution with the pseudo-inverse of the matrix $A$ of the linear system.

**Theorem 6.5** (Least-squares solution using the pseudo-inverse [18]). *The unique least-squares solution $\underline{x}$ to the full-rank overdetermined least-squares problem $A\,\underline{x} = \underline{b}$ is given by*

$$\underline{x} = (A^t A)^{-1} A^t\,\underline{b} = A^\dagger\,\underline{b}$$

*where $A^\dagger = (A^t A)^{-1} A^t$ is the $n \times m$ pseudo-inverse or Moore-Penrose generalized inverse of $A$.*

**Definition 6.2.** If an $m \times n$ matrix $A$ has full rank, then the condition number of A is given by

$$\mathrm{Cond}(A) = \|A\|\,\|A^\dagger\|$$

The next result, due to Wedin [80], shows that it is the condition number of matrix $A$ that plays a significant role in the sensitivity analysis of the pseudo-inverse matrix.

**Theorem 6.6** (Pseudo-inverse sensitivity theorem [18, 80]). *Let $A$ be $m \times n$, where $m \geq n$. Let $A^\dagger$ and $\tilde{A}^\dagger$ be, respectively, the pseudo-inverse of $A$ and of $\tilde{A} = A + E$. Then, provided that $\rho(A) = \rho(\tilde{A})$, we have*

$$\frac{\|\tilde{A}^\dagger - A^\dagger\|_2}{\|\tilde{A}^\dagger\|_2} \leq \epsilon\,\mathrm{Cond}(A)\,\frac{\|E\|_2}{\|A\|_2}$$

*where $\epsilon$ is a small number.*

The following theorem shows that the residual sensitivity always depends upon the condition number of the matrix $A$. A precise statement and proof of a result in residual sensitivity is given in [25, 27].

**Theorem 6.7** (Least-Squares residual sensitivity theorem [18, 27]). *Let $\underline{u}$ and $\underline{\tilde{u}}$ denote the residual, respectively, for the original and the perturbed least-squares problems*

$$\begin{aligned} \underline{u} &= \underline{b} - A\,\underline{x} \\ \underline{\tilde{u}} &= \underline{b} - (A + E)\,\underline{\tilde{x}} \end{aligned}$$

*Then*

$$\frac{\|\underline{\tilde{u}} - \underline{u}\|_2}{\|\underline{b}\|_2} \leq \varepsilon\,(1 + 2\,\mathrm{Cond}(A)) + O\left(\varepsilon^2\right)$$

*where $\varepsilon = \frac{\|E\|_2}{\|A\|_2}$.*

Finally, the next theorem refers to the sensitivity of the least-squares solution when the matrix $A$ is perturbed.

**Theorem 6.8** (Least-squares perturbation theorem [18])**.** *Let $\underline{x}$ and $\hat{\underline{x}}$ be the unique least-squares solutions to $A\,\underline{x} = \underline{b}$ and $(A+E)\,\hat{\underline{x}} = \underline{b}$, and let $\rho(A+E)$ be the same as $\rho(A)$. If $\Delta x = \hat{\underline{x}} - \underline{x}$, then*

$$\frac{\|\Delta x\|}{\|\underline{x}\|} \leq 2\,\mathrm{Cond}(A)\,\frac{\|E_A\|}{\|A\|} + 4(\mathrm{Cond}(A))^2\,\frac{\|E_N\|}{\|A\|}\,\frac{\|b_N\|}{\|b_R\|} + O\left(\frac{\|E_N\|}{\|A\|}\right)^2$$

*where $E_A$, $b_R$ denote the projections of $E$ and $b$ onto the range of $A$, denoted by $\mathcal{R}(A)$, and $E_N$, $b_N$ denote their projections onto the orthogonal complement of $\mathcal{R}(A)$, respectively.*

The above theorem tells us that the sensitivity of the unique least-squares solution, in general, depends upon the square of the condition number of $A$, expect in cases where the residual is zero. Then, the sensitivity depends only on the condition number of $A$. The condition number of linear least squares problems has been an open issue since 1966 when Golub and Wilkinson [28] found an error bound that contains the square of the coefficient matrix's condition number. Golub and Wilkinson's bound, when applied to a general least squares problem, is

$$\frac{\|\Delta x\|_2}{\|\underline{x}\|_2} \leq \frac{\|\Delta b\|_2}{\|\underline{b}\|_2}\,\frac{\|\,\underline{b}\|_2}{\|\underline{x}\|_2\,\sigma_{min}} + \frac{\|\Delta A\|_2}{\|A\|_2}\left(1 + \frac{\|\Delta u\|_2}{\|\underline{x}\|_2\,\sigma_{min}}\right)\frac{\sigma_{max}}{\sigma_{min}} + O(\varepsilon^2)$$

where $\sigma_{max}$ and $\sigma_{min}$ are the maximum and minimum singular values of $A$ and it is assumed that a) $A$ has full column rank, b) $\frac{\|\Delta A\|_2}{\|A\|_2} \leq \varepsilon$ and $\frac{\|\Delta b\|_2}{\|\underline{b}\|_2} \leq \varepsilon$, where $\varepsilon$ is arbitrarily small.

However, in recent years it has been proved that *optimal condition numbers* depend on the size of *optimal backward errors* [29, 78].

**Definition 6.3** (Function of optimal backward errors [29])**.** The optimal backward errors, for an approximate solution $x \approx x_0$ , may be defined as the solution of a minimization problem

$$\mu(x) \triangleq \min_{y:F(x,y)=0}(\|y - y_0\|) \tag{6.48}$$

where $F(x,y)$ represents the residual function of $x$, $y$ with the property $F(x_0,y_0) = 0$. If $y$ attains equation (6.48)'s minimum, then $y - y_0$ is an *optimal backward error*, and $\mu(x)$ is its size.

The size of the optimal backward error is a function of $x$ that depends on $y_0$ and on the norm chosen for $\mathbb{R}^m$. For a given vector $\underline{x}$, a backward error is a perturbation matrix, $\Delta A$, for which the given $\underline{x}$ exactly solves the perturbed problem

$$\mathcal{L}(A,\underline{b}) \triangleq \min_{\underline{x}} \|(A+\Delta A)\,\underline{x} - \underline{b}\|_2$$

In 1995, Waldén, Karlson and Sun [78] showed that the smallest Frobenius-norm perturbations, $\Delta A$, that make a given nonzero $\underline{x}$ into a solution of the perturbed problem have size

$$\mu_F^{(LS)}(\underline{x}) = \left( \frac{\|\underline{u}\|_2^2}{\|\underline{x}\|_2^2} + \min(0, \lambda) \right)^{\frac{1}{2}} , \quad \text{for } \lambda = \lambda_{min} \left( A\,A^t + \frac{\underline{u}\,\underline{u}^t}{\|\underline{x}\|_2^2} \right)$$

where $\underline{u} = A\,\underline{x} - \underline{b}$ is the approximate least-square residual of the unperturbed problem, and $\lambda_{min}$ is the smallest eigenvalue of the $m \times m$ matrix $A$.

A thorough review of different approaches to the problem of least-squares numerical sensitivity regarding condition numbers and backward errors is presented in [29]. In this report, simple expressions are given for the asymptotic size of optimal backward errors for least squares problems and it is shown that such formulas can be used to evaluate condition numbers. For full rank problems, Frobenius norm condition numbers are determined exactly, and spectral norm condition numbers are determined within a factor of the square-root-two. Specifically, if $A$ has full rank, then in [29, 78] it is proved that the Frobenius norm relative condition number is

$$\chi_F^{(LS,rel)} = \frac{\|A\|_F}{\sigma_{min}} \left( \frac{\|\underline{u}\|_2^2}{\|\underline{x}\|_2^2\,\sigma_{min}^2} + 1 \right)^{\frac{1}{2}}$$

where $\sigma_{min}$ is the smallest nonzero singular value of $A$. Moreover, the following expression overestimates the spectral norm relative condition number by at most the factor $\sqrt{2}$ :

$$\chi_2^{(LS,rel)} \approx \kappa_2 \left( \frac{\|\underline{u}\|_2}{\|\underline{x}\|_2\,\sigma_{min}^2} + 1 \right)$$

where $\kappa_2 = \frac{\sigma_{max}}{\sigma_{min}}$ is the spectral matrix condition number, and $\sigma_{max}$ is the largest singular value of $A$. These results can be used to draw conclusions about the conditioning of the problem.

**Theorem 6.9** (Well conditioned least-squares problems [29])**.** *Suppose least-squares problem (6.47) has $A$ of full column rank and exact solution $\underline{x}_0 \neq 0$. The problem is well conditioned with respect to perturbations of the matrix if and only if:*

*i) $\|\underline{u}_0\|_2$ is at most moderately larger than $\|\underline{x}_0\|_2\,\sigma_{min}$ , and*

*ii) $A$ is well conditioned,*

*where $\underline{u}_0 = A\,\underline{x}_0 - b$ is the exact least squares residual, and $\sigma_{min}$ is the smallest singular value of $A$.*

Similar results from different approaches on backward error estimations have also been presented in [9, 31, 51, 71].

**Methods of computing least-squares solutions.** One of the most widely used approaches for computing the least-squares solution is the *normal equations method*. This method is based upon the solution of the system of normal equations:

$$A^t A\underline{x} = A^t\,\underline{b}$$

using Cholesky factorization. Unfortunately, the normal equations method is not characterized as a reliable numerical method for computing a least-squares solutions. The accuracy of the least-squares solution using normal equations will depend upon the square of the condition number of the matrix $A$. However, in certain cases, such as when the residual is zero, the sensitivity of the problem depends only on the condition number of $A$ and thus the normal equations method may introduce more errors in the solution than the data warrant [18].

Another approach to the least-squares problem is the use of orthogonal transformations [18, 27]. Specifically, the QR decomposition of matrix $A$ can lead to a simplified linear system which can be solved easily by backward substitution. If

$$Q^t\,A = R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$$

is the QR decomposition of $A$, then, because the length of a vector is preserved by an orthogonal matrix multiplication, it holds:

$$\|A\,\underline{x} - \underline{b}\|_2 \;=\; \|Q^t\,A\,\underline{x} - Q^t\,\underline{b}\|_2 \;=\; \|R_1\,\underline{x} - \underline{c}\|_2 + \|\underline{d}\|_2$$

where

$$Q^t\,\underline{b} = \begin{bmatrix} \underline{c} \\ \underline{d} \end{bmatrix}$$

Therefore, $\|A\,\underline{x} - \underline{b}\|_2$ will be minimized if $\underline{x}$ is chosen [18] so that

$$R_1\,\underline{x} - \underline{c} = 0, \quad \|\underline{u}\|_2 = \|\underline{d}\|_2$$

A method for solving the least-squares problem with QR decomposition using Householder transformations was first proposed by Golub [25]. The *Golub-Householder* algorithm is a stable and efficient method that computes least-squares solutions by using Householder matrices to decompose $A$ into QR. An alternative technique for QR factorization is the Gram-Schmidt process [18]. The Gram-Schmidt method, when used to solve the linear least-squares problem, is slightly more expensive than Householder method, but it seems that there are numerical difficulties either using its classical version (CGS) or the modified version (MGS) [10, 18]. However, the Gram-Schmidt method, as far as the least-squares problem is concerned, is considered to be numerically equivalent to Householder QR

factorization.

Another reliable numerical least-squares method relies on the Singular Value Decomposition (SVD) of the system's matrix. The computation of the singular values of $A$ is needed to determine the numerical rank of $A$ and it has also been proved [25] that SVD is an effective tool to solve the least-squares problem, both in the full rank and rank deficient cases. Considering the least-squares problem $\min_{\underline{x}} \|A\,\underline{x} - \underline{b}\|_2$, let $A = U\,\Sigma\,V^t$ be the SVD of $A$. Then we have

$$\begin{aligned} \|u\|_2 &= \|U\,\Sigma\,V^t\,\underline{x} - \underline{b}\|_2 \\ &= \|U(\Sigma\,V^t\,\underline{x} - U^t\,\underline{b})\|_2 \\ &= \|\Sigma\,\underline{y} - \underline{b'}\|_2 \end{aligned} \tag{6.49}$$

where $V^t\,\underline{x} = \underline{y}$ and $U^t\,\underline{b} = \underline{b'}$. Therefore, the use of SVD of $A$ reduces the least-squares problem for a full matrix $A$ to one with a diagonal matrix $\Sigma$ which is trivial to solve. If $\sigma_i,\ i = 1, 2, \ldots, n$ are the singular values of $A$, then the vector $\underline{y} = [y_1, y_2, \ldots, y_n]^t$ that minimizes (6.49) is given by [18, 27]:

$$y_i = \begin{cases} \dfrac{b'_i}{\sigma_i}, & \text{if } \sigma_i \neq 0 \\ t = \text{arbitrary}, & \text{if } \sigma_i = 0 \end{cases} \tag{6.50}$$

Once $\underline{y}$ is computed, the solution can be recovered from

$$\underline{x} = V \cdot \underline{y} \tag{6.51}$$

The numerical algorithm for solving least-squares problems with the SVD method, using the Golub-Kahan-Reinsch (GKR) algorithm, is described in [27]. A numerical analysis of this algorithm is also presented in [18]. The algorithm requires about $2mn^2 + 4n^3$ floating-point operations to solve the least-squares problem when $A$ is $m \times n$ and $m \geq n$. Furthermore, there is no need to compute the whole vector $\underline{b'}$ and only the columns of $U$ that correspond to the non-zero singular values are needed in computation.

### 6.4.1 The numerical computation of an approximate LCM using the Hybrid LCM method

The system (6.46) is a full-rank overdetermined linear system and, instead of solving the equations exactly, we seek only to minimize the sum of the squares of the residuals

$$\underline{u} = \widehat{F}_{\mathcal{P}} \cdot \underline{\hat{a}} + \underline{f}_{r+1}$$

According to the previous theoretical results, the linear system

$$\widehat{F}_{\mathcal{P}} \cdot \underline{\hat{a}} \approx -\underline{f}_{r+1}$$

has a unique least-squares solution, which can be represented as

$$\underline{\hat{a}} = \widehat{F}_{\mathcal{P}}^{\dagger} \cdot (-\underline{f}_{r+1}) \tag{6.52}$$

where $\widehat{F}_{\mathcal{P}}^{\dagger}$ is the pseudo-inverse of $\widehat{F}_{\mathcal{P}}$.

*REMARK* 6.3. The residual norm $\|\underline{u}\|_2$ characterises the proximity of the computed solution to the exact solution of the LCM problem and we may consider it as a measure of quality of the approximate $\varepsilon_t$-LCM.

The main result from the preceding analysis can be summarized in the following theorem.

**Theorem 6.10.** *Let $\mathcal{P} = \{p_i(s) \in \mathbb{R}[s],\ i = 1, 2, \ldots, h\}$ a set of real monic polynomials. The matrix $F_{\mathcal{P}} \in \mathbb{R}^{d\times(d+1)}$ is associated with the remainder sequence $(r_i(s))$, which derives from the application of the ERES Division algorithm on the pairs $(l(s),\, p_i(s))$ for all $i = 1, 2, \ldots, h$. If $r$ denotes the numerical $\varepsilon_t$-rank of $F_{\mathcal{P}}$, then an approximate $\varepsilon_t$-LCM of $\mathcal{P}$ is given by the solution of the least-squares problem :*

$$\mathcal{L}\left(\widehat{F}_{\mathcal{P}}, -\underline{f}_{r+1}\right) \triangleq \min_{\underline{\hat{a}}} \left\| \widehat{F}_{\mathcal{P}} \cdot \underline{\hat{a}} + \underline{f}_{r+1} \right\|_2 \tag{6.53}$$

*where the matrix $\widehat{F}_{\mathcal{P}}$ is constructed from the first $r$ columns of $F_{\mathcal{P}}$, $\underline{\hat{a}} = [a_0, ..., a_{r-1}]^t$ is the vector of the first $r$ coefficients of the LCM and $\underline{f}_{r+1}$ is the $r+1$ column of $F_{\mathcal{P}}$, which corresponds to the leading coefficient of the LCM, $a_r = 1$ .*

**Corollary 6.1.** *The residual from the solution of the linear least-squares problem (6.53) characterises the numerical quality of the given approximate LCM of the original set of polynomials $\mathcal{P}$.*

If the residual from the obtained least-squares solution is close enough to zero (according to the Euclidean norm), then it can be considered as a "good" approximation of the LCM of the original set of polynomials. The unique solution of the least-squares problem (6.53), which gives the coefficients of the $\varepsilon_t$-LCM, can be acquired by using either a QR or SVD least-squares method. In fact, mathematical software packages, such as Matlab and Maple, contain efficient built-in routines for the linear least-squares problem where QR decomposition or SVD algorithms are used.

► **The implementation of the Hybrid LCM algorithm**

Having made the necessary preparation, we are now ready to form the *Hybrid LCM algorithm* for computing the approximate LCM of a set of several real polynomials.

---

**ALGORITHM 6.2. The Hybrid LCM algorithm**

Input : $\mathcal{P} = \big\{p_i(s) \in \mathbb{R}[s],\ i = 1, 2, \ldots, h\big\}$.
Set a small numerical accuracy $\varepsilon_t > 0$.

Step 1 : Compute the degrees of the polynomials $p_i(s)$ :
$d_i := \deg\{p_i\},\ i = 1, 2, \ldots, h.$
$d := d_1 + d_2 + \ldots + d_h.$
Step 2 : Form the arbitrary polynomial $l(s) = a_d\, s^d + \ldots + a_1\, s + a_0$.

**For** $i = 1, 2, \ldots, h$ **do**
Step 3 : Apply the ERES Division algorithm to $(l(s), p_i(s))$ :
$P := ERESDiv(l(s), p_i(s))$
Step 4 : Take the first row of $P$ and form the matrix $F_i$.
**end for**

Step 5 : Form the $d \times (d+1)$ matrix $F_{\mathcal{P}} = [F_1, \ldots, F_h]^t$.
Step 6 : Normalize the rows of $F_{\mathcal{P}}$ using norm-2.
Step 7 : Compute the singular values of $F_{\mathcal{P}}$ and specify $r$
such that $\sigma_1 > \ldots > \sigma_r > \varepsilon_t \geq \sigma_{r+1} > \ldots > \sigma_d$.
Step 8 : Form the matrix $\widehat{F}_{\mathcal{P}}$ from the first $r$ columns of $F_{\mathcal{P}}$.
Step 9 : Form the vector $\underline{f}_{r+1}$ from the $r+1$ column of $F_{\mathcal{P}}$.
Step 10 : Compute the solution of the least-squares problem :
$\mathcal{L}(\widehat{F}_{\mathcal{P}}, -\underline{f}_{r+1}\} = \min \|\widehat{F}_{\mathcal{P}} \cdot \underline{\hat{a}} + \underline{f}_{r+1}\|_2$

Output : $\underline{\hat{a}} = [a_0, a_1, \ldots, a_{r-1}]^t$,
$l(s) = s^r + a_{r-1}\, s^{r-1} + \ldots + a_1\, s + a_0$

---

► **Computational complexity**

Assuming that all the polynomials $p_i(s) \in \mathcal{P}$ have the same degree $\bar{d}$, that is $d_i = \bar{d}$ for all $i = 1, 2, \ldots, h$, then $d = \bar{d}\, h$. The required number of operations for

the ERES Division algorithm is

$$\begin{aligned} fl_{ED}(h,\bar{d}\,) &= h(\bar{d}+2)(d-\bar{d}+1) \\ &= \left(\bar{d}^2+2\,\bar{d}\right)h^2-\left(\bar{d}^2+\bar{d}-2\right)h \end{aligned} \tag{6.54}$$

Since we have $h$ polynomials, the number of data that the algorithm produces is $\bar{d}\,h$. Thus, the algorithm is considered to be efficient [18] if

$$fl_{ED}(h,\bar{d}\,) \leq (\bar{d}\,h)^3$$

that is:

$$\bar{d}^3\,h^3-\left(\bar{d}^2+2\,\bar{d}\right)h^2+\left(\bar{d}^2+\bar{d}-2\right)h \geq 0$$

It can be easily proved that the above inequality holds for every $\bar{d},\ h \geq 2$ and therefore, the ERES Division algorithm is efficient in handling sets of many polynomials.

The computation of the singular values of the $d \times (d+1)$ matrix $F_{\mathcal{P}}$ requires [27] :

$$fl_{SV}(d) = \frac{4}{3}\,d^3+2\,d^2 \tag{6.55}$$

operations, using the GKR-SVD algorithm. Alternatively, we could use the Chan-SVD algorithm [13], but it requires $2\,d^3+d^2$ operations, which is greater than $fl_{SV}(d)$. Furthermore, it has been observed that any SVD algorithm gives more accurate results when it is applied to a normalized matrix. Thus, if we normalize the rows of the matrix $F_{\mathcal{P}}$ by using the Euclidean norm, the additional arithmetic operations required are:

$$fl_N(d) = 2\,d^2+4\,d+2 \tag{6.56}$$

When solving the least-squares problem through the SVD, only $\Sigma$ and $V$ have to be computed. According to the algorithm that is used, the required numerical operations for the $d \times r$ least-squares problem are [27] :

$$\begin{aligned} \text{GKR}-\text{SVD} &: 2\,d\,r^2+4\,r^3 \\ \text{Chan}-\text{SVD} &: d\,r^2+\frac{17}{3}\,r^3 \end{aligned}$$

If $r < \frac{3}{5}\,d$ it is more efficient to use the GKR-SVD algorithm, otherwise we should use the Chan-SVD algorithm. Therefore, the required operations for the least-squares problem (6.53) are:

$$fl_{LS}(d,r) = \begin{cases} 2\,d\,r^2+4\,r^3, & r < \frac{3}{5}\,d \\ d\,r^2+\frac{17}{3}\,r^3, & r \geq \frac{3}{5}\,d \end{cases} \tag{6.57}$$

Alternatively, we can obtain the least-squares solution by using the QR method.

Then the computational cost of the algorithm is dominated by the QR decomposition of $\widehat{F}_{\mathcal{P}}$. The overall cost for the full-rank least-squares solution using the Golub-Householder method [27] is :

$$fl_{QR}(d,r) = r^2\left(d - \frac{r}{3}\right) \tag{6.58}$$

Conclusively, given a set $\mathcal{P}$ of $h$ real polynomials with average degree $\bar{d} \geq 2$ and the rank of $F_{\mathcal{P}}$ is $r$ (which is equal to the degree of its LCM), then, considering the equations (6.54)-(6.58), the total amount of operations for the Hybrid LCM algorithm is:

$$O\left(\frac{4}{3}\,(\bar{d}\,h)^3 + \frac{3}{2}\,(\bar{d}\,h)r^2 + 5\,r^3\right) \tag{6.59}$$

In the worst case, where the LCM has the maximum degree $d = \bar{d}\,h$, the algorithm performs less than $8\,d^3$ operations for the final result, which is computationally acceptable for large sets of polynomials.

### 6.4.2 Numerical behaviour of the Hybrid LCM algorithm

The numerical analysis of the Hybrid LCM algorithm mainly concerns the least-squares problem (6.53). The application of the ERES Division algorithm to the polynomials of the original set $\mathcal{P}$ involves symbolic-rational operations, which lead to the formulation of the initial matrix $F_{\mathcal{P}}$ without additional numerical error. In case where the input data are given inexactly in finite precision floating-point format, they may be converted to rational format introducing a negligible numerical error close enough to hardware precision.

In order to achieve a better computation of the singular values of $F_{\mathcal{P}}$, it is preferable to apply the SVD algorithm on a normalised copy of $F_{\mathcal{P}}$, such that all its elements be less than 1 in absolute value [27]. This process is an elementary row transformation which provides better numerical stability and does not affect the properties of the system $F_{\mathcal{P}} \cdot \underline{a} = \underline{0}$ and hence the final solution. The normalisation of the rows of the matrix $F_{\mathcal{P}}$ by using the Euclidean norm in floating-point arithmetic with unit round-off $\mathbf{u}$, satisfies the properties [18] :

$$\widetilde{F}_{\mathcal{P}} = N \cdot F_{\mathcal{P}} + E_N\ , \tag{6.60}$$

$$\|E_N\|_2 \leq d\,\mathbf{u}\,\|F_{\mathcal{P}}\|_2 + O(\mathbf{u}^2), \quad \|N\|_2 \leq \sqrt{d}$$

where $d = \sum_{i=1}^{h} \deg\{p_i(s)\}$, $N \in \mathbb{R}^{d\times d}$ is a diagonal matrix accounting for the performed transformations and $E_N \in \mathbb{R}^{d\times d+1}$ the error matrix. Therefore, if $r$ is the numerical rank of $\widetilde{F}_{\mathcal{P}}$, the matrix $\widehat{F}_{\mathcal{P}}$ in the least-squares problem (6.53) is actually formed from the first $r$ columns of $\widetilde{F}_{\mathcal{P}}$.

Let the perturbation $E_r$ of the matrix $\widehat{F}_{\mathcal{P}}$ be small enough, such that

$$\rho(\widehat{F}_{\mathcal{P}}) = \rho(\widehat{F}_{\mathcal{P}} + E_r)$$

According to Theorem 6.8 the unique least-squares solution for the problem (6.53), in general, depends upon the square of the condition number of the $d \times r$ matrix $\widehat{F}_{\mathcal{P}}$. If we compute the SVD of $\widehat{F}_{\mathcal{P}}$, then [27] :

$$\|\widehat{F}_{\mathcal{P}}\|_2 = \sigma_{max} \ , \quad \|\widehat{F}_{\mathcal{P}}^{\dagger}\|_2 = \frac{1}{\sigma_{min}}$$

where $\sigma_{max}$ and $\sigma_{min}$ are the maximum and minimum singular values of $\widehat{F}_{\mathcal{P}}$. Hence,

$$\text{Cond}(\widehat{F}_{\mathcal{P}}) = \|\widehat{F}_{\mathcal{P}}\|_2 \, \|\widehat{F}_{\mathcal{P}}^{\dagger}\|_2 = \frac{\sigma_{max}}{\sigma_{min}} \tag{6.61}$$

The residual of the least-squares solution also depends upon $\text{Cond}(\widehat{F}_{\mathcal{P}})$ when $\widehat{F}_{\mathcal{P}}$ is perturbed, as showed in Theorem 6.7. However, the results so far from the numerical sensitivity analysis of the linear least-squares problem show that condition numbers are connected with backward error estimations. Different methods lead to a variety of estimates for the optimal size of backward errors for least-squares problems [29]. Numerical tests [30] regarding least-squares problems have shown that the optimal backward errors for least-squares problems are much smaller – orders of magnitude smaller – than the solution errors and furthermore the QR method results in noticeably smaller solution errors than the SVD method. Regarding the backward error estimates, as the computed solution of the least-squares problem becomes more accurate, the estimate may become more difficult to evaluate accurately because of the unavoidable rounding error in forming the residual.

If QR-Householder factorisation is used in order to solve the full rank least-squares problem (6.53), then, taking into account the result given by Lawson and Hanson in [54], the computed solution $\underline{\hat{a}}$ is such that it satisfies

$$\left\| (\widehat{F}_{\mathcal{P}} + E_r)\, \underline{\hat{a}} - \left( \Delta f - \underline{f}_{r+1} \right) \right\|_2 = minimum$$

where

$$\|E_r\|_2 \leq (6d - 3r + 41) r \mathbf{u} \left\| \widehat{F}_{\mathcal{P}} \right\|_F + O(\mathbf{u}^2) \tag{6.62}$$

and

$$\|\Delta f\|_2 \leq (6d - 3r + 40) r \mathbf{u} \left\| \underline{f}_{r+1} \right\|_2 + O(\mathbf{u}^2) \tag{6.63}$$

and $\mathbf{u}$ is the machine's precision (hardware accuracy). The above inequalities show that $\underline{\hat{a}}$ satisfies a "nearby" least-squares problem. Numerically, trouble can be expected whenever $\text{Cond}(\widehat{F}_{\mathcal{P}}) \approx \mathbf{u}^{-1}$, [27].

Therefore, having a well conditioned matrix and considering the results in theorems 6.7–6.9, the least-squares solution of (6.53) is considered a good approximate solution to the LCM problem, provided that the corresponded residual is small enough according to a specified numerical tolerance $\varepsilon_t$. In most cases, the chosen numerical tolerance is quite satisfactory to be set equal to $\mathbf{u}\,\|F_{\mathcal{P}}\|_{\infty}$ .

### ▶ Computational examples

**Example 6.4.** Consider the set $\mathcal{P}_3 = \{p_i(s) \in \mathbb{R},\ i = 1, 2, 3\}$ with

$$\begin{aligned} p_1(s) &= (s+1)(s+2)^2 &&= s^3 + 5s^2 + 8s + 4 \\ p_2(s) &= (s+2)(s+3)(s+4) &&= s^3 + 9s^2 + 26s + 24 \\ p_3(s) &= (s+4)^2(s+5) &&= s^3 + 13s^2 + 56s + 80 \end{aligned} \tag{6.64}$$

The exact LCM of the set $\mathcal{P}_3$ is

$$\begin{aligned} l(s) &= (s+1)(s+2)^2(s+3)(s+4)^2(s+5) \\ &= s^7 + 21\,s^6 + 183\,s^5 + 855\,s^4 + 2304\,s^3 + 3564\,s^2 + 2912\,s + 960 \end{aligned} \tag{6.65}$$

and the $\gcd\{\mathcal{P}\} = 1$. We will compute the LCM of the set by using the Hybrid LCM Algorithm 6.2.

Assuming that we do not know in advance the actual degree of the LCM, we can represent the LCM regarding its maximum theoretical degree $d = 3+3+3 = 9$ with arbitrary coefficients in the form:

$$l(s) = a_9 s^9 + a_8 s^8 + a_7 s^7 + a_6 s^6 + a_5 s^5 + a_4 s^4 + a_3 s^3 + a_2 s^2 + a_1 s^1 + a_0$$

If we denote by

$$p(s) = s^3 + b_2 s^2 + b_1 s + b_0$$

an arbitrary polynomial of degree 3, which corresponds to the maximum degree polynomial of the set, then the next steps to be followed are:

- Apply the ERES Division Algorithm 3.1 to the polynomials $l(s)$ and $p(s)$ to obtain a symbolic algebraic formula for the remainder $r(s)$ of the division $\frac{l(s)}{p(s)}$.
- In this symbolic algebraic formula, substitute the terms $b_j,\ j = 0, 1, 2$ with the corresponding coefficients of the polynomials $p_i(s),\ i = 1, 2, 3$ and construct the $9 \times 10$ matrix $F_{\mathcal{P}}$ :

$$F_{\mathcal{P}} = \left[\begin{array}{ccc|ccccccc} 1 & 0 & 0 & -4 & 20 & -68 & 196 & -516 & 1284 & -3076 \\ 0 & 1 & 0 & -8 & 36 & -116 & 324 & -836 & 2052 & -4868 \\ 0 & 0 & 1 & -5 & 17 & -49 & 129 & -321 & 769 & -1793 \\ \hline 1 & 0 & 0 & -24 & 216 & -1320 & 6840 & -32424 & 145656 & -632040 \\ 0 & 1 & 0 & -26 & 210 & -1214 & 6090 & -28286 & 125370 & -539054 \\ 0 & 0 & 1 & -9 & 55 & -285 & 1351 & -6069 & 26335 & -111645 \\ \hline 1 & 0 & 0 & -80 & 1040 & -9040 & 65680 & -430800 & 2645520 & -15521360 \\ 0 & 1 & 0 & -56 & 648 & -5288 & 36936 & -235880 & 1421064 & -8219432 \\ 0 & 0 & 1 & -13 & 113 & -821 & 5385 & -33069 & 194017 & -1101157 \end{array}\right]$$

- Normalize the rows of $F_{\mathcal{P}}$ by using the Euclidean norm and compute its rank. A numerical accuracy $\varepsilon_t = 10^{-16} \approx \mathbf{u}\,\|F_{\mathcal{P}}\|_\infty$ can be set, and the numerical $\varepsilon_t$-rank of the normalised $F_{\mathcal{P}}$, denoted by $\widetilde{F}_{\mathcal{P}}$, is $r = 7$. Therefore, the aim is to find an LCM with degree equal to 7.

- If $\widehat{F}_{\mathcal{P}}$ is the $9 \times 7$ matrix, which derives from $\widetilde{F}_{\mathcal{P}}$ by deleting its last two columns, the next overdetermined linear system has to be solved:

$$\widehat{F}_{\mathcal{P}} \cdot \underline{\hat{a}} = -\underline{f}_{r+1}$$

  where $\underline{f}_{r+1}$ is the $8^{th}$ column of $\widetilde{F}_{\mathcal{P}}$. The condition number of $\widehat{F}_{\mathcal{P}}$ is

$$\mathrm{Cond}(\widehat{F}_{\mathcal{P}}) = 5.283454 \cdot 10^7$$

- Proceed with the solution of the least-squares problem:

$$\min_{\underline{\hat{a}}} \left\| \widehat{F}_{\mathcal{P}} \cdot \underline{\hat{a}} + \underline{f}_{r+1} \right\|_2 \tag{6.66}$$

In the least-squares problem (6.66) we applied three different methods using QR factorization (LS-QR), singular value decomposition (LS-SVD), and finding the pseudo-inverse of $\widehat{F}_{\mathcal{P}}$ (LS-PInv). The quality of the obtained solution is measured regarding the magnitude of the residual

$$\|u\|_2 = \left\| \widehat{F}_{\mathcal{P}} \cdot \underline{\hat{a}} + \underline{f}_{r+1} \right\|_2$$

and the relative error

$$\mathrm{Rel} = \frac{\|\underline{\hat{a}} - \underline{a}\|_2}{\|\underline{a}\|_2}$$

since we know the exact solution $\underline{a}$. These results are presented in the next table.

| | LS-QR | LS-SVD | LS-PInv |
|---|---|---|---|
| Residual | $6.255761 \cdot 10^{-15}$ | $1.533661 \cdot 10^{-11}$ | $8.876500 \cdot 10^{-12}$ |
| Relative error | $4.641785 \cdot 10^{-13}$ | $1.535942 \cdot 10^{-11}$ | $1.405633 \cdot 10^{-11}$ |

Table 6.1: Results for the approximate LCM of the set $\mathcal{P}_3$ in Example 6.5 given by different least-squares methods.

Obviously, the QR method gives a more accurate solution than the other two least-squares methods. □

**Example 6.5.** Consider the same set of three polynomials from above adding a small perturbation $\varepsilon = 10^{-7}$,

$$\mathcal{P}'_3 = \left\{ \begin{array}{l} p_1(s) = (s+1)(s+2+\varepsilon)^2 \\ p_2(s) = (s+2)(s+3)(s+4+\varepsilon) \\ p_3(s) = (s+4)^2(s+5) \end{array} \right\} \tag{6.67}$$

The degree of the exact LCM of the set $\mathcal{P}'_3$ is equal to 9 and $\gcd\{\mathcal{P}'_3\} = 1$. We will attempt to find an approximate LCM of the set $\mathcal{P}'_3$ by using again the Hybrid LCM Algorithm 6.2.

The rank of the $9 \times 10$ normalized input matrix $\widetilde{F}_{\mathcal{P}}$, computed in 34-digits numerical precision (quadruple precision), is 9, as expected. However, if we set a numerical tolerance $\varepsilon_t = 10^{-8} = 0.1\,\varepsilon$, then its rank drops to 7. Indeed, the singular values of $\widetilde{F}_{\mathcal{P}}$ in quadruple precision are:

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \\ \sigma_7 \\ \sigma_8 \\ \sigma_9 \end{bmatrix} = \begin{bmatrix} 2.9802039976570089550434545038338 \\ 0.34366973145489844885746482429270 \\ 0.01646184489225013854776060403632 8 \\ 0.00204571795979518531250960427855 47 \\ 0.00026540982252781169619909257850821 \\ 0.00001699983326532030038263900685023 3 \\ \mathbf{5.52078476535294390888618848295 65 \cdot 10^{-7}} \\ 3.34781904877851462387462017484444 \cdot 10^{-22} \\ 3.56054763972340415822230187561 45 \cdot 10^{-24} \end{bmatrix}$$

Obviously, the $7^{th}$ singular value $\sigma_7$ is greater than $\varepsilon_t = 10^{-8}$, which implies that the numerical $\varepsilon_t$-rank of $\widetilde{F}_{\mathcal{P}}$ is 7, and hence the degree of the obtained approximate

$\varepsilon_t$-LCM is 7. Indeed, the solution of the corresponding least-squares problem is a real polynomial with degree 7 and the residual is equal to $1.488148 \cdot 10^{-10}$.

If we try now to compute an approximate LCM by using the S-R LCM Algorithm 6.1, we will notice that the LCM of the set $\mathcal{P}'_3$ is a polynomial with degree 9, which implies that the associated GCD is equal to 1. However, for a numerical tolerance $\varepsilon_t = 10^{-4}$ the Hybrid ERES algorithm is able to give a non trivial GCD of degree 2. Then, the obtained solution is an approximate $\varepsilon_t$-LCM of degree 7 with residual (remainder norm $\|u\|_2$) equal to $1.296138 \cdot 10^{-2}$. If we denote by $l_1(s)$ the approximate solution given by the Hybrid LCM Algorithm 6.2 and $l_2(s)$ the approximate solution given by the S-R LCM Algorithm 6.1, then the distance between these two approximations is

$$\|l_1(s) - l_2(s)\|_2 = 0.4057382668$$

The two methods gave us results with great numerical difference and, judging from the residuals in Table 6.2, we conclude that the Hybrid LCM algorithm produced a more reliable approximate solution.

| | $\varepsilon_t$ | Degree | Residual |
|---|---|---|---|
| S-R LCM alg. | $10^{-4}$ | 7 | $1.296138 \cdot 10^{-2}$ |
| Hybrid LCM alg. | $10^{-8}$ | 7 | $1.488148 \cdot 10^{-10}$ |

Table 6.2: Numerical difference between the result from the S-R LCM and Hybrid LCM algorithms for the set $\mathcal{P}'_3$ in Example 6.5.

□

**Example 6.6.** The following polynomial set contains three real polynomials in one variable:

$$\mathcal{P} = \left\{ \begin{array}{l} p_1(s) = s^2 - 5\,s + 6 \\ p_2(s) = s^2 - (5 - \varepsilon_1)\,s + 6 \\ p_3(s) = s - (2 - \varepsilon_2) \end{array} \right\} \tag{6.68}$$

The coefficients of the polynomials of set $\mathcal{P}$ are perturbed by the parameters $\varepsilon_1$, $\varepsilon_2$, which are small positive numbers taking values from $10^{-1}$ to $10^{-15}$.

Considering the exact coefficients of the polynomials when $\varepsilon_1 = \varepsilon_2 = 0$, the LCM of the set is $l(s) = s^2 - 5\,s + 6$. However, if the coefficients of the polynomials become inexact ($\varepsilon_1 \neq \varepsilon_2 > 0$), we have several LCMs whose degrees vary from 2 to 5. These LCMs depend strongly on the selection of $\varepsilon_1$, $\varepsilon_2$ and the specified numerical tolerance $\varepsilon_t$. The next test gives us a picture of the sensitivity of the LCM to small perturbations in the coefficients of the polynomials of the given set.

For each value of $\varepsilon_1$ and $\varepsilon_2$ from $10^{-1}$ to $10^{-15}$, we applied the Hybrid LCM algorithm to the given set $\mathcal{P}$. The numerical accuracy $\varepsilon_t$ was set equal to $0.5 \cdot 10^{-15}$

to simulate the standard hardware precision in Maple (Digits=16). For all the pairs $(\varepsilon_1, \varepsilon_2)$ there have been 225 approximate LCMs and their degrees are shown in the following table:

| (i,j) | -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 | -9 | -10 | -11 | -12 | -13 | -14 | -15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| -1 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 4 | 4 | 4 | 4 | 4 |
| -2 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| -3 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| -4 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| -5 | 5 | 5 | 5 | 5 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| -6 | 5 | 5 | 5 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| -7 | 5 | 5 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| -8 | 5 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| -9 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| -10 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| -11 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| -12 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 3 | 3 |
| -13 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| -14 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 2 |
| -15 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 2 |

Table 6.3: LCM degrees for the set $\mathcal{P}$ in Example 6.6.

Every entry $(i, j)$ in the above table represents the degree of the approximate LCM of the set $\mathcal{P}$ for $(\varepsilon_1, \varepsilon_2) = (10^i, 10^j)$, $i, j = -1, -2, \ldots, -15$. For example, the degree of the LCM for $(\varepsilon_1, \varepsilon_2) = (10^{-2}, 10^{-3})$ is 5. As we can see, the majority of the obtained LCMs – 132 polynomials – have a degree equal to 4 . The variation of the degrees is also presented in Figure 6.1.

The obtained results from the least-squares minimisation process characterise the quality of the given approximate LCMs and have showed that

a) the maximum residual is $3.3450 \cdot 10^{-14}$, the minimum residual is $0.6037 \cdot 10^{-16}$ and they correspond to LCMs of degree 4 and 2 respectively,

b) the "best" LCMs with minimal residual have degrees 3 or 2. However we have a lot of "good" LCMs of degree 4, and

c) when $\varepsilon_1 = 10^{-i}$, $\varepsilon_2 = 10^{-j}$ for $i = 1, 2, \ldots 11$ and $j = 12 - i$, the obtained $4^{th}$ degree LCMs have the worst residuals. (The white band in Figure 6.2 illustrate these values.)

The "best" LCMs for each degree from 2 to 5 are shown in the next table:

| | | |
|---|---|---|
| Degree | = | 2 |
| $(\varepsilon_1, \varepsilon_2)$ | = | $(10^{-15}, 10^{-14})$ |
| Residual | = | $6.037065380953120 \cdot 10^{-17}$ |
| LCM | = | $s^2 - 4.999999999999998\, s + 5.999999999999985$ |
| Degree | = | 3 |
| $(\varepsilon_1, \varepsilon_2)$ | = | $(10^{-15}, 10^{-6})$ |
| Residual | = | $1.673712805750593 \cdot 10^{-16}$ |
| LCM | = | $s^3 - 6.999998946115123\, s^2 + 15.99999473057559\, s$ |
| | | $-11.99999367669075$ |
| Degree | = | 4 |
| $(\varepsilon_1, \varepsilon_2)$ | = | $(10^{-3}, 10^{-13})$ |
| Residual | = | $2.925394959602412 \cdot 10^{-16}$ |
| LCM | = | $s^4 - 9.999000000011530\, s^3 + 36.99500000009273\, s^2$ |
| | | $-59.99400000024462\, s + 36.00000000021050$ |
| Degree | = | 5 |
| $(\varepsilon_1, \varepsilon_2)$ | = | $(10^{-8}, 10^{-1})$ |
| Residual | = | $1.897380776005375 \cdot 10^{-15}$ |
| LCM | = | $s^5 - 11.89972320221808\, s^4 + 55.99724156156302\, s^3$ |
| | | $-130.2898547515662\, s^2 + 149.9836922323088\, s$ |
| | | $-68.39032671536748$ |

Table 6.4: Best LCMs for the set $\mathcal{P}$ in Example 6.6.

□

## 6.5 Discussion

In this chapter the problem of computing the LCM of sets of several polynomials in $\mathbb{R}[s]$ has been considered. The study was focused on two different LCM methods where the ERES methodology is involved. The first LCM method, originally developed in [47], is based on the associativity property of the LCM and therefore it requires the computation of the GCD. In the current approach the computation of the associated GCD was carried out by the Hybrid ERES algorithm and the LCM is given as a result provided from the ERES Division algorithm. The developed symbolic LCM method provides excellent results when the polynomials have exactly known coefficients. In the approximate case, the provided solution relies on the existence and use of an approximate GCD. However, when several polynomials are involved, the complexity of this method rises to high levels and this can cause serious complications when an approximate LCM is required.

The second method which is developed here, relies on the principal that

every polynomial of a given set must divide evenly into the LCM and does not require the computation of the GCD. With the aid of the ERES Division method, a remainder sequence is formed which leads to the formulation of an homogeneous underdetermined linear system. The rank of the particular system of linear equations corresponds to the degree of the LCM and the coefficients of the LCM are given by the unique solution of a reduced full rank overdetermined linear system. When exact data are used, the accuracy of the produced LCM depends on the numerical method that is used to solve the final linear system, such as LU factorisation, Gaussian elimination or QR factorisation [18]. When the data are given in symbolic-rational format, the LCM is produced in great accuracy. However, when numerical inaccuracies are present, the final linear system is transformed to an appropriate minimisation problem and then, an approximate LCM is produced. Therefore, the approximate LCM problem can be also considered as an optimisation problem. In the present study, linear least-squares methods have been utilized to obtain approximate LCMs and their quality is characterised by the residual of the least-squares solution. The derived algorithm for the computation of the approximate LCM, referred to as the Hybrid LCM algorithm, combines pure symbolic and numerical finite precision computations in order to form and solve the linear system which provides the coefficients of the approximate LCM of a given set of several real univariate polynomials.

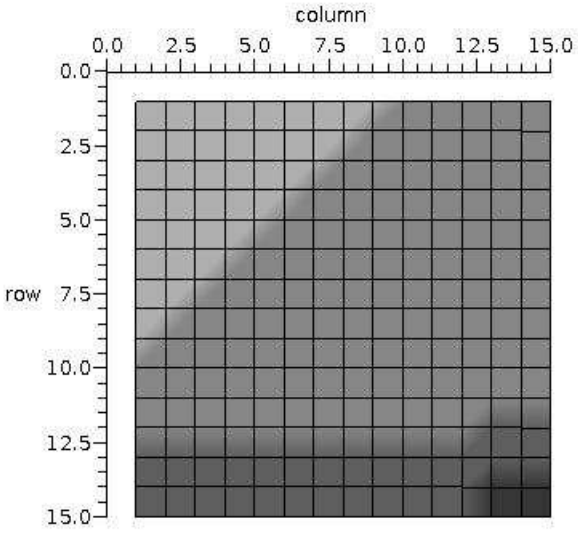


Figure 6.1: LCM degrees of the polynomial set $\mathcal{P}$ in Example 6.6. Darker areas correspond to approximate LCMs with lower degrees as the values of the parameters $\varepsilon_1$ and $\varepsilon_2$ decrease.
`row` : $\varepsilon_1 = 10^{-i}$ `column` : $\varepsilon_2 = 10^{-j}$

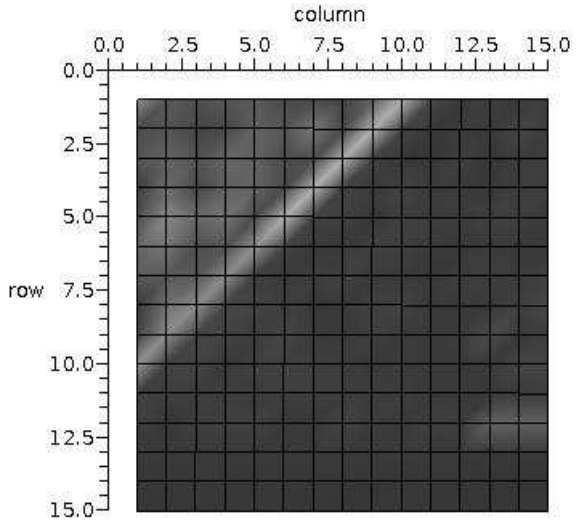


Figure 6.2: LCM residuals of the polynomial set $\mathcal{P}$ in Example 6.6. Darker areas correspond to approximate LCMs with smaller residuals as the values of the parameters $\varepsilon_1$ and $\varepsilon_2$ decrease.
`row` : $\varepsilon_1 = 10^{-i}$ `column` : $\varepsilon_2 = 10^{-j}$

# Chapter 7

# Stability evaluation for linear systems using the ERES method

## 7.1 Introduction

Stability is a basic requirement for the design of a control system. The notion of stability of systems can be defined in many ways depending on the type of the system. *Lyapunov stability* and *input-output stability* are two of the most important stability types for dynamical systems.

A qualitative characterisation of dynamical systems is the expectation that bounded system inputs will result in bounded system outputs. System properties of this type are referred to as *bounded input - bounded output (BIBO) stability*. A linear system is said to be BIBO stable if and only if for any bounded input vector $u(t)$, the output vector $y(t)$ is bounded[1]. For single input - single output (SISO) linear systems, BIBO stability implies that for every input $u(t)$ and $k_1$ constant such that $|u(t)| \leq k_1 < \infty$ , $\forall\, t \geq 0$, there exists $k_2$ constant such that the output $y(t)$ satisfies $|y(t)| \leq k_2 < \infty$ , $\forall\, t \geq 0$. Several different conditions for BIBO stability can be established for different types of systems such as time-invariant, time-varying, continuous-time, and finite-dimensional systems [1, 38].

Stability for nonlinear systems basically relies on the notion of *Lyapunov stability* [1, 38]. The notion of Lyapunov stability occurs in the study of dynamical systems and it is associated with internal, or state space descriptions. In simple terms, if all solutions of the dynamical system that start out near an equilibrium point $x_e$ stay near $x_e$ forever, then $x_e$ is *Lyapunov stable*. The idea of Lyapunov stability can be extended to infinite-dimensional manifolds, where it is known as *structural stability*, which concerns the behaviour of different but "nearby" solutions to differential equations. Specialized Lyapunov's stability criteria can be applied to linear systems as well, but there are also other important criteria based on algebraic and geometric properties [1]. Amongst the most important

[1] A vector is bounded if every component is bounded.

stability criteria for linear systems, the *Routh-Hurwitz criterion* is distinguished for its theoretical and practical value in stability analysis. The name refers to E. J. Routh and A. Hurwitz who contributed to the formulation of this criterion dating from the end of the 19th century [37, 66].

The Routh-Hurwitz stability criterion is a necessary and sufficient method to establish the stability of a SISO, linear time-invariant (LTI) control system [1, 24]. Generally, given the characteristic polynomial of a linear system, some calculations using only the coefficients of that polynomial can lead to the conclusion that it is stable or unstable. The criterion can be performed using either polynomial divisions or determinant calculus. In this chapter, a new approach to the problem of evaluating the stability of a linear system is presented. This particular approach is based on the combination of the Routh-Hurwitz criterion and the ERES method in order to form a matrix-based criterion for the evaluation of the stability of an LTI system. The developed method involves the computation of the coefficients that are necessary to form a continued fraction representation for the characteristic polynomial of a linear system. Then, the stability of the linear system is deduced from the distribution of the roots of the characteristic polynomial on the complex plane, which is determined by the signs of these coefficients. The main result of this study is the formulation of the RH-ERES algorithm. This algorithm computes the coefficients of the continued fraction representation of the characteristic polynomial, either symbolically, or numerically and faster than Routh's algorithm [24, 66]. The problem of finding the minimum distance of an unstable to a stable polynomial as well as the minimum norm stabilization are also addressed.

*NOTATION* 7.1. $\mathbb{C}^+ = \{s \in \mathbb{C} : \Re(s) > 0\}$ denotes the *right half-plane* where the complex numbers have positive real parts. Similarly, $\mathbb{C}^- = \{s \in \mathbb{C} : \Re(s) < 0\}$ denotes the *left half-plane* where the complex numbers have negative real parts. ($\Re(s)$ denotes the real part of a complex number $s \in \mathbb{C}$.)

## 7.2 Stability of linear systems and the Routh-Hurwitz criterion

In this section we provide some theoretical background material regarding the stability of linear systems and the Routh-Hurwitz criterion, which can be found in [1, 24, 38]. Specifically, we shall concern ourselves with a simple type of linear system of $n^{th}$-order linear homogeneous ordinary differential equations of the form:

$$c_n x^{(n)} + c_{n-1} x^{(n-1)} + \ldots + c_1 x^{(1)} + c_0 x = 0, \quad c_n \neq 0 \tag{7.1}$$

where the coefficients $c_0, \ldots, c_n$ are all constant real numbers. Usually, $x$ is time dependent and if $t$ represents time, the solution of the system (7.1) is actually

a vector $x(t) \in \mathbb{R}^n$ (state-vector). A linear system that does not depend on the variable $t$ is a *Linear Time-Invariant* system (LTI). This type of linear systems is very common in the area of applied mathematics with direct applications in control theory, signal processing, circuits, seismology, and others.

The algebraic equation (7.1) is equivalent to the system of first-order ordinary differential equations

$$\mathcal{S}: \ \dot{x} = Ax \tag{7.2}$$

if we develop an appropriate state-space realisation for (7.1) [1]. In the realisation (7.2), $x = x(t) \in \mathbb{R}^n$ and $A$ denotes the companion matrix given by:

$$A = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -\frac{c_0}{c_n} & -\frac{c_1}{c_n} & -\frac{c_2}{c_n} & \dots & -\frac{c_{n-1}}{c_n} \end{bmatrix} \in \mathbb{R}^{n \times n} \tag{7.3}$$

The notion of an *equilibrium* is critical in the stability analysis of linear systems.

**Definition 7.1.** Given a system of first-order differential equations $\dot{x} = \mathcal{F}(t, x)$, $x \in \mathbb{R}^n$, a point $x_e \in \mathbb{R}^n$ is called an *equilibrium point* of the system (or simply an equilibrium) at time $t_0 > 0$, if $\mathcal{F}(t, x_0) = 0$ for all $t > t_0$.

There are several qualitative characterisations of an equilibrium point that are of fundamental importance in systems theory. These characterizations are concerned with various types of stability properties of an equilibrium and are referred to in the literature as *Lyapunov stability*. In general, an equilibrium point is:

- *stable* (*uniformly stable*), if for initial conditions that start near an equilibrium point the resulting trajectory stays near that equilibrium point,
- *asymptotically stable*, if it is stable and, in addition, the state trajectory of the system converges to the equilibrium point as time tends to infinity,
- *unstable*, if it is not stable.

The linear time-invariant, homogeneous system of ordinary differential equations $\mathcal{S}$ in (7.2), has a unique equilibrium that is at the origin if and only if $A$ is nonsingular. Otherwise, $\mathcal{S}$ has nondenumerably many equilibria. Therefore, the most interesting equilibrium for $\mathcal{S}$ is located at the origin where $x = 0$ and will be referred to as the *trivial solution* of the system $\mathcal{S}$.

In order to determine whether the equilibrium $x = 0$ of $\mathcal{S}$ is asymptotically stable, it suffices to determine if all the eigenvalues $\lambda$ of the matrix $A$ have negative real parts, or what amounts to the same thing, if the roots of the polynomial

$$f(\lambda) = c_n\lambda^n + c_{n-1}\lambda^{n-1} + \ldots + c_1\lambda + c_0 \tag{7.4}$$

all have negative real parts. To see this, we must show that the eigenvalues $\lambda$ of $A$ coincide with the roots of the polynomial $f(\lambda)$. This is most easily accomplished by induction.

For the first-order case $k = 1$, we have $A = -\dfrac{c_0}{c_n}$ and therefore,

$$\det(\lambda I_1 - A) = \lambda + \frac{c_0}{c_n}$$

where $I_1 = 1$, and so the assertion is true for $k = 1$. Next, assume that the assertion is true for $k = n - 1$. Then

$$\begin{aligned} \det(\lambda I_n - A) &= \lambda \det(\lambda I_{n-1} - A_1) + \frac{c_0}{c_n} \\ &= \lambda^n + \frac{c_{n-1}}{c_n}\lambda^{n-1} + \ldots + \frac{c_1}{c_n}\lambda + \frac{c_0}{c_n} = 0 \end{aligned} \tag{7.5}$$

where $I_k$ is the $k \times k$ identity matrix for $k = n, n-1$ and

$$A_1 = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \ldots & 1 \\ -\dfrac{c_1}{c_n} & -\dfrac{c_2}{c_n} & -\dfrac{c_3}{c_n} & \ldots & -\dfrac{c_{n-1}}{c_n} \end{bmatrix} \in \mathbb{R}^{(n-1)\times(n-1)}$$

Clearly, (7.5) is equivalent to $f(\lambda) = 0$. The polynomial $\phi(\lambda) \triangleq \det(\lambda I_n - A)$ is also known as the *characteristic polynomial* of the matrix $A$. The most important stability criteria contain algebraic conditions, which are formed form the coefficients of the characteristic polynomial $\phi(\lambda)$. The stability of a linear system can be assessed by the distribution of the roots of its characteristic polynomial $\phi(\lambda)$ in the imaginary plane.

**Definition 7.2** ([1])**.** A real $n \times n$ matrix $A$ is called:

a) *stable* or a *Hurwitz matrix*, if all its eigenvalues have negative real parts,

b) *unstable*, if at least one of the eigenvalues has positive real part and

c) *critical*, when it is neither stable nor unstable.

Analogously to matrices, an $n^{th}$-order polynomial $f(s)$ with real coefficients is called:

a) *stable*, if all the roots of $f(s)$ have negative real parts,

b) *unstable*, if at least one of the roots of $f(s)$ has a positive real part.

*REMARK* 7.1. A stable polynomial, which has all its roots in the left half-plane, is also called a *Hurwitz polynomial* or *Hurwitzian.*

*REMARK* 7.2. If $x(t)$ is the input to a general linear time-invariant system, and $y(t)$ is the output, and the Laplace transform of $x(t)$ and $y(t)$ be $\hat{x}(s) = \mathcal{L}\{x(t)\}$ and $\hat{y}(s) = \mathcal{L}\{y(t)\}$ respectively, then the output $\hat{y}(s)$ is related to the input $\hat{x}(s)$ by the *transfer function* $H(s)$ as

$$\hat{y}(s) = H(s) \cdot \hat{x}(s)$$

Considering a linear time-invariant SISO system $\mathcal{S}$, as in (7.2), and its transfer function $H(s)$, which represents the ratio of the output $\hat{y}(s)$ of the system to the input $\hat{x}(s)$ of the system, the stability analysis shows that the poles of the transfer function $H(s)$ are in general a subset of the eigenvalues of $A$. Thus, it is sufficient to check the poles of the transfer function $H(s)$ of the system, that is the roots $s_i$ of its characteristic equation:

$$f(s) = c_n\, s^n + c_{n-1}\, s^{n-1} + c_{n-2}\, s^{n-2} + \ldots + c_1\, s + c_0 = 0 \tag{7.6}$$

The following necessary and sufficient stability conditions can be formulated:

- A linear system $\mathcal{S}$ is *asymptotically stable*, if the real part of the roots of $f(s)$ is
$$\Re(s_i) < 0, \;\; \forall\, s_i,\; i = 1, 2, \ldots, n$$
or, in other words, if all poles of $H(s)$ lie in the left half-plane.

- A linear system $\mathcal{S}$ is *unstable*, if at least one pole of $H(s)$ lies in the right half-plane, or, if at least one multiple pole is on the imaginary axis of the complex plane.

For systems with eigenvalues having zero real-part, stability is determined by using the Jordan normal form associated with the matrix. A system with eigenvalues that have no strictly positive real part is stable, if and only if the Jordan block corresponding to each eigenvalue with zero part is a scalar $(1 \times 1)$ block.

The number of the roots of the characteristic polynomial that lies in the left half-plane of the field of complex numbers is used as a criterion for checking the stability of an LTI system, known as the *Routh-Hurwitz stability criterion.* The

criterion is related to *Routh-Hurwitz theorem* which derives from the use of the Euclidean algorithm and Sturm's theorem in evaluating Cauchy indices.

**Definition 7.3.** A polynomial $f \in \mathbb{R}[s]$ is *square-free*, if and only if for every nonconstant $g \in \mathbb{R}[s]$, $g^2$ does not divide $f$.

*REMARK* 7.3. A square-free polynomial is a polynomial with no multiple roots. Let us consider a polynomial $p \in \mathbb{R}[s]$ of degree $n > 1$ and its first derivative $p'$. The following are equivalent [5]:

i) The polynomial $p$ is square-free, if $\gcd\{p, p'\} = 1$.

ii) Let $S_{\{p,p'\}}$ the resultant matrix of $p$ and $p'$. If $\rho\left(S_{\{p,p'\}}\right) = 2n - 1$, then $p$ is a square-free polynomial.

**Definition 7.4.** Given a real univariate polynomial $p(s)$, a *Sturm chain* or *Sturm sequence* is a finite sequence of polynomials $\mathcal{P}_s = \{p_0(s), p_1(s), \ldots, p_m(s)\}$ of decreasing degree with the following properties:

a) $p_0(s) := p(s)$ is square-free and $p_1(s) := p'(s)$,
where $p'(s)$ is the first derivative of the polynomial $p(s)$,

b) $p_i(s) = -rem(p_{i-1}, p_i),\ i = 2, 3, \ldots, m$,
where $rem\{\cdot\}$ denotes the remainder of the Euclidean division of two polynomials,

c) if $p(s_0) = 0$, then $sign(p_1(s_0)) = sign(p'(s_0))$,

d) if $p_i(s_0) = 0$ for $0 < i < m$ then $sign(p_{i-1}(s_0)) = -sign(p_{i+1}(s_0))$,

e) $p_m(s)$ does not change its sign.

**Theorem 7.1** (Sturm's theorem [24])**.** *Let $\mathcal{P}_s = \{p_0(s), p_1(s), \ldots, p_m(s)\}$ be a Sturm chain, where $p(s)$ is a real univariate square-free polynomial. If $V(s_0)$ denotes the number of sign changes in the sequence $\mathcal{P}_{s_0} = \{p_0(s_0), p_1(s_0), \ldots, p_m(s_0)\}$ for a specific $s_0$, then for two real numbers $a < b$, the number of distinct real roots of $p(s)$ in the interval $(a, b)$ is*

$$V(a) - V(b) \tag{7.7}$$

*REMARK* 7.4. If $p(s)$ is square-free, it shares no roots with its derivative $p'(s)$, hence $p_m(s)$ will be a nonzero constant polynomial. If $p(s)$ is not square-free, the derived sequence does not formally satisfy the definition of a Sturm chain above, nevertheless it still satisfies the conclusion of Sturm's theorem.

**Definition 7.5** (Cauchy index [24])**.** The Cauchy index of a rational function $R(s)$ between the limits $a$ and $b$ denoted by $I_a^b R(s)$, ($a$ and $b$ can be real numbers or $\pm\infty$ ) is the difference between the number of jumps of $R(s)$ from $-\infty$ to $+\infty$ and that of jumps from $+\infty$ to $-\infty$ as the argument changes from $a$ to $b$.

More specifically, a real polynomial $f(s)$ as in (7.6), can be written in the form:

$$f(s) = f_0(s) + f_1(s) \tag{7.8}$$

where

$$\begin{aligned} f_0(s) &= c_n\, s^n + c_{n-2}\, s^{n-2} + c_{n-4}\, s^{n-4} + \dots \\ f_1(s) &= c_{n-1}\, s^{n-1} + c_{n-3}\, s^{n-3} + c_{n-5}\, s^{n-5} + \dots \end{aligned}$$

The equation (7.8) implies the splitting of $f(s)$ in odd and even powers of the variable $s$. Then, the Cauchy index of the rational function

$$R(s) = \frac{f_0(s)}{f_1(s)} \tag{7.9}$$

over the real line is the difference between the number of roots of $f(s)$ located in the right half-plane and those located in the left half-plane. In the regular case, the polynomials $f_0(s)$ and $f_1(s)$ are coprime and hence, $R(s)$ is irreducible. Other special cases are analysed in [24] and will be discussed in the sequel.

Considering the irreducible rational function $R(s)$, we can construct a sequence of $n$ rational functions by using Euclidean division, such that

$$\begin{cases} R_k(s) \triangleq \dfrac{f_k(s)}{f_{k+1}(s)} = q_{k+1}(s) + \dfrac{f_{k+2}(s)}{f_{k+1}(s)}, \quad \text{for } k = 0, 1, \dots, n-2, n-1 \\ \\ \text{with} \quad f_n(s) = c_0, \quad f_{n+1}(s) = 0 \end{cases} \tag{7.10}$$

Sturm's theorem suggests a method of computing the Cauchy index of $R(s)$. The sequence $\{R_0(s), R_1(s), \dots, R_{n-1}(s)\}$ may be seen as a generalized Sturm chain and the Cauchy index is evaluated by the number of sign variations in the chain which eventually determines the number of the roots of $f(s)$ with negative real parts. Therefore, if we denote by:

i) $r_f^-$ and $r_f^+$ the number of roots of $f(s)$ in the left half-plane and the right half-plane, respectively, taking into account multiplicities,

ii) $V(s)$ the number of sign variations of the generalized Sturm chain $\{R_0(s), R_1(s), \dots, R_{n-1}(s)\}$, and

iii) $I_{-\infty}^{+\infty} R(s)$ the Cauchy index of the rational function $R(s)$ over the real line,

then, when we apply Sturm's theorem in the interval $(-\infty, +\infty)$ to $R(s)$ and by using (7.10), we obtain:

$$r_f^- - r_f^+ = \begin{cases} +I_{-\infty}^{+\infty} \frac{f_0(s)}{f_1(s)} & \text{, for odd degree } n \\ -I_{-\infty}^{+\infty} \frac{f_1(s)}{f_0(s)} & \text{, for even degree } n \end{cases} = V(+\infty) - V(-\infty) \qquad (7.11)$$

By the fundamental theorem of algebra, each polynomial of degree $n$ must have $n$ roots in the complex plane (i.e., for a polynomial with no roots on the imaginary line, $r_f^- + r_f^+ = n$). Thus, we have the condition that

$$f(s) \textit{ is a Hurwitz polynomial if and only if } r_f^- - r_f^+ = n.$$

Using Routh-Hurwitz theorem, the condition on $r_f^-$ and $r_f^+$ can be replaced by a condition on the generalized Sturm chain, which will give in turn a condition on the coefficients of $f(s)$.

**Theorem 7.2** (Routh-Hurwitz [24, 37])**.** *The number of roots of the real polynomial*

$$f(s) = c_n s^n + c_{n-1} s^{n-1} + \ldots + c_1 s + c_0, \quad c_n \neq 0 \qquad (7.12)$$

*in the right half-plane is determined by the formula:*

$$r_f^+ = V\left(c_n, \Delta_1, \frac{\Delta_2}{\Delta_1}, \frac{\Delta_3}{\Delta_2}, \ldots, \frac{\Delta_n}{\Delta_{n-1}}\right) \qquad (7.13)$$

An elaborate proof of the above theorem can be found in [24]. The terms $\Delta_i$, $i = 1, 2, \ldots, n$ are called *Hurwitz determinants* and we denote them by:

$$\Delta_1 = c_{n-1}, \quad \Delta_2 = \det \begin{pmatrix} c_{n-1} & c_{n-3} \\ c_n & c_{n-2} \end{pmatrix}, \ldots, \ \Delta_n = \det \begin{pmatrix} c_{n-1} & c_{n-3} & \ldots & c_1 \\ c_n & c_{n-2} & \ldots & c_0 \\ 0 & c_{n-3} & \ldots & c_3 \\ 0 & c_{n-2} & \ldots & c_2 \\ \ldots & \ldots & \ldots & \ldots \end{pmatrix}$$

*REMARK* 7.5. This statement of the Routh-Hurwitz theorem assumes that we have the regular case, where $\Delta_i \neq 0, \ \forall\, i = 1, 2, \ldots, n$.

Considering the above, the stability problem for $n^{th}$-order differential equations with constant coefficients has been reduced to a purely algebraic problem of determining whether the zeros of a polynomial all have negative real parts. A method for extracting information about the roots of a polynomial without solving for the roots relies on the *Routh array* [24, 66]. The Routh array is a tabular procedure for determining how many roots of a polynomial are in the right half of the complex plane. Given a polynomial $f(s)$ with real constant coefficients as defined in (7.12), the Routh array has the following tabular form:

$$\begin{array}{c|ccccc} s^n & c_n & c_{n-2} & c_{n-4} & c_{n-6} & \dots \\ s^{n-1} & c_{n-1} & c_{n-3} & c_{n-5} & c_{n-7} & \dots \\ \hline s^{n-2} & b_{n-2,1} & b_{n-2,2} & b_{n-2,3} & \dots & \\ s^{n-3} & b_{n-3,1} & b_{n-3,2} & b_{n-3,3} & \dots & \\ \vdots & \vdots & & & & \\ s^1 & b_{1,1} & & & & \\ s^0 & b_{0,1} & & & & \end{array} \tag{7.14}$$

where

$$b_{n-2,1} = \tfrac{-1}{c_{n-1}} \det \begin{pmatrix} c_n & c_{n-2} \\ c_{n-1} & c_{n-3} \end{pmatrix}, \qquad b_{n-2,2} = \tfrac{-1}{c_{n-1}} \det \begin{pmatrix} c_n & c_{n-4} \\ c_{n-1} & c_{n-5} \end{pmatrix}, \dots$$

$$b_{n-3,1} = \tfrac{-1}{b_{n-2,1}} \det \begin{pmatrix} c_{n-1} & c_{n-3} \\ b_{n-2,1} & b_{n-2,2} \end{pmatrix}, \quad b_{n-3,2} = \tfrac{-1}{b_{n-2,1}} \det \begin{pmatrix} c_{n-1} & c_{n-5} \\ b_{n-2,1} & b_{n-2,3} \end{pmatrix}, \dots$$

Two necessary, but not sufficient, conditions that all the roots of $f(s)$ have negative real parts are:

a) All the polynomial coefficients $c_i$ must have the same sign.

b) All the polynomial coefficients $c_i$ must be nonzero.

But the following theorem establishes necessary and sufficient conditions for a polynomial $f(s)$ to be a Hurwitzian, i.e. to have all its roots located in the left half-plane.

**Theorem 7.3** (Routh-Hurwitz stability criterion [24])**.** *The real polynomial*

$$f(s) = c_n s^n + c_{n-1} s^{n-1} + \ldots + c_1 s + c_0, \quad c_n \neq 0 \tag{7.15}$$

*is a Hurwitz polynomial (stable) if and only if*

$$\frac{c_i}{c_n} > 0 \ \ \textit{for every } i = 0, 1, \ldots, n-1 \tag{7.16}$$

*and*

$$b_{n-j,1} > 0 \ \ \textit{for every } j = 0, 1, \ldots, n-2 \tag{7.17}$$

*i.e. the elements in first column of the corresponding Routh array are nonzero and have the same sign.*

An elementary proof of the Routh-Hurwitz criterion can be found in [14, 34]. The Routh-Hurwitz stability criterion provides a simple algorithm to decide whether or not the zeros of a polynomial are all in the left half of the complex plane. The necessary condition that all roots have negative real parts is that all the elements of the first column of the Routh array (7.14) have the same sign. The number of changes of sign equals the number of roots with positive real

parts. However, there are cases where the condition (7.17) in the Routh-Hurwirz criterion needs special treatment in order to produce reliable results. In these cases the tabulation in the Routh array fails to come to an end due to a division by zero. This is commonly referred to as the *singular case*. Some techniques have been devised to cope with singular cases, leading to an *extended Routh-Hurwitz criterion* [7, 24, 66].

► **The singular cases in the Routh array.**

The types of the basic singular cases [7, 24, 66] and their numerical treatment are:

1. *The first element of a row in the Routh array is zero, but some other elements in that row are nonzero.*

   This means that at some place during the tabulation process of the Routh array the degree drops by more than one. In this case, we must simply replace the zero element by a number $\epsilon$, complete the array development, and then interpret the results assuming that $\epsilon$ is a small number of the same sign as the element above it. However, instead of the array for $f(s)$ we have the Routh array for a polynomial $f(s, \epsilon)$, where $f(s, \epsilon)$ is an integral rational function of $s$ and $\epsilon$ which reduces to $f(s)$ for $\epsilon = 0$. Since the roots of $f(s, \epsilon)$ change continuously with a change of the parameter $\epsilon$ and provided that there are no roots on the imaginary axis for $\epsilon = 0$, the polynomial $f(s, \epsilon)$ has the same number of roots in the right half-plane for values of $\epsilon$ of small modulus, [24]. The results must be interpreted in the limit as $\epsilon \to 0$.

2. *All the elements of a particular row in the Routh array are zero.*

   In this case, some of the roots of the polynomial are located symmetrically about the origin of the complex plane, for example a pair of purely imaginary roots. To overcome the problem, we write down the polynomial using the coefficients immediately previous to the all zero row. This polynomial is called *auxiliary polynomial*. To complete the array, we differentiate the auxiliary polynomial with respect to $s$ once and we use the coefficients in place of the all zero row. If the auxiliary polynomial has a root with multiplicity $\nu > 1$, then its derivative will have the same root with multiplicity $\nu - 1$, which means that it will also be a root of the greatest common divisor of the auxiliary polynomial and its derivative. Therefore, these changes are necessary in order to prevent the zeroing of $\nu$ rows in the array before the end of the tabulation process.

An alternative statement of the Routh-Hurwitz theorem was given by Wall [79], who formulated and proved Theorem 7.3 by means of continued fractions. More specifically, the recursive scheme (7.10) can be written in the form of a

continued fraction representing the rational function $R(s)$ in (7.9). Then, the following theorem describes equivalent conditions to Routh-Hurwitz theorem for the evaluation of the stability of a real polynomial.

**Theorem 7.4** ([24, 79]). *Given a real polynomial*

$$f(s) = s^n + c_{n-1}\, s^{n-1} + ... + c_1\, s\, + c_0$$

*define a rational function $R(s)$ such that*

$$R(s) = \begin{cases} \dfrac{ev\,(f(s))}{od\,(f(s))}, & \text{if } n \text{ is even.} \\ \\ \dfrac{od\,(f(s))}{ev\,(f(s))}, & \text{if } n \text{ is odd.} \end{cases} \tag{7.18}$$

*where $ev\,(f(s))$ is the even part of $f(s)$ (i.e. the power of $s$ is an even number) and $od\,(f(s))$ is the odd part of $f(s)$ (i.e. the power of $s$ is an odd number). Then all the roots of $f(s)$ have negative real parts if and only if*

$$R(s) = a_1\, s + \cfrac{1}{a_2\, s + \cfrac{1}{a_3\, s + ... \cfrac{1}{a_{n-1}\, s + \cfrac{1}{a_n\, s}}}} \tag{7.19}$$

*and all the $n$ coefficients $a_i$ are strictly positive numbers. Equivalently, if there exist negative $a_i$, then the number of negative $a_i$ is equal to the number of the roots of $f(s)$ in the right half-plane.*

In general, the proof of the above theorem is based on the Euclidean algorithm and the Routh-Hurwitz theorem. However, a detailed proof is presented in [24], based on the Hermite-Biehler theorem ([36] and references therein) and Stieltjes' theorem.

*REMARK* 7.6. The coefficients $a_i,\ i = 1, 2, \ldots, n$ in (7.19) are the ratios of two successive parameters in the first column of the Routh array and thus Theorem 7.3 and Theorem 7.4 provide equivalent results.

If we associate the stability of a linear system $\mathcal{S}$ with the stability of its characteristic polynomial $f(s)$, then, obviously, the signs of the coefficients $a_i$ are important to the assessment of the stability of the linear system. Specifically, if all the coefficients $a_i$ are positive, then the linear system $\mathcal{S}$ can be considered as asymptotically stable. The representation (7.19) of the rational function $R(s)$ as a finite continued fraction is an important algebraic representation which approaches the process of the evaluation of the parameters in the Routh array in a different

way and consequently motivates a different algorithmic procedure to assess the stability of a linear system.

In the following we will focus on the result given by Theorem 7.4. The main objective is to transform the rational representation (7.19) into an equivalent matrix representation in order to simplify the process of evaluating the stability of a linear time-invariant system based on the Routh-Hurwitz criterion. In this attempt the ERES method and particularly the ERES Division algorithm, which was developed in Chapter 3, will play a significant role.

## 7.3 The RH-ERES method for the evaluation of the stability of linear systems

Consider a linear time-invariant system $\mathcal{S} : \dot{x} = Ax$ and its characteristic polynomial:

$$f(s) = s^n + c_{n-1}\, s^{n-1} + ... + c_1\, s\, + c_0, \quad c_i \in \mathbb{R},\ s \in \mathbb{C} \tag{7.20}$$

We assume that $c_i > 0$ for every $i = 0, 1, \ldots, n-1$ which is a necessary condition to be $f(s)$ a stable (Hurwitz) polynomial. Naturally, the polynomial $f(s)$ can be written as the sum of two polynomials $f_0(s)$ and $f_1(s)$ such that

$$f(s) = f_0(s) + f_1(s) \tag{7.21}$$

with

$$\begin{cases} f_0(s) = s^n + c_{n-2}s^{n-2} + \ldots + c_3 s^3 + c_1 s \\ f_1(s) = c_{n-1}s^{n-1} + c_{n-3}s^{n-3} + \ldots + c_2 s^2 + c_0 \end{cases} \text{, if } n \text{ is odd,} \tag{7.22}$$

or

$$\begin{cases} f_0(s) = s^n + c_{n-2}s^{n-2} + \ldots + c_2 s^2 + c_0 \\ f_1(s) = c_{n-1}s^{n-1} + c_{n-3}s^{n-3} + \ldots + c_3 s^3 + c_1 s \end{cases} \text{, if } n \text{ is even.} \tag{7.23}$$

In this section, we focus on the development of an alternative ERES-based procedure for the computation of the coefficients $a_i$ of the continued fraction representation:

$$\frac{f_0(s)}{f_1(s)} = a_1\, s + \cfrac{1}{a_2\, s + \cfrac{1}{a_3\, s + ... \cfrac{1}{a_{n-1}\, s + \cfrac{1}{a_n\, s}}}} \tag{7.24}$$

This leads to formulations of the preceding theorems by means of the ERES

methodology. The derived computational method will be referred to as the *RH-ERES method* and its main usage is to evaluate the stability of a linear system from its characteristic polynomial. The following analysis refers to the regular case where the $n$ coefficients $a_i$ are nonzero and the fraction $\frac{f_0(s)}{f_1(s)}$ is irreducible. According to Theorem 7.4, if the sequence

$$\mathcal{A} = \{a_i,\ i = 1, 2, \ldots, n\} \tag{7.25}$$

contains strictly positive numbers, then the polynomial $f(s)$ is stable. In cases where any of the coefficients $a_i$ becomes zero, the expansion (7.24) fails to exist. However, the RH-ERES method can be modified to cope with singular cases, using similar techniques with those which are proposed for the Routh array [7, 24].

### ▶ The formulation of the RH-ERES method

Given a real polynomial $f(s)$ of the form (7.20), the RH-ERES method involves the following procedures:

1. Determine the degree $n$ of the polynomial $f(s)$ and then split it into two polynomials $f_0(s)$ and $f_1(s)$ according to the form (7.22) or (7.23). Then, create an initial matrix $P$ from the coefficients of $f_0(s)$ and $f_1(s)$.

2. Apply the ERES operations (Definition 3.3) to $P$, $n$ times iteratively.

Before all else, a proper initial basis matrix $P$ must be defined in respect to the polynomial $f(s)$.

**Definition 7.6.** If the degree $n$ of the polynomial $f(s)$ is an odd integer number, then a vector representative (vr) can be defined for the polynomial $f_0(s)$ regarding (7.22) such that

$$\underline{f}_0 = [1, 0, c_{n-2}, 0, \ldots, 0, c_3, 0, c_1, 0] \in \mathbb{R}^{n+1} \tag{7.26}$$

and for the polynomial $f_1(s)$ :

$$\underline{f}_1 = [0, c_{n-1}, 0, c_{n-3}, 0, \ldots, 0, c_2, 0, c_0] \in \mathbb{R}^{n+1} \tag{7.27}$$

Then,

$$\underline{p}(s) \triangleq \begin{bmatrix} f_0(s) \\ f_1(s) \end{bmatrix} = \begin{bmatrix} \underline{f}_0 \\ \underline{f}_1 \end{bmatrix} \cdot \underline{e}_n(s) \tag{7.28}$$

where $\underline{e}_n(s) = [s^n, s^{n-1} \ldots, s, 1]^t$ is a basis vector in $\mathbb{R}[s]$. Therefore, a basis matrix

representing the polynomial division $\frac{f_0(s)}{f_1(s)}$ can be defined as:

$$P = \begin{bmatrix} \underline{f}_0 \\ \underline{f}_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & c_{n-2} & 0 & \dots & c_3 & 0 & c_1 & 0 \\ 0 & c_{n-1} & 0 & c_{n-3} & \dots & 0 & c_2 & 0 & c_0 \end{bmatrix} \in \mathbb{R}^{2\times(n+1)} \quad (7.29)$$

Analogously, if $n$ is an even integer number, then, regarding (7.23), the basis matrix is defined as:

$$P = \begin{bmatrix} \underline{f}_0 \\ \underline{f}_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & c_{n-2} & 0 & \dots & 0 & c_2 & 0 & c_0 \\ 0 & c_{n-1} & 0 & c_{n-3} & \dots & c_3 & 0 & c_1 & 0 \end{bmatrix} \in \mathbb{R}^{2\times(n+1)} \quad (7.30)$$

Since we have assumed that the coefficients $c_i$ of the polynomial $f(s)$ are nonzero positive numbers, the degrees of the $f_0(s)$ and $f_1(s)$ will differ by 1. Then it occurs that the polynomial $f_0(s)$ can be written in the form:

$$\begin{aligned} f_0(s) &= \frac{1}{c_{n-1}} s \left( c_{n-1}s^{n-1} + c_{n-3}s^{n-3} + \ldots + c_2 s^2 + c_0 \right) \quad (7.31) \\ &+ \left( \left( c_{n-2} - \frac{c_{n-3}}{c_{n-1}} \right) s^{n-2} + \ldots + \left( c_3 - \frac{c_2}{c_{n-1}} \right) s^3 + \left( c_1 - \frac{c_0}{c_{n-1}} \right) s \right) \end{aligned}$$

and consequently,

$$\begin{aligned} f_0(s) &= \frac{1}{c_{n-1}} s f_1(s) + f_2(s) \quad \Leftrightarrow \\ \frac{f_0(s)}{f_1(s)} &= \frac{1}{c_{n-1}} s + \frac{f_2(s)}{f_1(s)} \quad \Leftrightarrow \\ \frac{f_0(s)}{f_1(s)} &= \frac{1}{c_{n-1}} s + \frac{1}{\frac{f_1(s)}{f_2(s)}} \quad (7.32) \end{aligned}$$

Clearly, in terms of the Euclidean division, the polynomial:

$$f_2(s) = \left( c_{n-2} - \frac{c_{n-3}}{c_{n-1}} \right) s^{n-2} + \ldots + \left( c_3 - \frac{c_2}{c_{n-1}} \right) s^3 + \left( c_1 - \frac{c_0}{c_{n-1}} \right) s \quad (7.33)$$

can be considered as the remainder $r_1(s)$ of the division $\frac{f_0(s)}{f_1(s)}$ and $a_1 = \frac{1}{c_{n-1}}$ is the coefficient of the quotient $q_1(s) = \frac{1}{c_{n-1}} s$ as well as the first term $a_1$ of the sequence (7.25). Obviously, the process may continue with the division $\frac{f_1(s)}{f_2(s)}$ in a similar way. Finally, it reaches the end after $n$ divisions of the form:

$$\frac{f_k(s)}{f_{k+1}(s)}, \text{ for } k = 0, 1, 2, \ldots, n-1 \quad (7.34)$$

and $f_n(s) = c_0$ with the convention $f_{n+1}(s) = 0$. Therefore, the continued fraction (7.24) can be acquired by using the next iterative procedure:

$$\begin{cases} \dfrac{f_{i-1}(s)}{f_i(s)} = a_i s + \dfrac{f_{i+1}(s)}{f_i(s)}, \quad i = 1, \ldots, n \\ \dfrac{f_{n+1}(s)}{f_n(s)} = 0 \end{cases} \tag{7.35}$$

In order to transform the above process into a coherent matrix-based procedure, we need a proper matrix representation of the polynomial division (7.34). Such a representation can be established in the context of the ERES methodology. More precisely, having two real polynomials $f_{i-1}(s)$ and $f_i(s)$ for any $i = 1, \ldots n$, the rational function:

$$R_i(s) = \frac{f_{i-1}(s)}{f_i(s)}$$

can be transformed into a new rational function:

$$R_{i+1}(s) = \frac{f_i(s)}{f_{i+1}(s)}$$

by using the ERES Division algorithm such that

$$R_i(s) = \frac{f_{i-1}(s)}{f_i(s)} \Longrightarrow \begin{bmatrix} f_{i-1}(s) \\ f_i(s) \end{bmatrix} \overset{ERES}{\longrightarrow} \begin{bmatrix} f_{i+1}(s) \\ f_i(s) \end{bmatrix} \Longrightarrow \frac{f_{i+1}(s)}{f_i(s)} = \frac{1}{R_{i+1}(s)}$$

The polynomial $f_{i+1}(s)$ actually represents the remainder of the division $\frac{f_{i-1}(s)}{f_i(s)}$. Therefore, considering the preceding analysis, a new iterative method can be developed for computing the sequence $\mathcal{A}$ in (7.25). We shall refer to this method as the *RH-ERES method*. This method suggests an algorithmic procedure for the computation of the coefficients $a_i$ in (7.35) by means of ERES operations.

The key element in computing (7.35) is to represent the remainder and the quotient of the initial division $\frac{f_0(s)}{f_1(s)}$ in matrix form. As we have analysed in Chapter 3, the ERES Division algorithm provides a matrix representation of the remainder and quotient of a polynomial division $\frac{a(s)}{b(s)}$ based on the ERES methodology. More particularly, having an initial matrix with dimensions $2 \times j$, $(j > 2)$, formed directly from the coefficients of the original polynomials, the RH-ERES method involves three basic procedures which are applied iteratively on the initial matrix and can be described as:

1. *Row switching.*
   The switching of the rows of the initial matrix $P$ can be achieved by multiplying it from its left side with a $2 \times 2$ matrix of the form:

$$J = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

2. *LU factorisation.*
   The LU factorization is basically Gaussian elimination without row interchanges. This process transforms the initial matrix $P$ into an upper trapezoidal matrix $U$ of the form:

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & \dots & u_{1,j-1} & u_{1,j} \\ 0 & u_{2,2} & \dots & u_{2,j-1} & u_{2,j} \end{bmatrix} \in \mathbb{R}^{2\times j}$$

   such that

$$U = L^{-1} \cdot P$$

   The transformation matrix is represented as:

$$L^{-1} = \begin{bmatrix} 1 & 0 \\ -\mu & 1 \end{bmatrix} \text{ with } \mu = \frac{p_{2,1}}{p_{1,1}}$$

   where $p_{1,1}$ and $p_{2,1}$ are the elements of the first column of $P$.

3. *Shifting.*
   The Shifting operation is applied in order to eliminate the first zero in the second row of the initial matrix, which occurs after the LU factorisation. Provided that the initial matrix has full rank, there always exists a $j \times j$ matrix $S$ which gives the proper shifting (Theorem 3.4).

The following statement provides the means for the implementation of the iterative procedure (7.35) and establishes a relation between the continued fraction representation (7.24) and the ERES operations, which characterises the RH-ERES method.

**Proposition 7.1.** *Given a real polynomial* $f(s) = s^n + c_{n-1}\, s^{n-1} + \dots + c_1\, s + c_0$ *with* $c_i \neq 0$ *for all* $i = 1, \dots, n$ *we may define a basis matrix* $P \in \mathbb{R}^{2\times(n+1)}$ *of the form (7.29) or (7.30). Then, all the terms in the sequence* $\mathcal{A} = \{a_i,\ i = 1, 2, \dots, n\}$, *which are also the coefficients* $a_i$ *in the continued fraction (7.24), can be obtained by applying ERES transformations to matrix* $P$. *Furthermore, there are* $n$ *matrices* $A_i \in \mathbb{R}^{2\times 2}$ *with trace* $tr(A_i) \in \mathbb{R}$ *such that*

$$\mathcal{A} = \{tr(A_i),\ i = 1, 2, \dots, n\} \tag{7.36}$$

*Proof.* First, we will simplify the structure of the basis matrix $P$. Let $n$ be an odd integer number. Since $c_{n-1} > 0$, the matrix $P$ is nonsingular and considering the ERES method, there is a shifting matrix $S$ (Theorem 3.4), which shifts the

second row of $P$ such that

$$\widetilde{P} = P \cdot S = \begin{bmatrix} 1 & 0 & c_{n-2} & 0 & \dots & c_3 & 0 & c_1 & 0 \\ c_{n-1} & 0 & c_{n-3} & 0 & \dots & c_2 & 0 & c_0 & 0 \end{bmatrix} \in \mathbb{R}^{2\times(n+1)} \tag{7.37}$$

If $n$ is an even integer number, then

$$\widetilde{P} = P \cdot S = \begin{bmatrix} 1 & 0 & c_{n-2} & 0 & \dots & 0 & c_2 & 0 & c_0 \\ c_{n-1} & 0 & c_{n-3} & 0 & \dots & 0 & c_1 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{2\times(n+1)} \tag{7.38}$$

The application of ERE operations does not alter the structure of a matrix and thus the zero columns of $\widetilde{P}$ remain unaffected. Then the zero columns of $\widetilde{P}$ can be deleted. This transformation can be achieved by using a matrix $Z = \left[\underline{e}_1, \underline{e}_3, \dots, \underline{e}_{n-2}, \underline{e}_n\right]$ with dimensions $(n+1) \times (\left[\frac{n}{2}\right] + 1)$, where $[\cdot]$ denotes the integer part of a real number. The vector $\underline{e}_j = [0, \dots, 0, 1, 0, \dots, 0]^t$ denotes the unit vector in $\mathbb{R}^n$ (the $j^{th}$ entry is equal to 1 , $j = 1, \dots, n$). The multiplication $\widetilde{P} \cdot Z$ results in a simplified form of $\widetilde{P}$ and hence the basis matrix $P$ is now transformed into a new matrix $P_0$ which basically contains the same data as $P$ and

- if the degree $n$ of the polynomial $f$ is odd, then

$$P_0 = \begin{bmatrix} 1 & c_{n-2} & \dots & c_3 & c_1 \\ c_{n-1} & c_{n-3} & \dots & c_2 & c_0 \end{bmatrix} \in \mathbb{R}^{2\times(\left[\frac{n}{2}\right]+1)} \tag{7.39}$$

- if the degree $n$ of the polynomial $f$ is even, then

$$P_0 = \begin{bmatrix} 1 & c_{n-2} & \dots & c_2 & c_0 \\ c_{n-1} & c_{n-3} & \dots & c_1 & 0 \end{bmatrix} \in \mathbb{R}^{2\times(\frac{n}{2}+1)} \tag{7.40}$$

We set now the initial matrix $P^{(0)} := P_0$ as defined in (7.39) or (7.40). Then, we apply to $P^{(0)}$ the ERES operations (procedures):

Row switching - LU factorisation - Shifting

and we repeat the same process $n$ times. At the $i^{th}$ iteration ($i \in \{1, 2, \dots, n\}$), let us denote by

- $J = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ the permutation matrix which is used for the row switching,
- $L^{(i)} = \begin{bmatrix} 1 & 0 \\ -\mu_i & 1 \end{bmatrix}$ the matrix for the LU factorisation,

- $S^{(i)}$ the shifting matrix which shifts the elements of the second row by one place to the left, and
- $P^{(i)}$ the matrix occurred at the end of the $i^{th}$ iteration.

Then we can represent the above process as follows:

$$P^{(i)} = L^{(i)} \cdot J \cdot P^{(i-1)} \cdot S^{(i)}, \quad \text{for } i = 1, 2, \dots n$$

Furthermore, let

$$A^{(i)} = L^{(i)} \cdot J \tag{7.41}$$

for all $i = 1, 2, \dots, n$. These matrices will have the form:

$$A^{(i)} = \begin{bmatrix} 0 & 1 \\ 1 & -\mu_i \end{bmatrix} \tag{7.42}$$

where the term $\mu_i$ represents the multiplier in the Gaussian elimination (LU factorisation). According to the theory of the ERES Division algorithm in Chapter 3, every $\mu_i$ also represents the coefficient of the quotient $q_i(s) = a_i\, s$ in (7.35). It is clear then that the terms $a_i$ in (7.35) and consequently in (7.24), satisfy the equality:

$$a_i = \mu_i \ , \ \forall\, i = 1, 2, \dots, n \tag{7.43}$$

At the end of the process the final matrix is:

$$P^{(n)} = \begin{bmatrix} c_0 & 0 & ... & 0 & 0 \\ 0 & 0 & ... & 0 & 0 \end{bmatrix} \in \mathbb{R}^{2\times(\left[\frac{n}{2}\right]+1)}$$

and

$$P^{(n)} = A^{(n)} \cdots A^{(1)} \cdot P^{(0)} \cdot S^{(1)} \cdots S^{(n-1)}$$

Since $\det(A^{(i)}) = -1 \neq 0$, all the matrices $A^{(i)}$ are invertible and, if we denote by $A_i$ their inverse, then

$$A_i \triangleq \left(A^{(i)}\right)^{-1} = \begin{bmatrix} \mu_i & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} a_i & 1 \\ 1 & 0 \end{bmatrix} \tag{7.44}$$

Therefore, we conclude that every trace $tr(A_i)$ is equal to $a_i$ for all $i = 1, 2, \dots, n$ and hence the relation (7.36) holds.

□

If we consider the above result, the result in Theorem 7.4 can be expressed in a different way which now involves the RH-ERES method.

**Theorem 7.5.** *Given a linear time-invariant system* $\mathcal{S}: \dot{x} = Ax$ *and its characteristic polynomial* $f(s) = s^n + c_{n-1}\, s^{n-1} + ... + c_1\, s + c_0$*, we may define an initial basis matrix* $P$ *such that*

- *If the degree* $n$ *of the polynomial* $f(s)$ *is odd, then*

$$P = \begin{bmatrix} 1 & c_{n-2} & ... & c_3 & c_1 \\ c_{n-1} & c_{n-3} & ... & c_2 & c_0 \end{bmatrix} \in \mathbb{R}^{2\times(\left[\frac{n}{2}\right]+1)} \tag{7.45}$$

- *If the degree* $n$ *of the polynomial* $f(s)$ *is even, then*

$$P = \begin{bmatrix} 1 & c_{n-2} & ... & c_2 & c_0 \\ c_{n-1} & c_{n-3} & ... & c_1 & 0 \end{bmatrix} \in \mathbb{R}^{2\times(\frac{n}{2}+1)} \tag{7.46}$$

*If the matrix* $P$ *is transformed according to the ERES methodology and the sequence of* $n$ *matrices* $A_i$*,* $i = 1, 2, \ldots, n$ *correspond to the performed ERE operations then, all the roots of the polynomial* $f(s)$ *have negative real parts if and only if the traces* $tr(A_i)$ *are strictly positive.*

Eventually, if we combine the Routh-Hurwitz theorem (7.11) with the previous Theorem 7.5, we conclude with the next corollary, which associates the ERES method with the evaluation of the stability of a linear system.

**Corollary 7.1** (RH-ERES stability criterion)**.** *The polynomial*

$$f(s) = c_n s^n + c_{n-1} s^{n-1} + \ldots + c_1 s + c_0, \quad c_n \neq 0$$

*is a Hurwitz polynomial (stable) if and only if*

$$\frac{c_i}{c_n} > 0 \;\; \textit{for every } i = 0, 1, \ldots, n-1$$

*and*

$$tr(A_i) > 0 \;\; \textit{for every } i = 1, 2, \ldots, n$$

*where the matrices* $A_i$ *represent the ERE operations, when the RH-ERES method is used for the transformation of the basis matrix* $P$*, which corresponds to the polynomial* $f(s)$*.*

*Proof.* The necessary and sufficient conditions for the polynomial $f(s)$ to be stable are a) to have strictly positive coefficients, and b) all its roots are lying in the left half-plane. These conditions are satisfied when $a_i = tr(A_i) > 0$ for every $i = 1, 2, \ldots, n$ as stated in Theorem 7.4, Theorem 7.5, and in Proposition 7.1. □

The above results provide the theoretical basis for the development of the *RH-ERES algorithm*, which can be used for the evaluation of the stability of a

linear system, based on the RH-ERES stability criterion. The formulation and implementation of this algorithm in a symbolic-numeric computational environment will be analysed and discussed in the following.

### 7.3.1 The RH-ERES algorithm and its implementation

The RH-ERES algorithm computes directly the terms $a_i$ in the continued fraction form (7.24) of the characteristic polynomial $f(s)$ of a given linear system and provides information about the stability of the linear system. The maximum number of iterations of the RH-ERES algorithm is equal to the degree $n$ of the original polynomial $f(s)$.

---

**ALGORITHM 7.1. The RH-ERES Algorithm**

Input : $2 \times k$ matrix $P^{(0)} = [p_{i,j}]$.

**For** $i = 1, 2, \ldots, n$ **do**
Step 1 : Reorder the rows of $P^{(i-1)}$.
Step 2 : Apply Gaussian elimination:
**If** $p_{1,1} \neq 0$ **then**
$\mu := -\dfrac{p_{2,1}}{p_{1,1}}$
**For** $j = 1, 2, \ldots, k$ **do**
$p_{2,j} := p_{2,j} + \mu \cdot p_{1,j}$
**end for**
$a_i := -\mu$
**else** break.
**end if**
Step 3 : Apply Shifting to the $2^{nd}$ row of $P^{(i-1)}$.
Step 4 : If the last column of $P^{(i-1)}$ is zero then
$k := k - 1$.
Set $P^{(i)} := P^{(i-1)}$.
**end for**

Output : Sequence $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$.

---

**▶ Computational complexity**

The RH-ERES algorithm can be implemented either numerically or symbolically. The symbolic implementation of the algorithm helps to avoid the accumulation

of numerical errors during the process of Gaussian elimination, when numerical data are used, but also it can provide useful theoretical results when symbolic variables are used. If the original data are given in floating-point format, they can be converted to a rational format and symbolic-rational operations are performed.

The row switching and Shifting do not require any numerical or symbolic computation. Therefore, the total amount of the required numerical operations (additions and multiplications) concerns only the process of Gaussian elimination. If $n$ denotes the degree of the original polynomial, then the number of columns of the initial matrix $P^{(0)}$ is equal to $k = \left[\frac{n}{2}\right] + 1$. The algorithm performs $n$ iterations and the Gaussian elimination requires $2k + 1$ numerical operations in every iteration. But on every second iteration, the number of nonzero columns of the transformed matrix $P^{(i-1)}$ decreases by 1. Finally, the total number of numerical operations required by the RH-ERES algorithm, given as a function of $k$, is

$$fl_{RH}(k) = 2\,k^2 + 2\,k - 1, \quad k \in \mathbb{N} \tag{7.47}$$

Conversely, for a given polynomial of degree $n$ and $k = \left[\frac{n}{2}\right] + 1$ as before, the tabulation process of the Routh array involves the computation of $k^2 - k$ determinants. Each determinant requires 3 numerical operations and thus, the computation of each element $b_{n-i,j}$ of the Routh array (7.14) requires 4 numerical operations. Therefore, the total number of numerical operations to complete the Routh array of a given polynomial of degree $n > 1$, is

$$fl_{RA}(k) = 4\,k^2 - 4\,k, \quad k \in \mathbb{N} \tag{7.48}$$

The following graph in Figure 7.1 shows that for $k > 3$ and consequently for $n > 5$, the RH-ERES algorithm requires less numerical operations than Routh's algorithm, which results in faster data processing and better numerical performance.

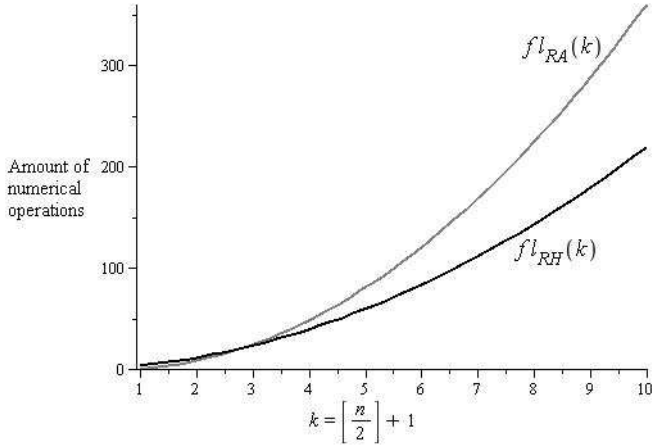


Figure 7.1: Comparison of the numerical complexity of the RH-ERES algorithm with Routh's algorithm.

This difference is actually more important when symbolic operations are used, due to the presence of symbolic variables in the data. This is the case where we want to do a complete stability analysis of a linear system theoretically, for instance to find relationships between design parameters and control parameters that give the best performance. However, the symbolic-rational operations require more computational time than the numerical floating-point operations in order to be executed. Therefore, since the total number of symbolic operations remains the same for the two algorithms as given in (7.47) and (7.48), obviously, the RH-ERES algorithm is faster in symbolic processing than Routh's algorithm.

### 7.3.2 Computational results of the RH-ERES algorithm

In the following examples, the RH-ERES algorithm is applied to polynomials in one variable with integer coefficients in order to determine the number of their roots which are located in the left half-plane and, hence, characterise them as stable or unstable.

**Example 7.1** (A stable polynomial)**.** Consider the polynomial:

$$f(s) = s^5 + 8\,s^4 + 35\,s^3 + 80\,s^2 + 94\,s + 52 \tag{7.49}$$

and we will examine whether it is stable or not by using the RH-ERES algorithm. The degree of the given polynomial is $n = 5$, which is an odd number and therefore, we form the initial matrix according to the form (7.45) :

$$P = \begin{bmatrix} 1 & 35 & 94 \\ 8 & 80 & 52 \end{bmatrix} \in \mathbb{R}^{2\times 3}$$

Using the matrix $P$ as input, the RH-ERES algorithm performs 5 iterations and produces the following results. The matrices $P^{(i)}$ occur at the end of the $i^{th}$ iteration of the algorithm, for $i = 1, 2, \ldots, 5$, and the matrices $A_i$ correspond to the ERE transformations.

- *Iteration 1*

$$P^{(1)} = \begin{bmatrix} 8 & 80 & 52 \\ 25 & \frac{175}{2} & 0 \end{bmatrix}, \quad A_1 = \begin{bmatrix} \frac{1}{8} & 1 \\ 1 & 0 \end{bmatrix}, \quad tr(A_1) = \frac{1}{8}$$

- *Iteration 2*

$$P^{(2)} = \begin{bmatrix} 25 & \frac{175}{2} & 0 \\ 52 & 52 & 0 \end{bmatrix}, \quad A_2 = \begin{bmatrix} \frac{8}{25} & 1 \\ 1 & 0 \end{bmatrix}, \quad tr(A_2) = \frac{8}{25}$$

- *Iteration 3*

$$P^{(3)} = \begin{bmatrix} 52 & 52 & 0 \\ \frac{125}{2} & 0 & 0 \end{bmatrix}, \quad A_3 = \begin{bmatrix} \frac{25}{52} & 1 \\ 1 & 0 \end{bmatrix}, \quad tr(A_3) = \frac{25}{52}$$

- *Iteration 4*

$$P^{(4)} = \begin{bmatrix} \frac{125}{2} & 0 & 0 \\ 52 & 0 & 0 \end{bmatrix}, \quad A_4 = \begin{bmatrix} \frac{104}{125} & 1 \\ 1 & 0 \end{bmatrix}, \quad tr(A_4) = \frac{104}{125}$$

- *Iteration 5*

$$P^{(5)} = \begin{bmatrix} 52 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad A_5 = \begin{bmatrix} \frac{125}{104} & 1 \\ 1 & 0 \end{bmatrix}, \quad tr(A_5) = \frac{125}{104}$$

According to Theorem 7.5, the produced sequence:

$$\mathcal{A} = \{tr(A_i),\ i = 1, \ldots, 5\} = \left\{\frac{1}{8}, \frac{8}{25}, \frac{25}{52}, \frac{104}{125}, \frac{125}{104}\right\}$$

implies that all the roots of the polynomial $f(s)$ have negative real parts, since there are no negative elements in the sequence $\mathcal{A}$. Additionally, the coefficients of $f(s)$ are strictly positive. Therefore, according to the RH-ERES stability criterion (Corollary 7.1), the given polynomial $f(s)$ is stable.

Indeed, the roots of $f(s)$ are:

$$s_1 = -1 + \mathtt{i}, \quad s_2 = -1 - \mathtt{i}, \quad s_3 = -2 + 3\,\mathtt{i}, \quad s_4 = -2 - 3\,\mathtt{i}, \quad s_5 = -2$$

and hence, the above results are confirmed. Furthermore, if we set

$$f_0(s) = s^5 + 35\,s^3 + 52 \quad , \quad f_1(s) = 8\,s^4 + 94\,s + 80\,s^2$$

so that $f_0(s) + f_1(s) = f(s)$, and the continued fraction representation of $f(s)$ can be easily verified:

$$\frac{f_0(s)}{f_1(s)} = \frac{s^5 + 35\,s^3 + 94\,s}{8\,s^4 + 80\,s^2 + 52} = \frac{1}{8}\,s + \cfrac{1}{\frac{8}{25}\,s + \cfrac{1}{\frac{25}{52}\,s + \cfrac{1}{\frac{104}{125}\,s + \cfrac{1}{\frac{125}{104}\,s}}}}$$

□

**Example 7.2** (An unstable polynomial)**.** We consider the polynomial:

$$f(s) = 4\,s^4 + s^3 + s^2 + 3\,s + 2 \tag{7.50}$$

The particular polynomial has 4 roots in the field of complex numbers $\mathbb{C}$, which are:

$$s_1 = \frac{3}{2} + \frac{\sqrt{5}}{2}\,\mathtt{i}, \quad s_2 = \frac{3}{2} - \frac{\sqrt{5}}{2}\,\mathtt{i}, \quad s_3 = -\frac{5}{8} + \frac{\sqrt{7}}{8}\,\mathtt{i}, \quad s_4 = -\frac{5}{8} - \frac{\sqrt{7}}{8}\,\mathtt{i}$$

Obviously, there are 2 roots with negative real parts and 2 roots with positive real parts. Therefore, although the polynomial $f(s)$ has strictly positive integer coefficients, it is unstable.

Now, we will verify these results with the RH-ERES algorithm. First, we notice that the leading coefficient of $f(s)$ is $c_n = 4 > 1$, but the algorithm can produce the same results without the restriction of $f(s)$ being a monic polynomial. Alternatively, we may divide all the coefficients of $f(s)$ by 4, and use the polynomial:

$$\hat{f}(s) = s^4 + \frac{1}{4}\,s^3 + \frac{1}{4}\,s^2 + \frac{3}{4}\,s + \frac{1}{2}$$

We form the initial matrix $P$ for the original polynomial $f(s)$ according to (7.46) :

$$P = \begin{bmatrix} 4 & 1 & 2 \\ \\ 1 & 3 & 0 \end{bmatrix} \in \mathbb{R}^{2\times 3}$$

and then, the RH-ERES algorithm produces the sequence:

$$\mathcal{A} = \{tr(A_i),\ i = 1, \ldots, 4\} = \left\{4, -\frac{1}{11}, -\frac{121}{35}, \frac{35}{22}\right\}$$

There are two negative elements in $\mathcal{A}$ which implies that there are two roots of $f(s)$ with positive real parts, as we expected. Furthermore, if we set

$$f_0(s) = 4\,s^4 + s^2 + 2 \quad , \quad f_1(s) = s^3 + 3\,s$$

so that $f_0(s) + f_1(s) = f(s)$, then,

$$\frac{f_0(s)}{f_1(s)} = \frac{4\,s^4 + s^2 + 2}{s^3 + 3\,s} = 4\,s + \cfrac{1}{-\frac{1}{11}\,s + \cfrac{1}{-\frac{121}{35}\,s + \cfrac{1}{\frac{35}{22}\,s}}}$$

□

**▶ The singular cases in the RH-ERES algorithm.**

Similar singular cases as in the Routh array can also appear in the RH-ERES algorithm when a coefficient $a_i$ cannot be determined due to a division by zero. Specifically, we may have:

- *Case 1:* The first element of the second row of the matrix $P^{(i)}$ at the end of the $i^{th}$ iteration of the RH-ERES algorithm is zero.
- *Case 2:* The entire second row of the matrix $P^{(i)}$ at the end of the $i^{th}$ iteration of the RH-ERES algorithm is zero.

In both cases, the continued fraction representation (7.24) fails to exist and we cannot come to a conclusion about the stability of a given polynomial $f(s)$. However, if we recall that the coefficients $a_i,\ i = 1, 2, \ldots, n$ in the continued fraction (7.24) are the ratios of two successive parameters in the first column of the Routh array (Remark 7.6), we can deal with the singular cases by following the special rules given by Routh [24, 66] for continuing the array (7.14), and apply them appropriately to the RH-ERES algorithm when a singular case appears.

Therefore, in Case 1, we have to replace the zero element by a parameter $\epsilon$ of definite (but arbitrary) sign, complete the main iterative procedure of the RH-ERES algorithm by using symbolic computations, and then interpret the results assuming that $\epsilon$ is a small number. The coefficients $a_i$ are given as rational functions of $\epsilon$, and their signs are determined by the "smallness" and the sign of $\epsilon$. The results must be interpreted in the limit as $\epsilon \to 0$. This process can be repeated several times with different arbitrary parameters, if this singular case appears more than once. However, the introduction of small parameters is justified only when the original polynomial $f(s)$ has no roots on the imaginary axis, because by varying the parameter $\epsilon$ some of these roots may pass over into the right half-plane and change the number of the roots with negative real parts [24].

If a singularity of the second type (Case 2) appears during the $i^{th}$ iteration of the RH-ERES algorithm, we have to replace the zero row of the processed matrix $P^{(i-1)}$ with the vector that correspond to the derivative of the polynomial $f_i(s)$ that fills the first row of $P^{(i-1)}$ (i.e. the auxiliary polynomial). Practically, this is equivalent to replacing the zero elements of the second row of $P^{(i-1)}$ with

$$p_{2,j} := (n - i - 2j + 2) \cdot p_{1,j}\ , \quad \text{for } j = 1, 2, \ldots, k \tag{7.51}$$

where $p_{1,j}$ and $p_{2,j}$ denote the elements of the first and second row of $P^{(i-1)}$ respectively, $j$ denotes the number of column, $n$ is the degree of the original polynomial $f(s)$, $k$ is the number of nonzero columns of $P^{(i-1)}$, and $i$ denotes the number of the iteration where the singular case appeared. Moreover, if the roots

of $f_i(s)$ are not simple, this process has to be applied several times to dispose of a singularity of this type [24].

**Example 7.3** (Singular Case 1)**.** Consider the polynomial:

$$f(s) = s^5 + 2\,s^4 + 3\,s^3 + 6\,s^2 + 5\,s + 3 \tag{7.52}$$

We will examine whether it is stable or not by using the RH-ERES algorithm. The given polynomial $f(s)$ has the following roots in $\mathbb{C}$ :

$s_1 = \ \ 0.3428775611 - 1.5082901610\,\mathtt{i}, \quad s_2 = \ \ 0.3428775611 + 1.5082901610\,\mathtt{i},$
$s_3 = -0.5088331416 - 0.7019951318\,\mathtt{i}, \quad s_4 = -0.5088331416 + 0.7019951318\,\mathtt{i},$
$s_5 = -1.6680888390$

The degree of $f(s)$ is equal to 5, which is an odd number and therefore, we form the initial matrix according to (7.45) :

$$P = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 6 & 3 \end{bmatrix} \in \mathbb{R}^{2\times 3}$$

The RH-ERES algorithm is expected to perform 5 iterations. But, at the end of the first iteration, a singular case appears and the algorithm temporarily stops.

- *Iteration 1*

$$P^{(1)} = \begin{bmatrix} 2 & 6 & 3 \\ 0 & \frac{7}{2} & 0 \end{bmatrix}, \quad tr(A_1) = \frac{1}{2}$$

  As we can see, the first element of the second row of $P^{(1)}$ is zero. Therefore, we must substitute this element with an arbitrary parameter $\epsilon$, assuming that it is a "small" number. The algorithm continues with the matrix:

$$P^{(1)} = \begin{bmatrix} 2 & 6 & 3 \\ \epsilon & \frac{7}{2} & 0 \end{bmatrix}$$

- *Iteration 2*

$$P^{(2)} = \begin{bmatrix} \epsilon & \frac{7}{2} & 0 \\ \frac{6\,\epsilon - 7}{\epsilon} & 3 & 0 \end{bmatrix}, \quad tr(A_2) = \frac{2}{\epsilon}$$

- *Iteration 3*

$$P^{(3)} = \begin{bmatrix} \frac{6\,\epsilon - 7}{\epsilon} & 3 & 0 \\ -\frac{1}{2}\,\frac{-42\,\epsilon + 49 + 6\,\epsilon^2}{6\,\epsilon - 7} & 0 & 0 \end{bmatrix}, \quad tr(A_3) = \frac{\epsilon^2}{6\,\epsilon - 7}$$

- *Iteration 4*

$$P^{(4)} = \begin{bmatrix} -\frac{1}{2}\,\frac{-42\,\epsilon+49+6\,\epsilon^2}{6\,\epsilon-7} & 0 & 0 \\ 3 & 0 & 0 \end{bmatrix}, \quad tr(A_4) = \frac{2\,(6\,\epsilon-7)^2}{42\,\epsilon^2-49\,\epsilon-6\,\epsilon^3}$$

- *Iteration 5*

$$P^{(5)} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad tr(A_5) = \frac{42\,\epsilon-49-6\,\epsilon^2}{36\,\epsilon-42}$$

Finally, the following sequence is produced:

$$\mathcal{A} = \{tr(A_i),\ i = 1,\ldots,5\} = \left\{\frac{1}{2}, \frac{2}{\epsilon}, \frac{\epsilon^2}{6\,\epsilon-7}, \frac{2\,(6\,\epsilon-7)^2}{42\,\epsilon^2-49\,\epsilon-6\,\epsilon^3}, \frac{42\,\epsilon-49-6\,\epsilon^2}{36\,\epsilon-42}\right\}$$

Since we have assumed that $\epsilon$ is a small number, close enough to zero, the signs of the elements of $\mathcal{A}$ depend on the sign of $\epsilon$. Then, we have:

- For $\epsilon \to 0^+$ : $sign\{\mathcal{A}\} = \{+,+,-,-,+\} \quad \Rightarrow\ r_f^+ = 2$
- For $\epsilon \to 0^-$ : $sign\{\mathcal{A}\} = \{+,-,-,+,+\} \quad \Rightarrow\ r_f^+ = 2$

Regardless of the sign of $\epsilon$, the above sequence $\mathcal{A}$ includes two negative elements, and according to Theorem 7.5, it is verified that there are two roots in the right half-plane; consequently the given polynomial $f(s)$ is unstable.

Now, if we use the elements of the sequence $\mathcal{A}$ to form the continued fraction (7.24) for $f(s)$, we obtain:

$$\frac{f_0(s)}{f_1(s)} = \frac{s^5 + (3+\epsilon)\,s^3 + 5\,s}{2\,s^4 + 6\,s^2 + 3}$$

The above rational representation corresponds to the polynomial:

$$f(s,\epsilon) = s^5 + 2\,s^4 + (3+\epsilon)\,s^3 + 6\,s^2 + 5\,s + 3$$

which is actually the same as the polynomial $f(s)$ with a small perturbation in the coefficient $c_3 = 3$. For $\epsilon = 0$ it holds

$$f(s,\epsilon) = f(s)$$

and since there are no roots on the imaginary axis for $\epsilon = 0$, the number of roots in the right half-plane is the same for values of $\epsilon$ of small modulus, as it is proved in [24]. □

**Example 7.4** ([24])**.** Consider the polynomial:

$$f(s) = s^6 + s^5 + 3\,s^4 + 3\,s^3 + 3\,s^2 + 2\,s + 1 \tag{7.53}$$

The given polynomial $f(s)$ has 3 pairs of roots in $\mathbb{C}$ :

$$\begin{aligned} s_{1,2} &= +0.1217444141 \pm 1.3066224030\,\mathtt{i} \\ s_{3,4} &= -0.6217444141 \pm 0.4405969990\,\mathtt{i} \\ s_{5,6} &= \pm\mathtt{i} \end{aligned}$$

and a pair of roots lies on the imaginary axis. When we apply the RH-ERES algorithm to $f(s)$ in order to evaluate its stability, we notice that a singularity of the first type appears at the end of the second iteration of the algorithm, i.e. in the second row of the processed matrix the first element is zero. Then, we substitute this zero element with an arbitrary parameter $\epsilon$, assuming that it is a "small" number of arbitrary sign. Finally, the following sequence is produced:

$$\mathcal{A} = \left\{1, \frac{1}{\epsilon},\ \frac{\epsilon^2}{3\,\epsilon - 1},\ \frac{-9\,\epsilon^2 + 6\,\epsilon - 1}{2\,\epsilon^3 - 4\,\epsilon^2 + \epsilon},\ \frac{-4\,\epsilon^4 + 16\,\epsilon^3 - 20\,\epsilon^2 + 8\,\epsilon - 1}{12\,\epsilon^3 - 7\,\epsilon^2 + \epsilon},\ \frac{4\,\epsilon^2 - \epsilon}{2\,\epsilon^2 - 4\,\epsilon + 1}\right\}$$

Since we have assumed that $\epsilon$ is a small number, close enough to zero, the signs of the elements of $\mathcal{A}$ depend on the sign of $\epsilon$. Then, we have:

- For $\epsilon \to 0^+$ : $sign\{\mathcal{A}\} = \{+, +, -, -, -, -\} \quad \Rightarrow\ r_f^+ = 4$
- For $\epsilon \to 0^-$ : $sign\{\mathcal{A}\} = \{+, -, -, +, +, +\} \quad \Rightarrow\ r_f^+ = 2$

It is obvious that the number of negative elements in $\mathcal{A}$ is different when we change the sign of $\epsilon$. Therefore, in this case we cannot decide how many roots of $f(s)$ are located in the right half-plane. However, since there exist negative elements in $\mathcal{A}$, regardless of the sign of $\epsilon$, the given polynomial $f(s)$ is unstable. □

**Example 7.5** (Singular Case 2)**.** Consider the polynomial:

$$f(s) = s^6 + 2\,s^5 + 8\,s^4 + 12\,s^3 + 20\,s^2 + 16\,s + 16 \tag{7.54}$$

We will examine if the polynomial $f(s)$ is stable or unstable by using the RH-ERES algorithm. The given polynomial $f(s)$ has 3 pairs of roots in $\mathbb{C}$ :

$$s_{1,2} = -1 \pm \mathtt{i}, \quad s_{3,4} = \pm 2\,\mathtt{i}, \quad s_{5,6} = \pm\sqrt{2}\,\mathtt{i}$$

and we notice that there are two pairs of roots in the imaginary axis. The degree of $f(s)$ is equal to 6, which is an even number, and therefore, we form the initial

matrix according to (7.46) :

$$P = \begin{bmatrix} 1 & 8 & 20 & 16 \\ 2 & 12 & 16 & 0 \end{bmatrix}$$

The RH-ERES algorithm will perform 6 iterations. However, at the second iteration, the entire second row of the matrix in process is zeroed.

$$P^{(2)} = \begin{bmatrix} 2 & 12 & 16 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Normally, since the algorithm has performed its second iteration, the first two elements of the second row of $P^{(2)}$ should be non zero. In this case, we substitute the elements of the second row of $P^{(2)}$ with multiples of the elements of the first row according to the rule in (7.51). The algorithm now continues with the matrix:

$$P^{(3)} = \begin{bmatrix} 2 & 12 & 16 & 0 \\ 8 & 24 & 0 & 0 \end{bmatrix}$$

Finally, the RH-ERES algorithm produces the sequence:

$$\mathcal{A} = \{tr(A_i),\ i = 1, \ldots, 6\} = \left\{\frac{1}{2}, 1, \frac{1}{4}, \frac{4}{3}, \frac{9}{4}, \frac{1}{6}\right\}$$

Since there are no negative elements in the sequence $\mathcal{A}$, we may only conclude that the given polynomial $f(s)$ does not have any root with positive real part, which is true. Therefore, the given polynomial $f(s)$ is definitely not unstable. Also, the continued fraction representation (7.24) with coefficients from the above sequence $\mathcal{A}$ cannot represent the given polynomial $f(s)$, because we have altered the data during the processing. In fact, we have:

$$\frac{1}{2}\,s + \cfrac{1}{s + \cfrac{1}{\frac{1}{4}\,s + \cfrac{1}{\frac{4}{3}\,s + \cfrac{1}{\frac{9}{4}\,s + \cfrac{1}{\frac{1}{6}\,s}}}}} = \frac{s^6 + 12\,s^4 + 32\,s^2 + 16}{2\,s^5 + 20\,s^3 + 40\,s}$$

which actually is the continued fraction representation of the polynomial:

$$\hat{f}(s) = s^6 + 2\,s^5 + 12\,s^4 + 20\,s^3 + 32\,s^2 + 40\,s + 16$$

Although the polynomials $\hat{f}(s)$ and $f(s)$ are different, they have the same number of roots in the right half-plane [24, 66]. Furthermore, the polynomial $\hat{f}(s)$ does

not have pure imaginary roots, and thus, it can be characterised as stable. □

► **Eliminating of imaginary roots.**

As is evident in the last two examples, the presence of pure imaginary roots causes some confusion when we examine the distributions of the roots of a polynomial on the complex plane either by using the RH-ERES algorithm or the Routh array. In the general case, having a real polynomial $f(s)$ of degree $n > 2$ and setting $f(s) = f_0(s) + f_1(s)$ as defined in (7.20) – (7.23), we must check for roots in the imaginary axis by finding the greatest common divisor $g(s)$ of the polynomials $f_0(s)$ and $f_1(s)$. Then,

$$\frac{f_0(s)}{f_1(s)} = \frac{g(s)\, f_0^*(s)}{g(s)\, f_1^*(s)}$$

and consequently,

$$\begin{aligned} f(s) &= g(s)\ (f_0^*(s) + f_1^*(s)) \quad \Leftrightarrow \\ f(s) &= g(s)\, f^*(s) \end{aligned} \tag{7.55}$$

**Proposition 7.2.** *The polynomial $g(s)$ in (7.55) contains only those roots of $f(s)$ which are opposite numbers in $\mathbb{C}$.*

*Proof.* We assume that $s_0 \in \mathbb{C}$ is a root of $f(s)$ for which $-s_0$ is also a root. This is a property of the roots which are located symmetrically about the origin of the complex plane, and of course the roots on the imaginary axis have this property. Without loss of generality, let $n$ be an odd number and the polynomials $f_0(s)$ and $f_1(s)$ as defined in (7.22). Then, it follows from $f(s_0) = f(-s_0) = 0$ that :

$$\left\{ \begin{array}{rl} f(s_0) & = 0 \\ f(-s_0) & = 0 \end{array} \right. \Leftrightarrow \left\{ \begin{array}{rl} f_0(s_0) + f_1(s_0) & = 0 \\ f_0(-s_0) + f_1(-s_0) & = 0 \end{array} \right. \Leftrightarrow \left\{ \begin{array}{rl} f_0(s_0) + f_1(s_0) & = 0 \\ -f_0(s_0) + f_1(s_0) & = 0 \end{array} \right. \Leftrightarrow$$

$$\left\{ \begin{array}{rl} f_0(s_0) = & -f_1(s_0) \\ f_1(s_0) = & f_0(s_0) \end{array} \right. \Leftrightarrow \left\{ \begin{array}{rl} f_0(s_0) & = 0 \\ f_1(s_0) & = 0 \end{array} \right.$$

Therefore, $s_0$ is a root of their greatest common divisor, $g(s)$. By the same argument, $-s_0$ is also a root of $g(s)$, and thus the polynomial $f^*(s)$ in (7.55) has no opposite roots.

Now, let $s_1 \in \mathbb{C}$ be a root of $f(s)$ for which $-s_1$ is not a root of $f(s)$.

- If $f_0(s_1) \neq f_1(s_1) \neq 0$, then

$$f(s_1) = f_0(s_1) + f_1(s_1) = 0 \quad \Leftrightarrow$$

$$f_0(s_1) = -f_1(s_1) \quad \Leftrightarrow$$

$$\frac{f_0(s_1)}{f_1(s_1)} = -1$$

thus, $s_1$ cannot be a root of $g(s)$.

- If $f_0(s_1) = f_1(s_1) = 0$, then

$$-f_0(s_1) + f_1(s_1) = 0 \quad \Leftrightarrow$$

$$f_0(-s_1) + f_1(-s_1) = 0 \quad \Leftrightarrow$$

$$f(-s_1) = 0$$

which is inappropriate, since we have assumed that $-s_1$ is not a root of $f(s)$.

Thereafter, the polynomial $g(s)$ contains only the opposite roots of $f(s)$, and furthermore, the degree of $g(s)$ is always an even number. □

If $r_f^+$ denotes the number of roots of $f(s)$ in the right half-plane, then

$$r_f^+ = r_g^+ + r_{f^*}^+ \tag{7.56}$$

where $r_g^+$ and $r_{f^*}^+$ denote the number of roots of $g(s)$ and $f^*(s)$ in the right half-plane, respectively. The polynomial $f^*(s)$ in (7.55) is now free of roots in the imaginary axis and the number of roots $r_{f^*}^+$ can be determined by the RH-ERES algorithm. In addition,

$$r_g^+ = \frac{1}{2}(d - r) \tag{7.57}$$

where $d$ is the degree of $g(s)$ and $r$ is the number of real roots of the polynomial $g(w\,\mathtt{i})$ or else the number of the conjugate pure imaginary roots of $g(s)$, [24]. Since the degree $d$ of $g(s)$ is an even number, the polynomial $g(\omega\,\mathtt{i})$ is a real polynomial and the number $r$ of its real roots can be determined by Sturm's theorem (Theorem 7.1).

**Example 7.6.** Consider the polynomial:

$$f(s) = s^{10} + s^9 + 3\,s^8 + 3\,s^7 + 5\,s^6 + 4\,s^5 + 7\,s^4 + 6\,s^3 + 6\,s^2 + 4\,s + 2 \tag{7.58}$$

When we apply the RH-ERES algorithm to $f(s)$ in order to evaluate its stability, we notice that a singularity of the first type appears at the end of the second iteration, i.e. in the second row of the processed matrix the first element is zero. Then, this zero element is substituted by a parameter $\epsilon$, assuming that it is a "small" number of arbitrary sign. The algorithm continues and performs another 8 iterations. Finally, a sequence $\mathcal{A} = \{a_i = tr(A_i),\ i = 1, 2, \ldots, 10\}$ is produced, and the values of its elements $a_i$ are given below.

$$a_1 = 1, \qquad a_6 = \frac{\epsilon}{-20\,\epsilon + 5 + 20\,\epsilon^2},$$

$$a_2 = \frac{1}{\epsilon}, \qquad a_7 = \frac{300\,\epsilon^2 - 150\,\epsilon - 200\,\epsilon^3 + 25}{72\,\epsilon^2 - 28\,\epsilon},$$

$$a_3 = \frac{\epsilon^2}{3\,\epsilon - 1}, \qquad a_8 = \frac{-324\,\epsilon^3 + 252\,\epsilon^2 - 49\,\epsilon}{80\,\epsilon^4 - 320\,\epsilon^3 + 340\,\epsilon^2 - 140\,\epsilon + 20},$$

$$a_4 = \frac{-9\,\epsilon^2 + 6\,\epsilon - 1}{4\,\epsilon^3 - 4\,\epsilon^2 + \epsilon}, \qquad a_9 = \frac{-32\,\epsilon^4 + 192\,\epsilon^3 - 352\,\epsilon^2 + 192\,\epsilon - 32}{144\,\epsilon^3 - 110\,\epsilon^2 + 21\,\epsilon},$$

$$a_5 = \frac{-12\,\epsilon^2 + 6\,\epsilon + 8\,\epsilon^3 - 1}{3\,\epsilon^2 - \epsilon}, \quad a_{10} = \frac{8\,\epsilon^2 - 3\,\epsilon}{4\,\epsilon^2 - 12\,\epsilon + 4}.$$

Since we have assumed that $\epsilon$ is a small number, close enough to zero, the signs of the elements of $\mathcal{A}$ depend on the sign of $\epsilon$. Then, we have:

- For $\epsilon \to 0^+$ : $sign\{\mathcal{A}\} = \{+,+,-,-,+,+,-,-,-,-\} \quad \Rightarrow\ r_f^+ = 6$
- For $\epsilon \to 0^-$ : $sign\{\mathcal{A}\} = \{+,-,-,+,-,-,+,+,+,+\} \quad \Rightarrow\ r_f^+ = 4$

Yet again, we cannot answer how many roots of $f(s)$ have negative or positive real parts. The problem is actually caused by the presence of roots in the imaginary axis and we have to examine this case carefully.

Let $f(s) = f_0(s) + f_1(s)$ with

$$\begin{aligned} f_0(s) &= s^{10} + 3\,s^8 + 5\,s^6 + 7\,s^4 + 6\,s^2 + 2 \\ f_1(s) &= s^9 + 3\,s^7 + 4\,s^5 + 6\,s^3 + 4\,s \end{aligned}$$

Then, as proved in Proposition 7.2, the greatest common divisor $g(s)$ of the polynomials $f_0(s)$ and $f_1(s)$ contains all the pairs of opposite roots of $f(s)$, if any exist . We follow the next process in order to detect them:

1. Compute the polynomial $g(s)$ by using the Hybrid ERES algorithm.

$$g(s) = s^6 + s^4 + 2\,s^2 + 2, \quad d = deg\{g(s)\} = 6$$

2. Factorise $f(s)$ by using the ERES Division algorithm.

$$f(s) = g(s)\,f^*(s) \quad \Leftrightarrow \quad f^*(s) = \frac{f(s)}{g(s)} = s^4 + s^3 + 2\,s^2 + 2\,s + 1$$

The polynomial $f^*(s)$ does not have conjugate pure imaginary roots.

3. We determine the number $r_{f^*}^+$ of the roots of $f^*(s)$ which are located in the right half-plane by using the RH-ERES algorithm.

$$\mathcal{A}_{f^*} = \left\{1,\ \frac{1}{\epsilon},\ \frac{\epsilon^2}{2\,\epsilon - 1},\ \frac{2\,\epsilon - 1}{\epsilon}\right\}$$

- For $\epsilon \to 0^+$ : $sign\{\mathcal{A}_{f^*}\} = \{+,+,-,-\} \quad \Rightarrow\ r_{f^*}^+ = 2$
- For $\epsilon \to 0^-$ : $sign\{\mathcal{A}_{f^*}\} = \{+,-,-,+\} \quad \Rightarrow\ r_{f^*}^+ = 2$

Therefore, $r_{f^*}^+ = 2$, regardless of the sign of $\epsilon$.

4. We determine the number $r_g^+$ of the roots of $g(s)$ which are located in the right half-plane by using the formula (7.57). First, we have to compute the number $r$ of the real roots of the polynomial:

$$g(w\,\mathtt{i}) = -w^6 + w^4 - 2\,w^2 + 2, \quad w \in \mathbb{R}$$

For this task, we construct the Sturm chain $\mathcal{P}_w$ of the real polynomial $g(w\,\mathtt{i})$ as described in Definition 7.4 :

$$\mathcal{P}_w = \left\{ \begin{array}{l} p_0(w) = -w^6 + w^4 - 2\,w^2 + 2, \\ p_1(w) = -w^5 + \frac{2}{3}\,w^3 - \frac{2}{3}\,w, \\ p_2(w) = -w^4 + 4\,w^2 - 6, \\ p_3(w) = +w^3 - \frac{8}{5}\,w, \\ p_4(w) = -w^2 + \frac{5}{2}, \\ p_5(w) = -w, \\ p_6(w) = -1 \end{array} \right\}$$

Then, according to Sturm's theorem (Theorem 7.1), the number of real roots of $g(w\,\mathtt{i})$ is $r = 2$. This also means that the polynomial $g(s)$, and consequently the original polynomial $f(s)$, has a pair of pure imaginary roots. The number of roots of $g(s)$ in the right half-plane is given by the formula (7.57) :

$$r_g^* = \frac{6-2}{2} = 2$$

Finally, from (7.56) we have $r_f^+ = 4$, and therefore, the given polynomial $f(s)$ is unstable. In Figure 7.2, the distribution of the roots of $f(s)$ in the complex plane is illustrated and the above results are verified. There are 4 roots in the left half-plane, 4 roots in the right half-plane, and 2 roots in the imaginary axis. Also, there are 6 opposite roots (including those in the imaginary axis), which are actually the roots of $g(s)$. □

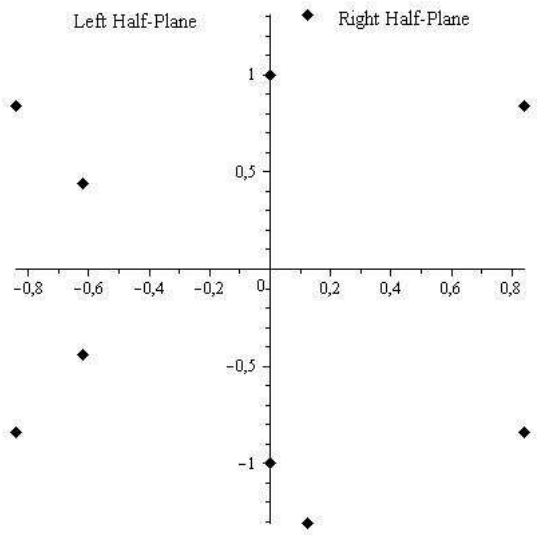


Figure 7.2: Distribution of the roots of the polynomial $f(s)$ in Example 7.6.

## 7.4 Distance of an unstable polynomial to stability

In this section we will consider the problem of finding the minimum distance of an unstable polynomial from the "stability domain", i.e. the set of all stable polynomials, as well as the required minimum norm for stabilisation of unstable polynomials by employing the developed framework of the RH-ERES method. The distance of a stable polynomial from instability provides vital information for robustness analysis of a stable system while the minimum norm stabilisation of unstable polynomials is a very important indicator for the study of systems under state or output feedback. The problem of minimum norm stabilisation of an unstable polynomial is to find a perturbation on the coefficients which stabilizes the polynomial, while the norm of the perturbation is minimized in certain sense. Here, the perturbation to the polynomials is defined in terms of the Euclidean norm $\|\cdot\|_2$.

We consider the real nominal polynomial:

$$f(s) = s^n + c_{n-1}s^{n-1} + \ldots + c_1 s + c_0, \quad c_k \in \mathbb{R},\ s \in \mathbb{C}$$

which in practise is the characteristic polynomial of a linear system $\mathcal{S}$. For the purposes of our study $f(s)$ is considered to be unstable. This polynomial is assumed to be perturbed by

$$\Delta f(s) = \delta_{n-1}\, s^{n-1} + \delta_{n-2}\, s^{n-2} + \ldots + \delta_1 + \delta_0 \tag{7.59}$$

where $\underline{\delta} = [\delta_0, \delta_2, \ldots, \delta_{n-1}]^t \in \mathbb{R}^n$ may denote the effect of parametric uncertain-

ties, or feedback as perturbations on the coefficients. Then, we can define the performance index:

$$Q\left(\Delta f\right) \triangleq \|\underline{\delta}\|_2^2 = \delta_{n-1}^2 + \delta_{n-2}^2 + \ldots + \delta_1^2 + \delta_0^2 \tag{7.60}$$

to be minimised. The new perturbed polynomial will be:

$$\widetilde{f}(s) = f(s) - \Delta f(s) = s^n + \tilde{c}_{n-1}s^{n-1} + ... + \tilde{c}_1 s + \tilde{c}_0 \tag{7.61}$$

and

$$\tilde{c}_k = c_k - \delta_k, \quad \forall\, k = 0, 1, \ldots, n-1 \tag{7.62}$$

The polynomial $\widetilde{f}(s)$ is assumed to be stable. Now, the problem under consideration will be:
*PROBLEM: For an unstable nominal polynomial $f$, i.e. $\Lambda(f) \cap \mathbb{C}^+ \neq \emptyset$, find a perturbation $\underline{\delta}$ or $\Delta f(s)$ polynomial with minimum norm $\gamma_{f^+}$, which stabilizes the resulting perturbed polynomial $\widetilde{f}(s) = f(s) - \Delta f(s)$, i.e.*

$$\gamma_{f^+} = inf\left\{Q\left(\Delta f\right) : \Lambda(\widetilde{f}\,) \cap \mathbb{C}^+ = \emptyset\right\} \tag{7.63}$$

*where* $\inf\{\cdot\}$ *denotes the infimum of the set, and* $\Lambda(f)$ *denotes the root set of* $f(s)$.

We will investigate the above problem by means of the RH-ERES method and Theorem 7.4. If $\mathcal{A}_{\widetilde{f}} = \{a_i, i = 1, 2, \ldots, n\}$ is the sequence provided from the RH-ERES algorithm for the perturbed polynomial $\widetilde{f}(s)$ and since it is required to be stable, the above problem (7.63) can be stated as:

$$\gamma_{f^+} = \begin{cases} \min\left\{Q(\Delta f)\right\} \\ a_i > 0, \quad \forall\ i = 1, 2, \ldots, n \\ \tilde{c}_k > 0, \quad \forall\ k = 0, 1, \ldots, n-1 \end{cases} \tag{7.64}$$

This problem is actually a linear least-squares problem with linear and non-linear inequality constraints. The objective function can be written in the simple form:

$$Q\left(\Delta f\right) = \underline{\delta}^t\, I_n\, \underline{\delta}$$

where $I_n$ denotes the $n \times n$ identity matrix. Then, the problem (7.64) can be solved by using special numerical methods for constrained linear least-squares problems [10], based on different techniques such as QR decomposition, or generalised singular value decomposition. However, in general, the constraints $a_i > 0$ are given as a fraction where the numerator and denominator are algebraic expressions of the parameters $\tilde{c}_i$ raised to a power up to $n$. As the degree $n$ of the original polynomial $f(s)$ increases, the complexity of the problem (7.64) increases drastically, and this may cause serious numerical problems that could lead to erroneous results.

But, because of the iterative nature of the RH-ERES algorithm, it is possible to simplify the constraints $a_i > 0$ to a non-fractional form and thus reduce the complexity of the problem. This procedure is described in the following and we will show that the minimisation problem (7.64) is equivalent to a more efficient minimisation problem with simplified constraints.

**Theorem 7.6.** *Given a real nominal and unstable polynomial*

$$f(s) = s^n + \sum_{k=0}^{n-1} c_k\, s^k, \quad c_k \in \mathbb{R}$$

*the minimum distance to a stable polynomial*

$$\widetilde{f}\,(s) = s^n + \sum_{k=0}^{n-1} \tilde{c}_k\, s^k, \quad \tilde{c}_k \in \mathbb{R}$$

*is given by the minimisation problem:*

$$\begin{cases} \min \left\{ \sum_{k=0}^{k=n-1} (c_k - \tilde{c}_k)^2 \right\} \\ C_k > 0, \quad \forall\ k = 2, \ldots, n-1 \\ \tilde{c}_k > 0, \quad \forall\ k = 0, 1, \ldots, n-1 \end{cases} \tag{7.65}$$

*where the terms $C_k$ are computed by applying the RH-ERES algorithm to $f(s)$.*

*Proof.* We will use the perturbed polynomial $\widetilde{f}(s)$ as an input polynomial to the RH-ERES algorithm and, since it is assumed to be stable, the following conditions must hold (Corollary 7.1):

$$\tilde{c}_k = c_k - \delta_k > 0 \quad , \qquad \forall\ k = 0, 1, \ldots, n-1 \tag{7.66}$$

$$a_i = tr(A_i) > 0 \quad , \qquad \forall\ i = 1, 2, \ldots, n \tag{7.67}$$

where $\tilde{c}_k$ are the coefficients of $\widetilde{f}(s)$ and the terms $a_i$ correspond to the continued fraction form (7.24). Furthermore,

$$Q\,(\Delta f) = \delta_{n-1}^2 + \delta_{n-2}^2 + \ldots + \delta_1^2 + \delta_0^2 = \sum_{k=0}^{n-1} (c_k - \tilde{c}_k)^2$$

and thus the problems (7.64) and (7.65) have the same objective function to be minimized. The computed terms $a_i$ from the RH-ERES algorithm has the following general form:

$$a_1 = \frac{1}{C_1}, \quad a_2 = \frac{{C_1}^2}{C_2}, \quad a_3 = \frac{{C_2}^2}{C_1 C_3}, \quad a_4 = \frac{{C_3}^2}{C_2\, C_4}, \ldots$$

$$a_{n-1} = \frac{C_{n-2}^2}{C_{n-3}\, C_{n-1}}, \quad a_n = \frac{C_{n-1}}{C_{n-2}\, C_n}$$

The terms $C_i$ can be computed along with the terms $a_i$ from the RH-ERES algorithm. Specifically, we have:

$$C_0 = 1, \quad C_1 = c_{n-1}, \quad C_n = c_0$$

and

$$C_k = \frac{C_{k-1}^2}{C_{k-2}\, a_k}, \text{ for } k = 2, 3, \ldots, n-1$$

Since we want $a_i > 0$, it is necessary and sufficient to have $C_i > 0$ for all $i = 1, 2, \ldots, n$. Therefore, instead of using the rational terms $a_i$ in the minimisation problem (7.64), we can use the terms $C_i$ which simplify the constraints and consequently reduce the complexity of the problem. □

The next example describes a procedure for calculating the minimum distance of an unstable to a stable polynomial by solving the optimisation problem (7.65).

**Example 7.7.** Consider the polynomial:

$$f(s) = s^4 + s^3 + 2\,s^2 + 2\,s + 1$$

with $c_0 = 1,\ c_1 = 2,\ c_2 = 2,\ c_3 = 1$. When we apply the RH-ERES algorithm to $f(s)$, the next sequence is produced:

$$\mathcal{A}_f = \left\{1,\ \frac{1}{\epsilon},\ \frac{\epsilon^2}{2\,\epsilon - 1},\ \frac{2\,\epsilon - 1}{\epsilon}\right\}$$

- For $\epsilon \to 0^+$ : $\quad sign\{\mathcal{A}_f\} = \{+, +, -, -\} \quad \Rightarrow\ r_f^+ = 2$
- For $\epsilon \to 0^-$ : $\quad sign\{\mathcal{A}_f\} = \{+, -, -, +\} \quad \Rightarrow\ r_f^+ = 2$

regardless of the sign of $\epsilon$, the number of roots with positive real parts is $r_f^+ = 2$. Therefore, the given polynomial $f(s)$ is unstable. We will attempt to find the minimum distance of $f(s)$ from a perturbed stable polynomial $\widetilde{f}(s)$ and compute the perturbation $\Delta f(s)$.

Let the perturbed polynomial be:

$$\widetilde{f}(s) = f(s) - \Delta f(s) = s^4 + \tilde{c}_3\, s^3 + \tilde{c}_2\, s^2 + \tilde{c}_1\, s + \tilde{c}_0 \tag{7.68}$$

and

$$\tilde{c}_0 = 1 - \delta_0, \quad \tilde{c}_1 = 2 - \delta_1, \quad \tilde{c}_2 = 2 - \delta_2, \quad \tilde{c}_3 = 1 - \delta_3 \tag{7.69}$$

Then, the performance index is:

$$Q\,(\Delta f) = \sum_{k=0}^{k=3} (c_k - \tilde{c}_k)^2 = {\delta_3}^2 + {\delta_2}^2 + {\delta_1}^2 + {\delta_0}^2$$

In the present case, $n = \deg\{f(s)\} = 4$, and if we apply the RH-ERES algorithm to the polynomial $\widetilde{f}(s)$, we get the sequence:

$$\mathcal{A}_{\widetilde{f}} = \{a_i = tr(A_i),\ i = 1, 2, 3, 4\}$$

where

$$a_1 = \frac{1}{\tilde{c}_3}, \qquad a_2 = \frac{(\tilde{c}_3)^2}{\tilde{c}_2\tilde{c}_3 - \tilde{c}_1},$$

$$a_3 = \frac{(\tilde{c}_2\tilde{c}_3 - \tilde{c}_1)^2}{\tilde{c}_3\left(\tilde{c}_1\tilde{c}_2\tilde{c}_3 - (\tilde{c}_1)^2 - (\tilde{c}_3)^2\tilde{c}_0\right)}, \qquad a_4 = \frac{\tilde{c}_1\tilde{c}_2\tilde{c}_3 - (\tilde{c}_1)^2 - (\tilde{c}_3)^2\tilde{c}_0}{(\tilde{c}_2\tilde{c}_3 - \tilde{c}_1)\,\tilde{c}_0}.$$

Then,

$$C_1 = \tilde{c}_3, \qquad C_2 = \tilde{c}_2\tilde{c}_3 - \tilde{c}_1,$$

$$C_3 = \tilde{c}_1\tilde{c}_2\tilde{c}_3 - (\tilde{c}_1)^2 - (\tilde{c}_3)^2\tilde{c}_0, \qquad C_4 = \tilde{c}_0.$$

Thereafter, if we substitute $\tilde{c}_k = c_k - \delta_k$ for all $k = 0, 1, 2, 3$ , we will have to solve the following minimisation problem:

$$\begin{cases} \min\left\{\sum_{k=0}^{k=3}(c_k - \tilde{c}_k)^2\right\} \\ C_k > 0, \quad \forall\ k = 2, 3 \\ \tilde{c}_k > 0, \quad \forall\ k = 0, 1, 2, 3 \end{cases} \tag{7.70}$$

The above minimisation problem (7.70) can be seen as a linear least-squares problem with non-linear inequality constraints. This problem can be solved by using the routine `LSSolve` from the special package `Optimization` of Maple which includes a built-in library of optimization routines provided by the Numerical Algorithms Group (NAG). This library performs its computations in floating-point arithmetic, and the `LSSolve` routine basically uses an iterative modified Gauss-Newton method to compute a local minimum of a given objective function. As the `LSSolve` requires, the objective function for the least-squares problem (7.70) is given in the form:

$$Q\left(\Delta f\right) = \sum_{k=0}^{3}(c_k - \tilde{c}_k)^2 = \left\|\underline{c} - I_n\,\underline{\tilde{c}}\right\|_2^2 \quad \Leftrightarrow$$

$$Q\left(\Delta f\right) = \left\| \begin{bmatrix} 1 \\ 2 \\ 2 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{c}_0 \\ \tilde{c}_1 \\ \tilde{c}_2 \\ \tilde{c}_3 \end{bmatrix} \right\|_2^2$$

The produced results show that the minimum value of $Q\left(\Delta f\right)$, i.e. the minimum distance of $f(s)$ from the set of stable polynomials, in 16-digit precision,

is

$$\gamma_{f^+} = 8.05133349284929689 \cdot 10^{-2}$$

for

$$\begin{aligned} \tilde{c}_0 &= 0.891379697058926057, & \tilde{c}_1 &= 1.89894721183885218, \\ \tilde{c}_2 &= 2.17708868646499587, & \tilde{c}_3 &= 1.16475101167653827. \end{aligned}$$

Consequently, the minimum perturbation is found to be:

$$\begin{aligned} \Delta f(s) &= -0.164751011676538\, s^3 - 0.177088686464996\, s^2 \\ &\quad +0.101052788161148\, s + 0.1086203029410739 \end{aligned}$$

which corresponds to the perturbed polynomial:

$$\begin{aligned} \widetilde{f}(s) &= s^4 + 1.16475101167653827\, s^3 + 2.17708868646499587\, s^2 \\ &\quad +1.89894721183885218\, s + 0.891379697058926057 \end{aligned}$$

according to the formulae (7.68) and (7.69).

Now, we apply again the RH-ERES algorithm to $\widetilde{f}(s)$ and, after a singularity of type 2 in the second iteration, we finally get the sequence:

$$\mathcal{A}_{\widetilde{f}} = \{\ a_1 = 0.8585525917,\ a_2 = 2.130346045,\ a_3 = 0.5,\ a_4 = 1.226733429\ \}$$

Since there are no negative elements in $\mathcal{A}_{\widetilde{f}}$, all the roots of $\widetilde{f}(s)$ have negative real parts, and therefore, it is verified that it is stable. □

## 7.5 Discussion

In this chapter the problem of the evaluation of the stability of a linear time-invariant system has been considered. The current study was focused on the Routh-Hurwitz stability criterion and the representation of real polynomials in a continued fraction form, which formed the basis for creating a new matrix-based method for assessing the stability of a linear system. This method, which is referred to as the RH-ERES method, is based on the ERES methodology and uses the ERES operations to transform appropriately a basis matrix which corresponds to a special rational form of the characteristic polynomial of a linear system. The sequence of the traces of the matrices $A_i$, which perform the ERE row transformations on the basis matrix, reveal an algebraic relationship which characterise the stability of the linear system.

In normal cases, the developed RH-ERES algorithm computes the coefficients $a_i$ that are necessary to form a continued fraction representation (7.19) for the

characteristic polynomial of a linear system. The stability of the linear system is deduced from the distribution of the roots of the characteristic polynomial on the complex plane, which is determined by the signs of these coefficients. However, in singular cases the algorithm continues according to Routh's rules, which are also used in Routh's algorithm (Routh array). The RH-ERES algorithm produces equivalent results with Routh's algorithm, but it works faster than Routh's tabulation process, especially when the degree of the polynomial is high. This fact becomes a significant advantage for the RH-ERES algorithm when symbolic computations are used in order to study the stability of a linear system, or the distribution of the roots of the characteristic polynomial, in theory.

The RH-ERES algorithm was also applied to the problem of finding the minimum distance of an unstable polynomial from the "stability domain" (i.e. the set characterising all stable polynomials). The current computational approach has been based on the properties of the Routh-Hurwitz criterion in order to formulate an appropriate optimization problem for calculating the minimum norm stabilisation. The use of the RH-ERES method helped to transform the original problem to a linear least-squares problem with simplified constraints which provides the required minimum norm stabilisation for unstable polynomials.

# Chapter 8

# Conclusions and future work

In this study we have presented and analysed the principles of the ERES method which formed the theoretical basis for the development of other different methods and algorithms for solving problems requiring approximate algebraic computations to be solved. The ERES method was initially proposed in [40, 57] as an iterative matrix-based method for the numerical computation of the GCD of sets of many polynomials in one variable with real coefficients. The method takes advantage of the invariance of the GCD under elementary row operations and shifting. These types of operations are applied iteratively on a basis matrix formed directly from the coefficient vectors of the polynomials of the original set. Finally, they lead to a unity rank matrix where theoretically every row of this rank-1 matrix gives the vector of coefficients of the GCD of the set. However, there were several theoretical and practical aspects of the method that had not been analysed in depth until now. The conclusions of this research and the objectives that have been achieved are summarised in the following.

1. **The algebraic representation of the Shifting operation for matrices and the ERES representation of the GCD.**

   A key problem in the present research was to prove that the ERES method is numerical stable not only for a single iteration of its main procedure as in [57], but for all the performed iterations. Therefore, an overall algebraic representation of the ERES method was crucial to be established in order to study its numerical stability in more detail. The algebraic representation of the ERES method, which has been established in Chapter 3, requires the Shifting operation to be written as a matrix product, just like the elementary row operations, so as to have an algebraic equivalence between the initial basis matrix and the last unity rank matrix. Although it is evident that the ERES operations preserve the GCD of a set of polynomials, in the previous work [57, 58] it was not clear how the iterative steps of the method connect to each other, because the Shifting operation alters the

column structure of the matrix in process. The shifting of the elements in the rows of a matrix, as used in the ERES method, is not a common transformation, but in essence it is an error-free transformation which can be implemented in a programming environment without using arithmetic operations. Nevertheless, the careful analysis in Chapter 3 has shown that the Shifting operation, applied to upper trapezoidal matrices with full rank, can be actually represented by a matrix product. This important theoretical result is based on the invertibility property for matrices and it has a direct application to the ERES representation of the GCD, because in every iteration of the main procedure of the ERES method, a nonsingular (invertible) upper trapezoidal matrix is involved. However, considering any type of real matrix, the algebraic representation of the Shifting operation, as described in Definition 3.7 and without modifying the original data of the matrix, is an issue which requires further study.

2. **The ERES representation of Euclid's division algorithm.**

   The study of the properties of the ERES method led also to the investigation of the link between the ERES operations and the Euclidean division of two polynomials, which brought about: a) the ERES representation of the remainder and quotient of the division of two polynomials, and b) the development of an ERES-based algorithm for computing the quotient and the remainder of the division of two polynomials, which is referred to as the ERES Division algorithm. The provided ERES representation of the Euclidean division suggests that the ERES method is actually the equivalent of Euclid's algorithm for several polynomials and the GCD is the total quotient. Therefore, we may view the ERES method as a generalisation of Euclid's division of two polynomials to many polynomials, simultaneously.

3. **Formulation of the PSVD1 method for the smart detection of a unity rank matrix.**

   The numerical computation of the GCD of a set of polynomials with the ERES method requires the development of a robust algorithm which must consist of numerically stable algebraic processes and use an efficient termination criterion. The termination of the ERES algorithm is based on the proper numerical detection of a rank-1 matrix during the iterations of the main procedure of the algorithm. Under certain conditions, this criterion (Proposition 4.1) relies on the singular value decomposition (SVD) of the processed matrix. Therefore, in the present implementation of the ERES algorithm a variation of the Partial SVD algorithm [75, 76] has been developed. The introduced PSVD1 algorithm, as presented in Chapter 4, is suitable for checking matrices with numerical rank equal to 1. Its use reduces the overall

computational cost and significantly improves the performance for the ERES algorithm. In addition, through the PSVD1 algorithm we can get different estimates of the numerical tolerance that we have to use for computing an approximate GCD. Therefore, having a set of several polynomials, it is now possible to smartly compute more than one approximate GCDs of various degrees for different values of the numerical tolerance $\varepsilon_t$.

4. **Implementation of the ERES method in a hybrid computational environment for computing the approximate GCD.**

   The ERES algorithm can be implemented in any programming environment using stable and well known numerical processes like Gaussian elimination with partial pivoting, Partial Singular Values Decomposition, and Normalisation according to the Frobenius norm. The main advantages of the ERES method is that it starts with a basis matrix with no larger dimensions than those which are implied by the original set of polynomials. In addition, the successive triangularisations and Shifting lead to a fast reduction of the dimension of the initial basis matrix, which increases the processing of the data. These features helps the ERES algorithm to be economical in memory bytes and faster than other methods which tend to create too large initial matrices without further reduction of their dimensions. However, due to its iterative nature, extra care must be taken when using floating-point data. The present implementation of the ERES algorithm is based on the effective use of symbolic-numeric (hybrid) computations, resulting in a more efficient and numerically stable algorithm, which is referred to as the Hybrid ERES algorithm. The use of hybrid computations helps to reduce the accumulation of unnecessary numerical rounding errors, but also enables computations within a specified tolerance in specific parts of the algorithm in order to get approximate solutions. The iterative nature of the ERES method and the use of hybrid computations makes it a useful mathematical tool in computing approximate GCDs of a given set of many polynomials.

5. **Improvement of the existed strength criterion for the evaluation of the quality of an approximate GCD.**

   Numerical procedures, such as ERES, always produce estimates of an exact solution. Estimating the size of the perturbations in the original data provides the means to evaluate how good approximations are produced. The investigation of the approximate GCD for a set of several polynomials has been analysed in [22, 42] and the overall approach has been based on its characterisation as a distance problem in a projective space. The study of this problem has led to the definition of the *strength* of an approximate GCD and its evaluation by using optimisation techniques. The strength of

an approximate GCD is a very important indicator, which gives information about the quality of the produced approximations. However, the method for the evaluation of the strength of an approximate GCD, as proposed in [22, 42], requires the computation of a global minimum, where in case of large sets of polynomials it is very likely to give unsatisfactory results. Alternatively, it is easier to get information about the quality of a given approximation by computing some tight bounds for the strength and compute the *average strength*. A method for computing the strength bounds and the average strength has been presented in Chapter 5. The main characteristic of this method is that it exploits the properties of resultant matrices in order to produce meaningful results without using optimisation routines. The combination of the Hybrid ERES algorithm and the Average Strength algorithm suggests a complete procedure for the computation and evaluation of an approximate GCD of a set of several polynomials.

6. **Formulation of the Hybrid LCM method for computing an approximate LCM.**

   The analysis of the ERES method provided also the means for computing the LCM of a set of several polynomials. In Chapter 3, the ERES operations were appropriately used to represent the remainder of the Euclidean division of two polynomials. This representation has played an important role in the development of a new matrix-based method for computing the coefficients of an approximate LCM. The new LCM method is actually based on the fact that every polynomial of a set must divide evenly into LCM and thus the remainder of the division must be equal to zero. The method has two stages: i) the use of the ERES Division algorithm to symbolically compute the remainder of the division of the LCM in an arbitrary symbolic form, by each of the polynomials of the original set, and ii) the formulation of an homogeneous system of linear equations with unknowns the coefficients of the LCM. The initial matrix of the system created, has no greater dimensions than the degree of the LCM implies and the solution is given by solving a linear least-squares optimisation problem. The quality of the obtained solution depends on the proper handling of the type of data and the proper method to solve the final least-squares problem. The developed algorithm, which is referred to as *Hybrid LCM* algorithm, is implemented in a hybrid computational environment, because the first stage of the algorithm involves computations with arbitrary variables and, on the other hand, the least-squares solution requires floating-point numerical operations for computing an approximate LCM within a specified numerical tolerance. Conclusively, this method has three main advantages: a) it avoids the computation of roots, b) it does not require the computation of the GCD of the polynomials

of the given set, and c) it produces estimates of approximate LCMs. The latter is of great interest when the initial data are given inexactly; then the quality of the given approximation is determined by the residual of the least-squares solution.

7. **Formulation of the RH-ERES method for evaluating the stability of a linear system.**
   Furthermore, the ERES methodology appeared to have a useful application in the representation of continued fractions. This was the motivation to study an alternative approach for assessing the stability of linear time-invariant systems through the characteristic polynomial. The continued fraction representation of the characteristic polynomial and the properties of the Routh-Hurwitz stability criterion formed the basis for creating an alternative matrix-based method for the evaluation of the stability of a linear system. This method, which is referred to as the RH-ERES method, uses the ERES operations to transform a basis matrix which corresponds to a special rational form of the characteristic polynomial of a given linear system. The sequence of the traces of the matrices $A_i$, which perform the ERE row transformations on the basis matrix, reveal an algebraic relationship which characterises the stability of the linear system. The developed RH-ERES algorithm is implemented in a hybrid computational environment and produces very accurate results. Moreover, the RH-ERES method has been applied to the problem of finding the minimum distance of an unstable to a stable polynomial, resulted in a least-squares problem with simplified constraints.

ERES is a simple and effective method and, although it was originally developed for the computation of the GCD of sets of several polynomials, its use can be extended to other algebraic problems and applications, and possibly provide us with new better results. However, the success of an algebraic method, such as ERES or any other computational method, depends greatly on how it is implemented in a computing environment. In recent years, there has been a significant change in the area of numerical analysis with the presence of mathematical software packages that combine symbolic and numerical floating-point arithmetic systems. This has changed a lot the way of implementing numerical methods and motivated the construction of sophisticated algorithms that allow the use of symbolic and numerical data through appropriate data structures. Under certain conditions, the simultaneous use of symbolic and floating-point operations (hybrid computations) improves the accuracy of the obtained results by reducing the accumulation of rounding errors, and also preserves the ability of computing approximate solutions.

The hybridisation of the ERES method (i.e. the implementation of its algorithm by using hybrid computations) was based on the requirement to reduce the accumulation of rounding errors during the course of the main iterative procedure, especially from the Gaussian elimination. Simultaneously, it was crucial to maintain the ability of the algorithm to produce approximates results. Therefore, the structure of the Hybrid ERES algorithm is based on the appropriate separation of the procedures which form the algorithm, so as to meet the previous requirements. The current separation of procedures follows the natural structure of the method, which involves two main parts: i) the iterative transformation of the basis matrix, and ii) the frequent check of the termination criterion. Numerous tests have proven that the current formulation of the Hybrid ERES algorithm exploits to the maximum the special structural properties of the ERES method and results in a nearly optimal hybrid algorithm for this method. The same concept lies beneath the formulation of the Hybrid LCM algorithm, which also involves two main parts that are naturally separated. However, the other two ERES-based algorithms, the ERES Division and the RH-ERES algorithm, which are developed in this research, can be characterised as pure symbolic algorithms, because there is no need to produce approximate results and arbitrary variables can be used freely. Therefore, these algorithms may become quite useful computational tools for the theoretical study of many related problems.

► **Further research**

**Optimal hybridisation of an algorithm.** The proper hybridisation of an algorithm, is an issue that is not entirely clear. From our study so far, when it comes to the implementation of methods and algorithms, there are three important questions to be answered: a) When it is necessary to use hybrid computations? b) Is it effective to use hybrid computations? c) How can the method be hybridised in an *optimal* way?

The answer to the first two questions is rather simple. We can use pure symbolic or hybrid computations when we want to increase the accuracy of the obtained solution without messing with the system's variable floating-point precision. Or else, we can use hybrid computations in order to implement parts of the method separately, so as to avoid the accumulation of rounding errors and at the same time to maintain the ability of computing approximate solutions. Of course, hybrid computations can be used any time, but it is not always effective to use them, especially when we have large amounts of numerical data to be processed. In this case, symbolic computations may not be effective at all, since they require more computational time to be executed. Practically, numerical floating-point computations in high precision (quadruple precision) is proved to be more effective for algorithms with sequential procedures involving only numerical

data. The last question cannot be answered directly. In general, we could define as *optimal hybridisation* the implementation of the algorithm in a symbolic-numeric computational environment so as to produce the "best" possible results in a "short" period of time. However, this is an issue with multiple parameters that should be examined very carefully in future research in order to develop *the* optimal hybrid algorithm for the ERES and other related methods.

**Generalisation of the Shifting operation for matrices.** The Shifting transformation is a crucial part of the ERES method which is responsible for reducing the size of the initial basis matrix. The matrix representation for the Shifting operation, established in Theorem 3.4, refers to nonsingular real upper trapezoidal matrices and has a direct application to the overall matrix representation of the ERES method. The developed representation of the Shifting operation is based on the invertibility property of the original matrix and therefore cannot be directly applied to rank deficient matrices. However, it could be applied to a proper submatrix of the original matrix, but this would also change completely some of the data of the shifted matrix. The essence of the current procedure is to form a new vector from the diagonal elements of a square matrix without changing their values, which is equivalent to form and theoretically study the mapping:

$$A \longmapsto \mathrm{diag}\{A\}$$

Hence, considering any type of real matrix, the algebraic representation of the Shifting operation, as described in Definition 3.7 and without modifying the original data of the matrix, is currently an issue which remains open. The Shifting operation is a rare matrix transformation which can be useful to other algebraic problems, such as the downsizing of expanded Sylvester matrices [73], and its theoretical aspects require further investigation.

**Optimal approximate common factors.** The strength problem, as described in Chapter 5, can be used not only for evaluating the quality of a given approximate GCD, but also for computing an approximate GCD of a fixed degree by using a proper optimization method. The process that was followed in Example 5.3 suggests an heuristic method for computing approximate common factors which is based on the average strength of an arbitrary simple common factor of the form $v(s) = s + c \in \mathbb{R}[s]$, $c \in \mathbb{R} \setminus \{0\}$ (or the scaled form $v(s) = c\,s + 1$). More particularly, the method relies on the minimisation of the derived objective function $\mathcal{S}_v^a(c)$ which corresponds to the average strength of the factor $v(s)$ for a given set $\mathcal{P}_{h+1,n}$. However, this heuristic method is not capable of providing sufficient solutions to the approximate GCD problem in general; for example, in the case of approximate common factors of degree equal or greater than 2, without

real roots. Therefore, further investigation for a more robust technique is required.

An issue that arises from the study so far is the existence of approximate common factors which can be characterised as *optimal approximate GCDs*. Regarding sets of several polynomials, this issue requires a thorough study of the properties of the optimization problem (5.16). What is absolutely necessary is to find a more efficient process of computing a certified global minimum. The obtained solution can be regarded as the *optimal strength* of a given approximation.

In the present research, we proved that the objective function of the optimisation problem (5.16):

$$\|\mathcal{S}_Q\|_F \triangleq \left\| S_{\mathcal{P}} - \left[\mathbb{O}_{m,r} | \tilde{S}^{(r)}_{\mathcal{P}^*}\right] \cdot \hat{\Phi}_v \right\|_F$$

has certain bounds:

$$\underline{\mathcal{S}}(v) \leq \|\mathcal{S}_Q\|_F \leq \overline{\mathcal{S}}(v)$$

An exact common factor of a set of polynomials actually zeros the function $\|\mathcal{S}_Q\|_F$. Therefore, a given approximate common factor may be considered as "optimal", if it has the least strength amongst all the approximate common factors of the same degree. This can be extended to the approximate GCD case, provided that a maximum polynomial degree of the GCD is certified. However, the computation of the optimal approximate GCD of a set of several polynomials is a problem that needs further investigation with critical issue the search for a robust optimisation method, which can guarantee the existence of a global minimum.

Moreover, the issues and the results presented in Chapters 5 and 6 provide the motives for further research on the subject of the approximate LCM of sets of univariate polynomials which involves the search for an optimal combination of symbolic and numerical computations as well as the selection of proper optimisation methods, which eventually will set the basis for the computation of an *optimal approximate LCM* for sets of many univariate polynomials.

**Application of the ERES method to multivariate polynomials.** The Hybrid ERES algorithm can also be used for the computation of the GCD of a set of multivariate polynomials, if we just change the procedure that constructs the initial basis matrix. Considering the case of sets $\mathcal{P}_{m,n,r}$ of $m$ polynomials in two variables $(s,t)$ (bivariate polynomials) with coefficients in rational numbers $\mathbb{Q}$, the basis matrix $P_m$ can be formed according to the bivariate power basis:

$$\mathcal{E}_{n,r}(s,t) = \{(1,t,\ldots,t^r),(s,st,\ldots,st^r),\ldots,(s^n,s^nt,\ldots,s^nt^r)\} \tag{8.1}$$

where $n, r$ are the maximum powers of the variables $s$, $t$, respectively. The dimension of the corresponding basis vector $\underline{e}_{n,r}(s,t)$ is equal to $(n+1)\cdot(r+1)$. Similar base vectors can be formed for polynomials in several variables. The

produced matrix $P_m$ is structured according to the bivariate power basis vector:

$$\underline{e}_{n,r}(s,t) = \left[\begin{array}{cccc} 1 & 0 & \dots & 0 \\ t & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ t^r & 0 & \dots & 0 \\ \hline 0 & 1 & \dots & 0 \\ 0 & t & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & t^r & \dots & 0 \\ \hline \vdots & \ddots & \ddots & \vdots \\ \hline 0 & \dots & 0 & 1 \\ 0 & \dots & 0 & t \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & t^r \end{array}\right] \cdot \left[\begin{array}{c} 1 \\ s \\ \vdots \\ s^n \end{array}\right] = \left[\begin{array}{c} 1 \\ t \\ \vdots \\ t^r \\ \hline s \\ st \\ \vdots \\ st^r \\ \hline \vdots \\ \hline s^n \\ s^n t \\ \vdots \\ s^n t^r \end{array}\right]$$

The column dimension of the basis matrix $P_m$, is equal to $(n+1)\cdot(r+1)$, where $n,\ r$ denote the maximal powers of $s,\ t$, respectively. For example, consider the set of bivariate polynomials:

$$\mathcal{P}_{3,2,2} = \left\{ \begin{array}{rcccl} p_1(s,t) & = & (s-3\,t+1)(t-1) & = & st-s-3\,t^2+4\,t-1 \\ p_2(s,t) & = & (s-3\,t+1)(s-2) & = & s^2-s-3\,st+6\,t-2 \\ p_3(s,t) & = & (s-3\,t+1)(s\,t-3) & = & s^2t-3\,s-3\,t^2s+9\,t+st-3 \end{array} \right\}$$

with GCD, $g(s) = s-3\,t+1$. The maximal power of the variable $s$ is $n=2$ and the maximal power of the variable $t$ is also $r=2$. Then, the initial basis matrix $P_m$ for $m=3$ and the basis vector of the variables $s,\ t$ are:

$$P_3 = \left[\begin{array}{ccccccccc} -1 & 4 & -3 & -1 & 1 & 0 & 0 & 0 & 0 \\ -2 & 6 & 0 & -1 & -3 & 0 & 1 & 0 & 0 \\ -3 & 9 & 0 & -3 & 1 & -3 & 0 & 1 & 0 \end{array}\right] \in \mathbb{Q}^{3\times 9}$$

$$\underline{e}_{2,2}(s,t) = \left[\ 1, t, t^2, s, st, st^2, s^2, s^2t, s^2t^2\ \right]^t$$

If we apply the ERES operations to $P_3$, we finally get the vector of coefficients:

$$\underline{g} = [1, -3, 0, 1, 0, 0, 0, 0, 0]$$

which obviously gives the GCD of the set $\mathcal{P}_{3,2,2}$ ,

$$\gcd\{\mathcal{P}_{3,2,2}\} = \underline{g}\cdot\underline{e}_{2,2}(s,t) = s-3t+1$$

The Hybrid ERES algorithm can compute the GCD of bivariate polynomials,

if we change the form of the basis matrix which is the main input. However, a proper framework for the algebraic and geometric properties of the GCD of sets of many polynomials in a multidimensional space has to be set in order to define and evaluate exact or approximate solutions given by the ERES method. This problem is challenging for further research, because several real-time applications, such as image and signal processing, rely on GCD methods where multivariate polynomials (especially in two variables) are used.

**Extension of the application of the ERES methodology.** The development of ERES-based methods and algorithms for computing solutions of Diophantine equations, expressing partial fraction expansion and Padé approximations, or computing matrix divisors, are challenging problems for further research, which can significantly contribute to the study of broader problems, such as the possible extension of the framework to families of polynomials where the coefficients come from a certain interval, the extension of the definition of *almost GCD* to the definition of *approximate matrix divisor*, and the distance of a system to *almost uncontrolability* and *almost unobservability*.

# Appendix A

# Codes of Algorithms

All the algorithms presented in this thesis were implemented and tested in "Maple". Maple is a general-purpose commercial computer algebra system which was first developed in 1980 by the Symbolic Computation Group at the University of Waterloo in Waterloo, Ontario, Canada. The Maple computation engine combines high-performance numeric computations with exceptional symbolic capabilities. Maples hybrid system provides many advantages :

- It allows us to work with exact quantities such as fractions, radicals, and symbols, eliminating accumulated round-off errors.
- Approximations can be computed at any precision that is required, and are not restricted by hardware limitations.
- We can choose from a variety of approximate and exact techniques, as best suits our needs.
- Maple allows us to defer numeric approximations until they are needed, using symbolic parameters in our problem instead. The parameters are then carried through each stage in our analysis, making it easy to do parameter sweeps, optimize values, and study the behavior of the system.
- Symbolic computations allow us to obtain highly accurate results, eliminate the need to simplify problems by hand, and provide insight into our problem structure from which we can develop conjectures and conclusions about the behavior.
- Internally, Maples solvers can also use a combination of symbolic and numeric techniques, allowing it to solve problems for which either approach alone would be insufficient.
- There is extensive support for numeric computations, to arbitrary precision, as well as symbolic computation and visualization.

Maple incorporates a dynamically typed imperative-style programming language which is based on a small kernel, written in C, which provides the Maple language. Many numerical computations are performed by the NAG Numerical Libraries, ATLAS libraries, or GMP libraries. The following codes of algorithms are written in Maple language using internal data structures and built-in routines from the package `LinearAlgebra`, which provides several matrix-based procedures such as Gaussian elimination, Singular Value Decomposition, Least-Squares minimisation and many others. The following procedures are specially designed for the purposes of this thesis and they are not listed in any package of Maple.

# A.1 The code of algorithms based on ERES

## A.1.1 The procedure `ERESDivision`

The procedure `ERESDivision` computes symbolically the exact division of two univariate polynomials and it corresponds to the algorithm 3.1.

**Primary input parameters :**

A, B : Numeric univariate polynomials. (Type: `polynom`)

**Output parameters :**

(Q, R): A pair of vectors corresponding to the Quotient and the Remainder of the division. (Type: sequence of `Vector`)

**Maple code :**

```
ERESDivision := proc( A::polynom(anything), B::polynom(anything) )

description"The procedure gives the result of the division A/B
of two polynomials with degrees m and n, where m > n.";

local a,b,i,j,k,m,n,t,s,K,M,N,P,Q,R,RN,T,t1,output;

   s := op(1, indets( A ) );

#  Initialize vectors of input polynomials A(s), B(s).

   a := ListTools:-Reverse( PolynomialTools[CoefficientList](A,s) );
   b := ListTools:-Reverse( PolynomialTools[CoefficientList](B,s) );

#  Determine dimensions of initial basis matrix P.
```

```
   m := degree( A, s );
   n := degree( B, s );

   if m <= n then
      error" the degree of the 1st polynomial must be greater than
          the degree of the 2nd polynomial."
   end if;

#  Initialize basis matrix P.

   P := Matrix( 2, m+1, fill=0 );

   for j from 1 to n+1 do
       P[1,j] := b[j];
   end do;

   for j from 1 to m+1 do
       P[2,j] := a[j];
   end do;

#  Initialize vector of quotient.

   Q := Vector[column]( m-n+1, fill=0 );

#  Normalise the rows of P using norm-2.

   P  := normrows( 2, m+1, P );

#  Determine the number of steps of main iterative procedure.

   N := m-n+1;

#  Start of main iterative procedure.

   for i from 1 to N  do

#  Gaussian elimination.

       M := - P[2,1] / P[1,1];
       P[2,1] := 0;
```

```
        for j from 2 to n+1 do
              P[2,j] := P[2,j] + M * P[1,j];
        end do;

        Q[i] := -M;

#  Matrix Shifting.

        T := P[2, 2..m+1];
        P[2, 1..m] := T;
        P[2, m+1] := 0;ND OF PROCEDURE

   end do;

#  End of main iterative procedure.

   R := LinearAlgebra:-Transpose( P[2,1..n] );
   R := simplify( R/Q[1] );
   Q := simplify( Q/Q[1] );
   RN := evalf( LinearAlgebra:-Norm( R, infinity ) );

#  Q = Quotient's coefficients (first element corresponds to x^{m-n}).
#  R = Remainder's coefficients(first element corresponds to x^{n-1}).
#  RN= Remainder's Euclidean norm.

   printf( "Remainder Norm = %e\n",RN );

   if nargs > 2 then
      if args[3] = 'quotient' then
         output := sort( add( Q[i]*x^(m-n+1-i), i=1..m-n+1 ) );
      elif args[3] = 'remainder' then
         output := sort( add( R[i]*x^(n-i), i=1..n ) );
      else
         output := ( Q, R );
      end if;
   else
      output := ( Q, R );
   end if;

   output
```

```
end proc;

# End of procedure ERESDivision
```

### A.1.2 The procedure HEresGCD

The procedure HEresGCD computes in hybrid mode the exact or an approximate GCD of a list of univariate numeric polynomials and it corresponds to the Hybrid ERES algorithm as presented in figure 4.1.

**Primary input parameters :**

P : Initial basis matrix formed from the coefficients of the polynomials of the given set $\mathcal{P}_{m,n}$. (Type: Matrix)

**Secondary input parameters :**

tolS : Numerical accuracy $\varepsilon_t$ applied to the termination criterion of the Hybrid ERES algorithm. (Type: float)

tolG : Numerical accuracy $\varepsilon_G$ applied to the processed matrix in order to control the magnitude of its elements. (Type: float)

digits : Number of digits which determine the system's software accuracy. (Type: posint)

stopit : Specifies a maximum number of iterations for the main procedure of the Hybrid ERES algorithm. (Type: posint)

**Output parameters :**

GCD : The vector of coefficients of the GCD presented in ascending order according to the degree of the main variable of the polynomials. (Type: Vector)

**Maple code :**

```
HEresGCD := proc( P::Matrix )

description" The procedure HEresGCD computes in hybrid mode the exact
or an approximate GCD of a list of univariate numeric polynomials
using Maples package [LinearAlgebra].";

#  Definition of local parameters.

   local i, j, k, m, n, c, a, P, Pf, iter, param , GCD, r, init, w,
         degs, vn, tol,  N, maxdegs, mindegs, rowind, Sval, W, piv,
```

```
        maxi, M, t, t1, T, crit, Rec, eps, err, mintolS, svdtolS;

#  Initial values for local parameters.

   param[digits_] := Digits;
   eps            := evalhf(DBL_EPSILON);
   param[tolS_]   := eps;
   param[tolG_]   := eps;
   param[inf]     := false;
   param[flpdata] := false;
   Rec[tolS_]     := 1.0;
   param[stop_]   := 0;

   for i from 2 to nargs do
       if op( 1, args[i] ) = 'digits' then
          param[digits_] := op( 2, args[i] );
          Digits := param[digits_];
          break;
       end if;
   end do;

# Specify the precision eps of the numeric system.

   if Digits > evalhf(Digits) then
      eps := evalf( 2^(-3*Digits) );
      while evalf(1.0 + eps) > 1.0 do
            eps := evalf( eps/2 )
      end do;
      param[tolG_] := eps;
      param[tolS_] := eps;
   end if;

# Assign values to the local parameters.

   for i from 2 to nargs do
       if op( 1, args[i] ) in
          { 'digits', 'tolS', 'tolG','stopit' } then
          if op( 1, args[i] ) = 'tolS' then
             param[tolS_] := evalf( op( 2, args[i] ) )
          elif op( 1, args[i] ) = 'tolG' then
```

```
            param[tolG_] := evalf( op( 2, args[i] ) )
        elif op( 1, args[i] ) = 'stopit'  then
            param[stop_] := op( 2, args[i] )
        end if;
     else
        error"invalid parameters."
   end if;
end do;

#  Check the dimensions and data type of the initial basis matrix P.

   m, n := LinearAlgebra:-Dimensions( P );
   if hastype( P, float ) = true then
      P := conversion( m, n, P );
      param[flpdata] := true;
   end if;

#  Set the counter of main iterations and SVD calls.

   iter := [0,0];

#  Start the Main iterative procedure.

   while iter[1] >= 0 do

#  Check the degrees of the polynomials
#  and compute of the maximum degree n.

      degs := array(1..m);
      for i from 1 to m do
          j := n;
          while P[i,j] = 0 do
                j := j-1 ;
          end do;
          degs[i] := j;
      end do;
      maxdegs := max( seq(degs[i], i=1..m) );
      mindegs := min( seq(degs[i], i=1..m) );
      n := maxdegs;
      #  Reduce the dimensions of P.
```

```
      P := P[ 1..m, 1..n ];

#  Check whether to continue with the main procedure or
#  proceed to the Rank-1 procedure.

      if mindegs <> maxdegs then

#  Reorder the rows of P in descending order
#  according to the degree of the polynomials.

         rowind := convert( inssort( degs )[2], list );
         init := [ seq( convert( LinearAlgebra:-Row ( P, rowind[i] ),
                   list ), i=1..m ) ];

#  Convert P to rational format.

         P := Matrix( m, n, init, storage=rectangular,
                      datatype=rational,order=Fortran_order,fill=0 );
      else

#  Start Rank-1 procedure.

#  Convert P to numerical (floating-point) format.

         Pf := evalf( P );

#  Normalise the rows of P by using norm-2.

         Pf := normrows( m, n, Pf );

#  Compute the singular values of P by using PSVD1.

         N, Sval, tol, w := psvd1( m, n, Pf, param[tolS_] );
         iter[2] := iter[2] + 1;

#  Determine the appropriate value for the
#  termination criterion.
         if N > 1 then
            crit[1] := -1;
            crit[2] := tol;
```

```
         else
            crit[1] := abs( Sval - evalf( sqrt(m) ) );
            crit[2] := tol;
         end if;

#  Record a new value for tolS and print info.

         Rec[tolS_] := max( crit[1], crit[2] );
         svdtolS[iter[2]] := Rec[tolS_];
         if svdtolS[iter[2]] < 0.1 then
            t := min( seq( max(seq(baseexp(P[i,j]),j=1..n)),i=1..m));
            printf("GCD degree = %d, tolS > %.6e, tolG < %.6e,
                   Iterations = %d \n", n-1, Rec[tolS_], t, iter[1]);
         end if;

#  Decide whether to continue or stop the overall process.

         if N=1 and crit[1] <= param[tolS_] and crit[2]<=param[tolS_]
            or n=1 or m=1 then
            if param[flpdata] = true then

#  Numerical solution obtained from the right singular vector.

               GCD := w;
            else
               maxi := max( seq( P[i,n], i=1..m ) );
               i := 1;
               while P[i,n] <> maxi do
                     i := i + 1
               end do;

#  Rational solution obtained from the last matrix.

               GCD := simplify( LinearAlgebra:-Row ( P, i ) );
            end if;
            if iter[1] >= param[stop_] then
               break;
            end if;
         end if;
      end if;
```

```
#  Apply Scaling to the rows of P in order to make P[1,1] the
#  maximum element (in absolute value) in the first column.

      maxi := max( evalf( seq( abs(P[i,1]), i=1..m ) ) );
      if evalf( abs(P[1,1]) ) <= evalf( maxi ) then
         t1 := baseexp( P[1,1] );
         t  := baseexp( maxi );
         LinearAlgebra:-RowOperation( P , 1, (10*t)/t1 );
      end if;

#  Apply Gaussian elimination with partial pivoting to P.

      P := LinearAlgebra:-GaussianElimination( P );

#  Eliminate the zero rows according to the tolerance tolG.

      i := m;
      while i > 1 do
            maxi := max( evalf( seq( abs(P[i,j]), j=1..n ) ) );
            if evalf( maxi ) <= param[tolG_] then
               P := LinearAlgebra:-DeleteRow ( P, i );
               m := m-1;
               i := i-1;
            else
               i := i-1;
            end if;
      end do;

#  Eliminate all the elements of P which are approximately zero
#  according to the tolerance tolG.

      if param[tolG_] <> eps then
         for i from 2 to m do
             for j from i to n do
                 if evalf( abs(P[i,j]) ) < param[tolG_] then
                    P[i,j] := 0
                 end if;
             end do;
         end do;
      end if;
```

```
#  Apply Shifting to the rows of P.

      P := shifting( m, n, P );
      iter[1] := iter[1] + 1;

   end do;

#  End of Main iterative procedure.
#  Computational information.

   for i from 1 to m do
       P[i,1..n] := map( x -> x / P[i,1..n][n], P[i,1..n] );
   end do;
   GCD := map( x -> x / GCD[n], [1..n] );
   err[1] := evalf( seq( LinearAlgebra:-Norm( P[i,1..n] - GCD , 2),
                    i=1..m ) );
   err[2] := ( max( err[1] ) + min( err[1] ) ) /2;
   mintolS := min( seq( svdtolS[i], i=1..iter[2] ) );
   printf("Parameters = { tolS-> %e, tolG-> %e, digits-> %d } \n",
           param[tolS_], param[tolG_], param[digits_] );
   printf("Statistics = { Iterations = %d, SVDcalls = %d}\n",
           iter[1],iter[2] );
   printf("Distance from rational solution = %e,
              Minimum termination tolerance = %e", err[2], mintolS);
   end if;

#  Final output.

   GCD

end proc;

#  End of procedure HEresGCD.
```

### A.1.3 The procedure `SREresLCM`

The procedure `SREresLCM` computes the LCM of a set of several univariate polynomials. The following procedure is the implementation of the algorithm 6.1 and involves the procedure `HEresGCD` for the computation of the GCD and `ERESDivision` for the computation of the LCM.

**Primary input parameters :**

P : Initial set of polynomials $\mathcal{P}_{m,n}$. (Type: `list(polynom)`)

**Output parameters :**

Ls : The LCM presented as polynomial. (Type: `polynom`)

**Maple code :**

```
SREresLCM := proc( P::list(polynom(anything)) )

description"The procedure computes the LCM of polynomials by using
the ERES and ERESDivision procedures in symbolic-rational mode.";

local h, Q, w, p, k, T, Ps, Gs, Ls;

#  Detect the number of polynomials.

   h  := nops( P );

#  Compute the (h-1)-combinations.

   Q  := combinat:-choose( h, h-1 );

#  Compute the polynomial set T.

   for w from 1 to h do
       p[w] := mul( P[k], k=Q[w] );
   end do;
   T := [ seq( expand( p[w] ), w=1..h ) ];

#  Compute of the polynomial P(s).

   Ps := expand( mul( P[i], i=1..h ) );

#  Compute the GCD of the polynomial set T.

   Gs := HEresGCD( T );

#  Compute the LCM of the input polynomial set P.
```

```
   Ls := ERESDivision ( Ps, Gs, quotient );

end proc;

# End of procedure SEresLCM.
```

### A.1.4 The procedure HEresLCM

The procedure `HEresLCM` computes the approximate LCM of a set of several univariate polynomials. The following procedure involves a slightly modified version of the procedure `ERESDivision` which is referred to as `EresDiv2` and computes only the remainder vector of the division of two polynomials using ERES operations. A special matrix is constructed and the built-in procedure `LeastSquares` is properly applied in order to compute an approximate LCM.

**Primary input parameters :**

Pmn : Initial set of polynomials $\mathcal{P}_{m,n}$. (Type: `list(polynom)`)
tol : Numerical accuracy $\varepsilon_t$. (Type: `float`)

**Output parameters :**

lcm : The LCM presented as polynomial. (Type: `polynom`)

**Maple code :**

```
HEresLCM := proc( Pmn::list(polynom(anything)), tol::float )

description"The procedure HEresLCM computes the approximate LCM of
a set of univariate polynomials using ERES and Least-Squares."

local t0,t,m,n,x,i,j,k,p_ind,degs,P,A,R,R1,V,S,dlcm,dmax,
      order,init,q,r,s,lc,lcm1,s_,F0,F1,FN,Cond,N,Ls;
global a;

#  Create initial matrix P and set initial parameters.

   p_ind := indets( Pmn );

   if nops( p_ind ) > 1 then
   error " invalid formal parameters. Only single variable
               polynomials are permitted. "
   else
```

```
        x := op(1, p_ind )                # VARIABLE NAME.
    end if;

    m := nops( Pmn );
    degs := seq( degree( Pmn[i], x ), i=1..m );
    dmax := max( degs );
    dlcm := add( degs[i], i=1..m ); # MAXIMUM DEGREE OF LCM.
    n := dmax + 1;
    lc := [ seq( lcoeff( Pmn[i], x ), i=1..m ) ];
    lcm1 := lcm( seq( lc[i], i=1..m ) );
    init := [seq(ListTools:-Reverse(PolynomialTools[CoefficientList](
              Pmn[i], x )) / lc[i], i = 1..m  ) ];
    P := Matrix( m, n, init, storage=rectangular, datatype=anything,
                 order=Fortran_order, fill=0 );
    P := convert( P, rational);

#   Start main procedure.

    a:='a';
    A := Matrix( dlcm, 1, fill=0);
    q := 0;

    for k from 1 to m do
        R1 := EresDiv2( dlcm, degs[k], P[k,1..degs[k]+1]  );
        for i from 1 to degs[k] do
            A[i+q,1] := R1[ degs[k]+1-i ];
        end do;
        q := q + degs[k];
    end do;

    m := dlcm;
    n := dlcm+1;
    A := subs( [ seq( a[i-1] = x^(i-1), i=1..n ) ], A );
    init := [seq(PolynomialTools[CoefficientList](A[i,1],x), i=1..m )];
    F0 := Matrix( m, n, init, datatype=rational );
    FN := NormRows( m, n, evalf(F0) );
    r  := LinearAlgebra:-Rank( FN );

    for i from 1 to m do
        for j from 1 to n do
```

```
            if abs( FN[i,j] ) <= tol and abs( FN[i,j] ) > 0 then
               FN[i,j] := 0.0;
            end if;
        end do;
    end do;

    F1 := FN[1..m,1..r];
    V  := - Vector[column]( FN[1..m, r+1] );
    s  := LinearAlgebra:-LeastSquares( F1, V );
    s_ := norm( evalf[64]( F1.s-V ), 2);

    S := SingularValues( F1[1..m, 1..r] );
    Cond := evalf(S[1]/S[r]);
    printf("Residual  = %e\n",s_);
    printf("Condition = %e\n",Cond);

# Final output.

    Ls := sort(  add( lcm1*s[i]*x^(i-1), i=1..r ) + lcm1*x^r );

end proc;

#  End of procedure EresLCM.
```

### A.1.5 The procedure `RHEres` in symbolic mode

The procedure `RHEres` evaluates the stability of a real univariate polynomial combining ERES and Routh-Hurwitz methods and corresponds to the algorithm 7.1.

**Primary input parameters :**

f : Initial polynomial. (Type: `polynom`)

**Output parameters :**

A : Sequence of RH-ERES parameters. (Type: `sequence`)

**Maple code :**

```
RHEres := proc( f::polynom(anything), s::name )

description"The procedure RHEres evaluates the stability of a
```

```
univariate polynomial combining ERES and Routh-Hurwitz methods."

local  i, j, k, l, np, n, a, temp, temp, M, polcoef;
global e;

# PROCEDURE'S PARAMETERS AND INITIAL VALUES.

   polcoef := ListTools:-Reverse(
                        PolynomialTools[CoefficientList](f,s));

   np := nops( polcoef );
   n  := np-1;              # Degree of polynomial f(s).
   k  := trunc(n/2)+1;      # Column dimension of initial matrix P.

# CREATE INITIAL MATRIX P.

   P  := Matrix(2,k, storage=rectangular, datatype=anything,
                order=Fortran_order, fill=0);
   for j from 1 to k do
       P[1,j] := polcoef[2*j-1];
       if np >= 2*j then
          P[2,j] := polcoef[2*j];
       end if;
   end do;

# START OF MAIN ITERATIVE PROCESS.

   for i from 1 to n do

   # REORDERING THE ROWS OF MATRIX P.

      temp := P[1,1..k];
      P[1,1..k] := P[2,1..k];
      P[2,1..k] := temp;

   # GAUSSIAN ELIMINATION.

      if P[1,1] = 0 then   # Singular Case 1.
         P[1,1] := e;
      end if;
```

```
      M := - P[2,1] / P[1,1];
      for j from 1 to k do
          P[2,j] := P[2,j] + M * P[1,j];
      end do;
      P := simplify(P);
      a[i] := -M;

      if i < n then
        if add(abs(P[2,j]),j=1..k) = 0 then   # Singular Case 2.
           for j from 1 to k do
               P[2,j] := (n-i-2*(j-1))*P[1,j];
           end do;
        else

   # MATRIX SHIFTING.

        temp := P[2, 2..k];
        P[2, 1..k-1] := temp;
        P[2, k] := 0;

   # DELETE ZERO COLUMNS.

        for j from 1 to k do
            if add(abs(P[l,k-j+1]), l=1..2) = 0 then
               P := LinearAlgebra:-DeleteColumn(P,k-j+1);
               k := k - 1;
               break;
            end if;
        end do;
        end if;
      end if;
   end do;

# END OF MAIN ITERATIVE PROCESS.

   A := [seq( a[i], i=1..n )]

end proc;

# END OF PROCEDURE RHEres.
```

## A.2 The code of the Average Strength algorithm

### The procedure `AVStrength`

The procedure `AVStrength` computes the strength bounds and the average strength of a given approximate GCD and corresponds to the algorithm 5.1.

**Primary input parameters :**

polyL : Initial set of polynomials $\mathcal{P}_{m,n}$. (Type: `list(polynom)`)
GCD : The GCD of the set input set. (Type: `polynom`)

**Output parameters :**

$\underline{\mathcal{S}}$, $\overline{\mathcal{S}}$, $C$, $\mathcal{S}^a$ : Lower, upper strength bounds and condition number and average strength. (Type: `float`)

**Maple code :**

```
strength := proc( polyL::list(polynom(anything)),
                  GCD::polynom(anything), varname::name )

description" The procedure evaluates the average strength of
approximation of an approximate GCD.";

local digits, i, m, n, p, r, x, c, degs, Pn, GCDcoeffs, F,
      F_, SP, SPF, SP1, v, v1, v2, D, dim_D, k, j, N, GCD_;

# INITIAL VALUES.

   digits := Digits;
   if digits < 34 then Digits := 34 fi;

   x := varname;
   m := nops( polyL );      # NUMBER OF POLYNOMIALS.
   Pn := expand(polyL);     # INITIAL SET OF POLYNOMIALS - TYPE ARRAY.

   degs := sort( [seq( degree( Pn[i], x ), i=1..m )] );
   n := degs[m];            # 1st MAXIMUM DEGREE OF POLYNOMIALS.
   p := degs[m-1];          # 2nd MAXIMUM DEGREE OF POLYNOMIALS.
   r := degree( GCD, x);    # DEGREE OF GCD.

   SP := ResMatrix( Pn, x );
```

```
   v  := PolynomialTools[CoefficientList]( GCD, x );
   v  := v/tcoeff(GCD);
   c  := 0;

   for i from 1 to nops(v) do
       if v[i] = 0 then
          c := c+1
       else break;
       end if;
   end do;

   r := r-c;
   GCDcoeffs := [ seq(v[i], i= 1+c..nops(v)) ];
   GCDcoeffs := ListTools[Reverse]( GCDcoeffs );

#  F => (n+p-r)x(n+p) CORRESPONDING TOEPLITZ MATRIX.

   F  := LinearAlgebra:-BandMatrix ( GCDcoeffs, r, n+p );
   F_ := LinearAlgebra:-MatrixInverse( Matrix( F,
                        shape=triangular[lower]),method=subs );
   SPF := LinearAlgebra:-MatrixMatrixMultiply( SP, F_ );
   SP1 := copy(SPF);

#  CONSTRUCT THE APPROPRIATE DEGREE-CHECK-MATRIX.

   v1 := Vector[column]( p, fill=1 );
   v2 := Vector[column]( n, fill=1 );
   D := Matrix( n+p-degs[1], m, shape=rectangular, fill=0 );
   dim_D := n+p-degs[1];     # ROW DIMENSION OF THE MATRIX D.
   D[ 1..p, 1 ] := v1;
   D[ 1..n, 2 ] := v2;

   for i from 3 to m do
       D[ p-degs[m-i+1]+1..n+p-degs[m-i+1], i ] := v2;
   end do;

   k := 0;

   for j from 1 to m do
       for i from 1 to dim_D do
```

```
            if D[i,j] = 1 then
               k := k+1;
               SP1[k, i+r..n+p] := Vector[row]( n+p-i-r+1, fill=0 );
            end if;
        end do;
    end do;

    N[1] := LinearAlgebra:-Norm(SP1,Frobenius);
    N[2] := LinearAlgebra:-Norm(F_,Frobenius);
    N[3] := LinearAlgebra:-ConditionNumber(F, Frobenius);

#   LOWER STRENGTH BOUND.

    N[4] :=  N[1] / N[2];

#   UPPER STRENGTH BOUND.

    SPF := SP1.F;
    N[5] := p* add( SPF[1,j]^2, j=1..n+1 );
    for k from 1 to h do
        N[5] := N[5] + n* add( SPF[1+p+(k-1)*n,j]^2, j=1..p+1 );
    end do;

    printf("Lower Strength =%e, Upper Strength =%e, Condition =%e\n",
           evalf( sqrt(N[4]) ), evalf( sqrt(N[5]) ), evalf( N[3]));

#   FINAL OUTPUT - AVERAGE STRENGTH

    output := evalf( 0.5 * sqrt( N[4] + sqrt(N[5])) );

end proc;

#   END OF PROCEDURE AVStrength.
```

## A.3 The code of the PSVD1 algorithm

### The procedure `psvd1`

The procedure `psvd1` computes the singular values of a numerical matrix using the partial singular value decomposition method and corresponds to the PSVD1 algorithm 4.1.

**Primary input parameters :**

A : Initial matrix. (Type: `Matrix`)
tolS : Initial numerical tolerance $\varepsilon_t$ of the method. (Type: `float`)

**Output parameters :**

(N,sigma,tol,w) : Rank of matrix, singular values, new tolerance, right singular vector. (Type: `sequence(float)`)

**Maple code :**

```
psvd1 := proc( A::Matrix, tol::float )

description"The procedure psvd1 computes the singular values of a
numerical matrix using partial singular value decomposition.";

local BD, V, A_, p, q, j, n, N, tol, maxdiagBD,
      v, sigma, w, c, s, d1, d2, UseBt, output;

   p, q := LinearAlgebra:-Dimensions( A );

#  Bidigonalization.

   if p >= (5/3)*q then
      A_ := LinearAlgebra:-QRDecomposition( evalf(A), output='R' );
      BD := LinearAlgebra:-BidiagonalForm( A_, output='B' );
      n := q;
      UseBt := false;
   elif p < q then
      A_ := evalf(A);
      BD := LinearAlgebra:-BidiagonalForm( A_, output='B' );
      BD := LinearAlgebra:-Transpose( BD );
      n := p;
      UseBt := true;
   else
      A_ := evalf(A);
      BD := LinearAlgebra:-BidiagonalForm( A_, output='B' );
      n := q;
      UseBt := false;
   end if;
```

```
#  Rank-1 detection.

   N, tol := SturmSeqBis( n, BD, tolS );
   w := 0;
   sigma := 0;

   if N = 1 then
      maxdiagBD := max( seq( abs(BD[i,i]), i=1..n ) );
      j := 1;
      while abs(BD[j,j]) <> maxdiagBD  do
            j := j + 1;
      end do;

      if j < n then
         d1 := BD[j,j]^2 + BD[j,j+1]^2;
         d2 := sqrt(d1);
         sigma := d1/d2;
      else
         sigma := BD[n,n];
      end if;

# Back transformation.

      if UseBt = true then
         V := LinearAlgebra:-BidiagonalForm( A_, output='Vt' );
         w := V[j, 1..q];
      else
         V := LinearAlgebra:-BidiagonalForm( A_, output='Vt' );
         if j = n then
            w := V[j, 1..q];
         else
            c := BD[j,j]/d2;
            s := BD[j,j+1]/d2;
            w := < c | s >.V[ j..j+1, 1..q ];
         end if;
      end if;
   end if;

#  Final output.
```

```
   output := (N, sigma, tol, w);

end proc;

# End of procedure psvd1.
```

## A.4 Complementary procedures

### A.4.1 The procedure MakeMatrix

The procedure `MakeMatrix` is used for the formulation of the basis matrix $P_m$ of a set of polynomials $\mathcal{P}_{m,n}$. The produced matrix is structured according to the base vector $e_n(s) = [\,1, s, s^2, \ldots, s^n\,]^t$.

**Maple code :**

```
MakeMatrix := proc( L::list(polynom(anything)), x::name )

description"The procedure MakeMatrix constructs the basis matrix of
a set of univariate polynomials.";

local i, output;

output := Matrix( [seq(PolynomialTools[CoefficientList]( L[i],x ),
           i = 1..nops(L))] )
end proc;

# END OF PROCEDURE MakeMatrix.
```

### A.4.2 The procedure bub

**Maple code :**

```
bub := proc( A::Matrix )

description"The procedure bub finds the best uncorrupted base
of the row space of a matrix A.";

local m, n, r, i, combs, ncombs, Gram, d, dmax, AN, output;

   m, n  := LinearAlgebra:-Dimension( A );
   AN := normrows( m, n, A );
```

```
   r  := LinearAlgebra:-Rank( AN );

   if r <> m then
      Gram   := LinearAlgebra:-MatrixMatrixMultiply( AN,
       LinearAlgebra:-Transpose(AN), outputoptions=[shape=symmetric]);
      combs  := combinat[choose]( m, r );
      ncombs := nops( combs );

      for i from 1 to ncombs do
          d[i] := LinearAlgebra:-Determinant(LinearAlgebra:-SubMatrix(
                  Gram, combs[i], combs[i]) );
      end do;

      dmax := max( seq( d[i], i=1..ncombs ) );

      for i do if dmax = d[i] then break end if end do;

      output := (combs[i],LinearAlgebra:-SubMatrix(A,combs[i],1..n));
   else
      output := ( [1..m], A );
   end if;

end proc;

# END OF PROCEDURE bub.
```

### A.4.3 The procedure ResMatrix

**Maple code :**

```
ResMatrix := proc( PL::list(polynom(anything)), varname::name )

description"The procedure ResMatrix constructs a (n,p)-Extended
Sylvester Matrix (or Resultant Matrix ).";

local i, k, j, m, x, n, p, s, degs, degs_i, degss,
      Pn, SM, D, C, dim_D, v1, v2, LastNonZero, output;

#  INITIAL VALUES.

   x := varname;
```

```
   m := nops( PL );
   Pn  := array( 1..m, expand(PL) );
   degs := array(1..m);
   degs_i := array(1..m);

   for k from 1 to m do
       degs[k] := degree( Pn[k], x );
   end do;

   degss  := inssort( degs );
   degs   := degss[1];
   degs_i := degss[2];
   n := degs[m];
   p := degs[m-1];

#  CONSTRUCT THE SYLVESTER MATRIX.

   SM := Matrix( p+(m-1)*n, n+p, shape=rectangular,
         order=Fortran_order, fill=0 );
   v1 := Vector[column]( p, fill=1 );
   v2 := Vector[column]( n, fill=1 );

#  CONSTRUCT THE APPROPRIATE DEGREE-CHECK-MATRIX.

   D := Matrix( n+p-degs[1], m, shape=rectangular, fill=0 );
   dim_D := n+p-degs[1];
   D[ 1..p, 1 ] := v1;
   D[ 1..n, 2 ] := v2;

   for i from 3 to m do
       D[ p-degs[m-i+1]+1..n+p-degs[m-i+1], i ] := v2;
   end do;

#  EXTRACT THE COEFFICIENTS OF THE POLYNOMIALS.

   for i from 1 to m do
       C[i] := PolynomialTools:-CoefficientList( Pn[i], x );
       C[i] := ListTools:-Reverse( C[i] );
       C[i] := convert( C[i], Vector[row] );
   end do;
```

```
#  CREATE THE EXTENDED SYLVESTER MATRIX USING
#  THE DEGREE-CHECK-MATRIX AS A GUIDE.

   k := 0;

   for j from 1 to m do
       for i from 1 to dim_D do
           if D[i,j] = 1 then
              k := k+1;
              SM[ k, i..i+degs[m-j+1] ] := C[ degs_i[m-j+1] ];
           end if;
       end do;
   end do;

#  FINAL OUTPUT

   output := SM

end proc;

# END OF PROCEDURE ResMatrix.
```

### A.4.4 The procedure normrows

**Maple code :**

```
normrows := proc( p::posint, q::posint,  A::Matrix )

description" The procedure normrows normalizes the rows of the
initial matrix A. ";

local i, r, init, output;

   r := array(1..p);

   for i from 1 to p do
       r[i] := LinearAlgebra:-Normalize (
               LinearAlgebra:-Row ( A, i ), Frobenius );
   end do;
   init := [ seq( convert( r[i], list ), i = 1..p ) ];
```

```
   output := Matrix(p,q,init,storage=rectangular,datatype=anything,
             order=Fortran_order, fill=0);

end proc;

#  END OF PROCEDURE normrows.
```

### A.4.5 The procedure baseexp

**Maple code :**

```
baseexp := proc( u::float )

description" The procedure baseexp computes the exponential
part of a floating point number.";

local eu, k, output;
   eu := evalf( u );
   k  := length( SFloatMantissa( eu ) ) + SFloatExponent( eu );
   output := 10^k
end proc;

#  END OF PROCEDURE baseexp.
```

### A.4.6 The procedure inssort

**Maple code :**

```
inssort := proc( a::array(integer) )

description" The procedure inssort sorts the elements
of an array of integers into ascending order.";

local N, i, j, tmp1, tmp2, ind, v, output;

   N   := nops( op(3, eval(a) ) );
   v   := array( 0..N, [0, seq( a[i], i=1..N )] );
   ind := array( [seq( i, i=1..N )] );

   for i from 2 to N do
       tmp1 := v[i];
       v[0] := tmp1;
```

```
        tmp2 := ind[i];
        j := i-1;
        while tmp1 < v[j] do
              v[j+1] := v[j];
              ind[j+1] := ind[j];
              j := j-1;
        end do;
        v[j+1] := tmp1;
        ind[j+1] := tmp2;
    end do;
    v := array( 1..N, [seq( v[i], i=1..N )] );

    output := ( v, ind )

end proc;

#  END OF PROCEDURE inssort.
```

### A.4.7 The procedure shifting

**Maple code :**

```
shifting := proc( p::posint, q::posint, A::Matrix )

description" The procedure shifting constructs the
shifted form of a matrix.";

local i, k, r, B, output;

   B := A;

   for i from 2 to p do
       k := 1;
       while B[i,k] = 0 do
             k := k+1;
       end do;
       r := B[i, k..q];
       B[i, 1..q] := Vector[row]( q, fill=0 );
       B[i, 1..q-k+1] := r;
   end do;
```

```
   output := B

end proc;

#  END OF PROCEDURE shifting.
```

### A.4.8 The procedure conversion

**Maple code :**

```
conversion := proc( p::posint, q::posint, A::Matrix )

description" The procedure conversion converts the data of a matrix
to fractions with denominator a power of 10.";

local B, i, j, f, b, output;

   B := Matrix( p, q, [], storage=rectangular, datatype=rational,
         order=Fortran_order, fill=0 );

   for i from 1 to p do
       for j from 1 to q do
           f := SFloatMantissa( evalf( A[i,j] ) );
           b := SFloatExponent( evalf( A[i,j] ) );
           B[i,j] := f / 10^(-b);
       end do;
   end do;

   output := B

end proc;

#  END OF PROCEDURE conversion.
```

### A.4.9 The procedure SturmSeqBis

**Maple code :**

```
SturmSeqBis := proc( coldim::posint, B::Matrix, theta::float )

description"The procedure SturmSeqBis computes the Sturm sequence for
‘theta’ and uses a bisection method to find a new bound for ‘theta’."
```

```
local i, n, a, b, c, N1, N2, eps, s1, s2, s3,
       d, norminf_T, dim_c, tol, output;

   if Digits < evalhf(Digits) then
      Digits := 18;
      eps := evalf( 2^(-52) );
   else
      eps := evalf( 2^(-2*Digits) );
      while evalf(1.0 + eps) > 1.0 do
            eps := evalf( eps*0.5 )
      end do;
      eps := evalf( 2.*eps );
   end if;

   n := coldim;
   a := [ seq( B[i,i], i=1..n ) ];
   b := [ seq( B[i,i+1], i=1..n-1 ) ];
   c := ListTools:-Interleave( a, b );
   dim_c := 2*n-1;
   c := Array( 1..dim_c, c );
   N1 := SignAgrees( dim_c, c, theta );
   tol := theta;

   if N1 >= 1 then
      norminf_T := max( abs(c[1]),
            seq(abs(c[i])+abs(c[i+1]), i=1..dim_c-1), abs(c[dim_c]));
      s1 :=   0.;
      s2 :=   norminf_T;
      d  :=   norminf_T;
      while d >= eps do
            s3 := 0.5 * (s1+s2);
            N2 := SignAgrees( dim_c, c, s3 );
            if N2 < 2 then
               s2 := s3
            else
               s1 := s3
            end if;
            tol := s3;
            if s1 > 1e-1 then
               break;
```

```
            end if;
            d := abs( s2 - s1 );
      end do;
   end if;

   output := ( N1, tol )

end proc;

#  END OF PROCEDURE SturmSeqBis.
```

### A.4.10 The procedure SignAgrees

**Maple code :**

```
SignAgrees := proc( dim::posint, b::Array, t::float)

description" The procedure SignAgrees computes the sign agreements
of the terms in the Sturm sequence."

local agrs, s1, s2, s, w, p, i, output;

   if Digits <= evalhf(Digits) then
      Digits := 18;
   end if;

   p[0] :=  1;
   p[1] := -t;
   agrs :=  0;
   s1   := -1;

   for i from 2 to dim+1 do
       p[i] := - t * p[i-1] - b[i-1]^2 * p[i-2];
       w := max( abs(p[i]), abs(p[i-1]) );

       if w > 1e+10 then
          s := 1e+10/w;
          p[i] := s * p[i];
          p[i-1] := s * p[i-1];
       elif w < 1e-10 then
          s := 1e-10/w;
```

```
            p[i] := s * p[i];
            p[i-1] := s * p[i-1];
        end if;

        s2 := sign( p[i] );

        if p[i] = 0. then
            agrs := agrs + 1;
        elif s2 = s1 then
            agrs := agrs + 1;
        else
            s1 := s2;
        end if;

        if agrs = 2 then
            break;
        end if;
    end do;

    output := agrs

end proc;

# End of procedure SignAgrees.
```

### A.4.11 The procedure EresDiv2

**Maple code :**

```
EresDiv2 := proc( m::posint, n::posint, b::Vector )

description"The procedure gives the theoretical result of the
division of two polynomial with degrees m and n where m > n.";

local  i, j, M, P, T, output;
global a;

    P := Matrix( 2,m+1,fill=0 );

    for j from 1 to n+1 do
        P[1,j] := b[j];
```

```
   end do;

   for j from 1 to m+1 do
       P[2,j] := a[m-j+1];
   end do;

   for i from 1 to m-n+1  do

#  GAUSSIAN ELIMINATION.

       M := - P[2,1] / P[1,1];
       P[2,1] := 0;
       for j from 2 to n+1 do
           P[2,j] := P[2,j] + M * P[1,j];
       end do;

#  MATRIX SHIFTING.

       T := P[2, 2..m+1];
       P[2, 1..m] := T;
       P[2, m+1] := 0;

   end do;

   output := LinearAlgebra:-Transpose( P[2,1..n] );

end proc;

# END OF PROCEDURE EresDiv2.
```

### A.4.12 The procedure Make2dMatrix

**Maple code :**

```
Make2dMatrix:=proc(L::list(polynom(anything)),var1::name,var2::name)

description"The procedure Make2dMatrix constructs the basis matrix
of a set of polynomials in two variables.";

local i, j,m, k, d, P, r, t, V1, V2 , VV, XY, d1, d2, n2, output;
```

```
    m  := nops(L);
    d1 := max( seq( degree( L[k], var1 ), k=1..m ) ) + 1 ;
    d2 := max( seq( degree( L[k], var2 ), k=1..m ) ) + 1 ;
    n2 := d1*d2;
    P  := Matrix( m, n2, fill=0 );

    for k from 1 to m do
        r := Vector[row]( d1, PolynomialTools:-CoefficientList(
             L[k], var1 ) );
        for i from 1 to d1 do
            t[i] :=Vector[row](d2, PolynomialTools:-CoefficientList(
                    r[i], var2 ) );
        end do;
        P[k,1..n2] := Vector[row]( [ seq( t[i], i=1..d1) ] );
    end do;

    V1 := Vector[column]( [seq( var1^i, i=0..d1-1 )] );
    V2 := Vector[column]( [seq( var2^i, i=0..d2-1 )] );
    VV := Matrix( n2, d1, fill=0 );

    for j from 1 to d1 do
        VV[ (j-1)*d2+1..j*d2, j ] := V2;
    end do;

    XY := VV.V1;     # BASE VECTOR.

    output := ( P, XY )

end proc;

# END OF PROCEDURE Make2dMatrix.
```

# References and Bibliography


[1] Antsaklis, P. J., and Michel, A. N. *Linear Systems.* Birkhäuser, Boston, 2006.

[2] Barlow, J. L. More accurate bidiagonal reduction for computing the singular value decomposition. *SIAM J. Matrix Anal. Appl. 23*, 3 (2002), 761–798.

[3] Barnett, S. Greatest common divisor of several polynomials. In *Proc. Cambridge Philos. Soc.* (1971), pp. 263–268.

[4] Barnett, S. *Matrices in Control Theory.* Van Nostrand Reinhold Company, 1971.

[5] Barnett, S. Greatest common divisor from generalized Sylvester matrices. In *Proc. Cambridge Philos. Soc.* (1980), vol. 8, pp. 271–279.

[6] Beelen, T., and Van Dooren, P. A pencil approach for embedding a polynomial matrix into a unimodular matrix. *SIAM J. Matrix Anal. Appl. 9* (1988), 77–89.

[7] Benidir, M., and Picinbono, B. Extended table for eliminating the singularities in Routh's array. *IEEE Trans. on Aut. Control 35* (1990), 218–222.

[8] Bini, D. A., and Boito, P. Structured matrix-based methods for polynomial $\epsilon$-gcd:Analysis and comparison. In *Proc. ISSAC'07* (Waterloo, Ontario, Canada, 2007), pp. 9–16.

[9] Björck, Å. Component-wise perturbation analysis and error bounds for linear least squares solutions. *BIT 31* (1991), 238–244.

[10] Björck, Å. *Numerical Methods for Least Squares Problems.* SIAM, 1996.

[11] Blankiship, W. A new version of Euclid's algorithm. *American Mathematics Monthly 70* (1963), 742–744.

[12] Brown, W. S. On Euclid's algorithm and the computation of polynomials greatest common divisors. *Journal of the Association for Computer Machinery 18*, 4 (1971), 478–504.

[13] Chan, T. F. An improved algorithm for computing the singular value decomposition. *ACM Trans. Math. Soft. 8* (1982), 72–83.

[14] Chapellat, H., Mansour, M., and Bhattacharyya, S. P. Elementary proofs of some classical stability criteria. *IEEE Trans. Education 33*, 3 (1990), 232–239.

[15] Chen, C. T. *Linear Systems Theory and Design*. Holt-Rinehart and Winston, New York, 1984.

[16] Chin, P., Corless, R. M., and Corliss, G. F. Optimization strategies for the approximate gcd problem. In *Proc. ISSAC'98* (Rostock, Germany, 1998), pp. 228–235.

[17] Corless, R. M., Gianni, P. M., Trager, B. M., and Watt, S. M. The singular value decomposition for polynomial systems. In *Proc. ISSAC'95* (Quebec, Canada, 1995), pp. 195–207.

[18] Datta, B. N. *Numerical Linear Algebra and Applications*. Brooks/Cole Publishing Company - ITP, 1995.

[19] Diaz-Toca, G. M., and Gonzalez-Vega, L. Computing greatest common divisors and squarefree decompositions through matrix methods: The parametric and approximate cases. *Linear Algebra and its Applications 412* (2006), 222–246.

[20] Emiris, I. Z., Galligo, A., and Lombardi, H. Certified approximate univariate GCDs. *Journ. Pure and Applied Algebra 117 & 118* (1997), 229–251.

[21] Fatouros, S. *Approximate Algebraic Computations in Control Theory*. PhD thesis, Control Engineering Centre, School of Engineering and Mathematical Sciences, City University London, U.K., 2003.

[22] Fatouros, S., and Karcanias, N. Resultant properties of the GCD of many polynomials and a factorisation representation of GCD. *Int. Journ. Control 76*, 16 (2003), 1666–1683.

[23] Foster, L. Rank and null space calculations using matrix decomposition without column interchanges. *Linear Algebra and its Applications 74* (1986), 47–71.

[24] Gantmacher, F. R. *Applications of the Theory of Matrices*. Dover Publications, New York, 1959.

[25] Golub, G. H. Numerical methods for solving least-squares problems. *Numerische Mathematik 7* (1965), 206–216.

[26] Golub, G. H., and Kahan, W. Calculating the singular values and pseudo-inverse of a matrix. *SIAM Journal on Numerical Analysis 2* (1965), 205–224.

[27] Golub, G. H., and Van Loan, C. F. *Matrix Computations*, 2nd ed. The John Hopkins University Press, Baltimore, London, 1989.

[28] Golub, G. H., and Wilkinson, J. H. Note on the iterative refinement of least squares solutions. *Numerische Mathematik 9*, 2 (1966), 139–168.

[29] Grcar, J. Optimal sensitivity analysis of linear least squares. Technical Report LBNL-52434, Lawrence Berkeley National Laboratory, Berkeley U.S.A., 2003.

[30] Grcar, J., Saunders, M. A., and Su, Z. Estimates of optimal backward perturbations for linear least squares problems. Technical Report LBNL-62909, Lawrence Berkeley National Laboratory, Berkeley U.S.A., 2008.

[31] Gu, M. Backward perturbation bounds for linear least squares problems. *SIAM Journal on Matrix Analysis and Applications 20*, 2 (1999), 363–372.

[32] Heath, T. L. *The Thirteen Books of Euclid's Elements*, 2nd ed. Dover Publications, 1956.

[33] Hirsch, M., and Smale, S. *Differential Equations, Dynamical Systems and Linear Algebra*. Academic Press, New York, 1974.

[34] Ho, M. T., Datta, A., and Bhattacharyya, S. P. An elementary derivation of the Routh-Hurwitz criterion. *IEEE Trans. Automatic Control 43*, 3 (1998), 405–409.

[35] Hoffman, K., and Kunze, R. *Linear Algebra*, 2nd ed. Prentice Hall, 1971.

[36] Holtz, O. Hermite-Biehler, Routh-Hurwitz, and total positivity. *Linear Algebra and its Applications 372* (2003), 105–110.

[37] Hurwitz, A. On the conditions under which an equation has only roots with negative real parts. *Mathematische Annalen 46* (1895), 273–284.

[38] Kailath, T. *Linear Systems*. Prentice Hall, Inc, Englewood Cliffs, New Jersey, 1980.

[39] Karcanias, N. Matrix pencil approach to geometric system theory. In *Proc. IEE* (1979), vol. 126, pp. 585–590.

[40] Karcanias, N. Invariance properties and characterisation of the greatest common divisor of a set of polynomials. *Int. J. Control 46* (1987), 1751–1760.

[41] Karcanias, N. Multivariable poles and zeros. In *Control Systems, Robotics and Automation*. 2002. From Encyclopedia of Life Support Systems (EOLSS). Developed under the Auspices of the UNESCO, EOLSS Publishers, Oxford, UK, [http://www.eols.net][Retrieved October 26, 2005].

[42] Karcanias, N., Fatouros, S., Mitrouli, M., and Halikias, G. Approximate greatest common divisor of many polynomials, generalised resultants and strength of approximation. *Computers & Mathematics with Applications 51*, 12 (2006), 1817–1830.

[43] Karcanias, N., Giannakopoulos, C., and Hubbard, M. Almost zeros of a set of polynomials of R[s]. *Int. J. Control 38* (1983), 1213–1238.

[44] Karcanias, N., and Kouvaritakis, B. The output zeroing problem and its relationship to the invariant zero structure: a matrix pencil approach. *Int. J. Control 30* (1979), 395–415.

[45] Karcanias, N., and Mitrouli, M. A matrix pencil based numerical method for the computation of the gcd of polynomials. *IEEE Trans. Autom. Cont. 39* (1994), 977–981.

[46] Karcanias, N., and Mitrouli, M. Approximate algebraic computations of algebraic invariants. In *Symbolic methods in control systems analysis and design*, vol. 56 of *IEE Control Engin. Series*. 1999, pp. 135–168.

[47] Karcanias, N., and Mitrouli, M. Numerical computation of the least common multiple of a set of polynomials. *Reliable computing 6*, 4 (2000), 439–457.

[48] Karcanias, N., and Mitrouli, M. System theoretic based characterisation and computation of LCM of a set of polynomials. *Linear Algebra and its Applications 381* (2004), 1–23.

[49] Karcanias, N., Mitrouli, M., and Triantafyllou, D. Matrix Pencil methodologies for computing the greatest common divisor of polynomials: Hybrid algorithms and their performance. *Int. Journ. of Control 79*, 11 (2006), 1447–1461.

[50] Karcanias, N., and Vafiadis, D. Canonical forms for state space descriptions. In *Control Systems, Robotics and Automation*. 2002. From Encyclopedia of Life Support Systems (EOLSS). Developed under the Auspices of the UNESCO, EOLSS Publishers, Oxford, UK, [http://www.eols.net][Retrieved October 26, 2005].

[51] Karlson, R., and Waldén, B. Estimation of optimal backward perturbation bounds for the linear least squares problem. *BIT Numerical Mathematics 37*, 4 (1997), 862–869.

[52] Karmarkar, N., and Lakshman, Y. N. Approximate polynomial greatest common divisors and nearest singular polynomials. In *Proc. ISSAC'96* (Zurich, Switzerland, 1996), pp. 35–39.

[53] Karmarkar, N., and Lakshman, Y. N. On approximate GCDs of univariate polynomials. *J. Symbolic Computation 26*, 6 (1998), 653–666.

[54] Lawson, C. L., and Hanson, R. J. *Solving Least-Squares Problems*. Prentice Hall, Englewood Cliffs, NJ., 1974.

[55] MacFarlane, A., and Karcanias, N. Poles and zeros of linear multivariable systems: a survey of the algebraic, geometric and complex variable theory. *Int. J. Control 24* (1976), 33–74.

[56] Marcus, M., and Minc, M. *A survey of matrix theory and matrix inequalities*. Allyn and Bacon, Boston, 1964.

[57] Mitrouli, M., and Karcanias, N. Computation of the GCD of polynomials using Gaussian transformation and shifting. *Int. Journ. Control 58* (1993), 211–228.

[58] Mitrouli, M., Karcanias, N., and Koukouvinos, C. Further numerical aspects of the ERES algorithm for the computation of the greatest common divisor of polynomials and comparison with other existing methodologies. *Utilitas Mathematica 50* (1996), 65–84.

[59] Mitrouli, M., Karcanias, N., and Koukouvinos, C. Numerical performance of the matrix pencil algorithm computing the greatest common divisor of polynomials and comparison with other matrix-based methodologies. *J. Comp. Appl. Math. 76* (1996), 89–112.

[60] Mitrouli, M., Karcanias, N., and Koukouvinos, C. Numerical aspects for nongeneric computations in control problems and related applications. *Congressus Numerantium 126* (1997), 5–19.

[61] Noda, M. T., and Sasaki, T. Approximate GCD and its applications to ill-conditioned algebraic equations. *Jour. of Comp. and Appl. Math. 38* (1991), 335–351.

[62] Pace, I. S., and Barnett, S. Comparison of algorithms for calculation of g.c.d of polynomials. *Int. Journ. Control 4*, 2 (1973), 211–216.

[63] Pan, V. Y. Computation of approximate polynomial GCDs and an extension. *Information and Computation 167* (2001), 71–85.

[64] Qui, W., Hua, Y., and Abed-Meraim, K. A subspace method for the computation of the gcd of polynomials. *Automatica 33*, 4 (1997), 255–284.

[65] Rosenbrock, H. *State Space and Multivariable Theory.* Nelson, London, 1970.

[66] Routh, E. J. *A treatise on the stability of a given state of motion, particularly steady motion.* Macmillan and co, London, 1877.

[67] Rupprecht, D. An algorithm for computing certified approximate GCD of $n$ univariate polynomials. *Journ. of Pure and Applied Algebra 139* (1999), 255–284.

[68] Sasaki, T., and Noda, M. Approximate square-free decomposition and root-finding of ill-conditioned algebraic equations. *J. Inf. Process. 12* (1989), 159–168.

[69] Sasaki, T., and Sasaki, M. Polynomial remainder sequence and approximate GCD. *ACM SIGSAM Bull. 31* (1997), 4–10.

[70] Schoenhage, A. Quasi-GCD computations. *J. Complexity 1* (1985), 118–137.

[71] Sun, J. G. On optimal backward perturbation bounds for the linear least squares problem. *BIT Numerical Mathematics 37*, 1 (1997), 179–188.

[72] Triantafyllou, D. *Algorithms for Computing the Rank and Nullspace of Sylvester, Toeplitz Matrices and Applications.* PhD thesis, Department of Mathematics, University of Athens, Greece, 2008. (In Greek).

[73] Triantafyllou, D., and Mitrouli, M. Two resultant based methods computing the greatest common divisor of two polynomials. *Lecture Notes in Computer Science 3401* (2005), 519–526.

[74] Van Der Sluis, A. Stability of the solutions of linear least squares problems. *Numerische Mathematik 23* (1975), 241–254.

[75] Van Huffel, S. Partial singular value decomposition algorithm. *Journ. of Comp. and Appl. Math. 33* (1990), 105–112.

[76] Van Huffel, S., and Vandewalle, J. An efficient and reliable algorithm for computing the singular subspace of a matrix, associated with its smallest singular values. *Journ. of Comp. and Appl. Math. 19* (1987), 313–330.

[77] Vardulakis, A. I. G., and Stoyle, P. N. R. Generalized resultant theorem. *IMA Journal of Applied Mathematics 22*, 3 (1978), 331–335.

[78] Waldén, B., Karlson, R., and Sun, J. G. Optimal backward perturbation bounds for the linear least squares problem. *Numerical Linear Algebra with Applications 2*, 3 (1995), 271–286.

[79] Wall, H. S. Polynomials whose zeros have negative real parts. *Amer. Math. Monthly 52* (1945), 308–322.

[80] Wedin, P. A. Perturbation theory for pseudo-inverses. *BIT 13* (1973), 217–232.

[81] Wilkinson, J. H. *Rounding Errors in Algebraic Processes.* Her Majesty's Stationary Office, London, 1963.

[82] Wonham, W. M. *Linear Multivariable Control: A Geometric Approach*, 2nd ed. Springer Verlag, New York, 1984.

[83] Zarowski, C. J., Ma, X., and Fairman, F. W. QR-factorization method for computing the greatest common divisor of polynomials with inexact coefficients. *IEEE Transactions on Signal Processing 48*, 11 (2000), 3042–3051.

[84] Zeng, Z. The approximate gcd of inexact polynomials, Part I: A univariate algorithm. [http://www.neiu.edu/ zzeng/uvgcd.pdf], 2004.

[85] Zeng, Z. A method computing multiple roots of inexact polynomials. *Mathematics of Computation 74* (2005), 869–903.

[86] Zhi, L., and Yang, Z. Computing approximate gcd of univariate polynomials by structure total least norm. MM Research Preprints, 375–387, 24, MMRC, AMSS, Academia Sinica, December 2004.

# Publications

[87] Christou, D., Karcanias, N., Mitrouli, M, and Triantafyllou, D. Numerical and symbolical methods for the GCD of several polynomials. In *Numerical Linear Algebra in Signals, Systems and Control*, vol. 80 of *Lecture Notes in Electrical Engineering*. 2011, pp. 123–144.

[88] Christou, D., Karcanias, N., and Mitrouli, M. The ERES method for computing the approximate GCD of several polynomials. *Applied Numerical Mathematics 60* (2010), 94–114.

[89] Christou, D., Karcanias, N., and Mitrouli, M. The Euclidean division as an iterative ERES-based process. In *NumAn 2008 Book of Proceedings* (Kalamata, Greece, 2008), pp. 68–71.

[90] Christou, D., Karcanias, N., and Mitrouli, M. A symbolic-numeric software package for the computation of the GCD of several polynomials. In *NumAn 2007 Book of Proceedings* (Kalamata, Greece, 2007), pp. 54–57.

[91] Christou, D., Karcanias, N., Mitrouli, M., and Triantafyllou, D. Numerical and symbolical comparison of resultant type, ERES and MP methods. In *Proc. ECC'07* (Kos, Greece, 2007), TuA 13.5, pp. 508–511.