KU LEUVEN
Faculty of Engineering Science
Department of Electrical Engineering
STADIUS
Kasteelpark Arenberg 10, B-3001 Leuven, Belgium

A Numerical Linear Algebra Framework for Solving Problems with Multivariate Polynomials

Kim Batselier

Jury:
Prof. dr. ir. Hendrik Van Brussel, chairman
Prof. dr. ir. B. De Moor, promotor
Prof. dr. ir. J.A.K. Suykens
Prof. dr. ir. J. Vandewalle
Prof. dr. ir. K. Meerbergen
Prof. dr. ir. J. Schoukens
(Vrije Universiteit Brussel)
Prof. dr. B. Hanzon
(University College Cork)

Dissertation presented
in partial fulfillment of the
requirements for the degree
of Doctor in Engineering

September 2013

D/2013/7515/89
ISBN 978-94-6018-703-2

Voorwoord

Dit doctoraatswerk zou nooit tot stand kunnen gekomen zijn zonder de bijdrage van vele mensen.

Vooreerst wil ik mijn promotor Prof. Bart De Moor bedanken. Niet alleen heeft hij mij de kans gegeven om een doctoraat te starten maar daarnaast heeft hij mij ook ingeleid in de wondere wereld van veeltermen en lineaire algebra. Ik zal niet snel onze meetings vergeten waarin we het niet alleen hadden over oplossingen op oneindig, de Hilbert functie en lineaire algebra maar ook over algemene relativiteit en kwantummechanica.

Ik zou ook graag alle leden van mijn begeleidingscommissie en examencommissie willen bedanken. Prof. Yves Moreau, voor de interesse in mijn onderzoek en de feedback. Prof. Johan Suykens, voor de goede discussies tijdens de Alma lunches, de interesse in mijn onderzoek en onmisbare feedback vooral tijdens de eerste jaren van het doctoraat. Prof. Joos Vandewalle, voor de raad en steun en het vertrouwen dat hij me heeft gegeven voor het uitwerken van de Junior College Wiskunde module. Prof. Karl Meerbergen, voor de hulp met numerieke aspecten en mij in te wijden tot de wereld van de Krylov subspace methodes. Prof. Johan Schoukens en Prof. Bernard Hanzon voor het nalezen van de tekst en de kritische vragen en opmerkingen en Prof. Hendrik Van Brussel om als voorzitter van mijn jury op te treden.

Ik had het geluk een compagnon te hebben tijdens het doctoraat. Iemand waarmee ik samen kon uitvissen hoe het nu zat met die Macaulay matrix en multivariate veeltermen. Philippe, we hebben samen een lange weg afgelegd en het doctoraat zou nooit zijn geweest wat het nu is zonder jouw vriendschap en steun. Je hebt bovendien niet alleen op professioneel vlak een lange weg afgelegd. Ik heb je ook zien evolueren van vrijgezel tot echtgenoot en van echtgenoot tot vader. Ik wens dan ook jou, Gülin en Maren Mina het allerbeste toe!

I would also like to thank all colleagues and friends for the many enjoyable hours we have spent together: Dries, Maarten, Pieter, Kris, Nico, Raf, Niels, Tillmann, Marco, Carlos, Rocco, Mauricio, Siamak, Fabian, Xiaolin, Vilen, Raghvendra, Lei,

Tom, Toni, Marko, Maarten Driesen, Jef, Eisuke, Kristien, Ozlëm, Claudia and the many others.

Graag zou ik ook Ida Tassens, John Vos en Maarten en Liesbeth willen bedanken voor de administratieve en technische ondersteuning tijdens het doctoraat.

Ten slotte zou ik ook graag mijn ouders, grootouders, broer en alle andere familieleden willen bedanken voor de vele steun. Jiang Lai, thank you for all your encouragement and support! Let us now continue to work for our happy future together.

Kim Batselier,
Leuven 2013

Abstract

Multivariate polynomials are ubiquitous in engineering. Not only are they a natural modelling tool, many parameter estimation problems can also be written as finding the roots of a multivariate polynomial system. Most computational methods in this setting are symbolical. In fact, a whole domain of research called computer algebra sprung into being to develop and study symbolical algorithms for problems with multivariate polynomials. This has somehow prevented the growth of a numerical approach to these problems. This thesis bridges this gap to numerical methods by developing a numerical linear algebra framework that allows us to solve problems with multivariate polynomials. The two main tools in this polynomial numerical linear algebra (PLNA) framework are the singular value decomposition (SVD) and rank-revealing QR decomposition.

First, we consider three basic operations on multivariate polynomials: addition, multiplication and division, and describe their corresponding operations in the PNLA framework: vector addition, vector matrix multiplication and vector decomposition along subspaces. Next, we introduce the Macaulay matrix and provide interpretations for three of its fundamental subspaces: its row space, its null space and its left null space. Orthogonal bases for these subspaces are crucial for the numerical backward stability of the algorithms and hence a recursive orthogonalization scheme is developed. This scheme exploits both the sparsity and structure of the Macaulay matrix. We then introduce the canonical decomposition of the vector space of multivariate polynomials. This concept will be crucial as it provides a stop criterion for many of the algorithms that are developed. Finally, we discuss 7 different applications with corresponding algorithms. These are finding a Gröbner basis, computing all affine roots of a polynomial system, solving the ideal membership problem, doing the syzygy analysis, multivariate elimination, finding approximate least common multiples and greatest common divisors and removing multiplicities of affine roots.

List of symbols

$\mathcal{C}^n$	polynomial ring of polynomials in n variables over $\mathbb{C}$
$\mathcal{C}_d^n$	vector space of n-variate polynomials up to degree d
$\mathcal{P}^n$	polynomial ring of $n+1$-variate homogeneous polynomials
$\mathcal{P}_d^n$	vector space of $n+1$-variate homogeneous polynomials of degree d
$f_1, \ldots, f_s$	polynomial system of s polynomials in n variables
$f_1^h, \ldots, f_s^h$	polynomial system of s homogeneous polynomials in $n+1$ variables
D	Divisor matrix for n-variate polynomials $p, f_1, \ldots, f_s$
$\mathcal{D}$	row space of the divisor matrix D
$M(d)$	Macaulay matrix of degree d for a polynomial system $f_1, \ldots, f_s$
$\mathcal{M}_d$	row space of the Macaulay matrix $M(d)$
$\mathcal{M}_d'$	dual vector space of $\mathcal{M}_d$
$\mathcal{M}_d^o$	annihilator of $\mathcal{M}_d$
$p(d)$	number of rows of $M(d)$
$q(d)$	number of columns of $M(d)$
$r(d)$	rank of $M(d)$
$c(d)$	nullity of $M(d)$
$l(d)$	nullity of $M(d)^T$
$N(d)$	orthogonal basis null space of $M(d)$
$U(d)$	orthogonal basis row space of $M(d)$
K	generalized Vandermonde matrix
$\partial_j\|_z$	differential functional evaluated in a point z
$A^\dagger$	Moore-Penrose pseudoinverse of matrix A

$\mathcal{A}_d$	vector space spanned by linearly independent leading monomials in $\mathcal{M}_d$
$\mathcal{B}_d$	vector space spanned by linearly dependent leading monomials in $\mathcal{M}_d$
$\langle f_1, \ldots, f_s \rangle$	polynomial ideal generated by $f_1, \ldots, f_s$
$\sqrt{I}$	radical polynomial ideal of $\langle f_1, \ldots, f_s \rangle$
$S(f_1, f_2)$	S-polynomial of multivariate polynomials f_1, f_2
p_{red}	square-free part of a univariate polynomial p
LCM	least common multiple
GCD	greatest common divisor

Contents

Contents vii

1 Introduction 1

1.1 Motivating Problems . . . 3

1.1.1 Prediction Error Method System Identification . . . 3

1.1.2 Maximum Likelihood Estimation of Mixtures of Multinomial Distributions . . . 7

1.1.3 Blind Image Deconvolution . . . 10

1.2 Contributions . . . 14

1.2.1 Polynomial Numerical Linear Algebra Framework . . . 14

1.2.2 Bounds for the singular values and condition number of the multiplication matrix and Macaulay matrix . . . 14

1.2.3 Recursive orthogonalization algorithm for the Macaulay matrix . . . 15

1.2.4 Interpretations and the introduction of the canonical decomposition . . . 15

1.3 Chapter overview of the thesis . . . 15

2 Basic Operations on Polynomials 21

2.1 Multivariate polynomials as vectors . . . 21

2.2 Multivariate polynomial Addition . . . 22

2.3 Multivariate Polynomial Multiplication . . . 23

2.3.1 Multiplication matrix 23
2.3.2 Condition number 25
2.4 Multivariate Polynomial Division 30
2.4.1 Divisor matrix 31
2.4.2 Quotient Space 32
2.4.3 Nonuniqueness of quotient 34
2.4.4 Nonuniqueness of remainder 34
2.4.5 The Geometry of Polynomial Division 38
2.4.6 Algorithm & Numerical Implementation 39
2.4.7 Numerical Experiments 42
2.4.8 Division by a Gröbner Basis 43
2.4.9 Buchberger's Algorithm 44

3 Macaulay matrix 47

3.1 Definition 47
3.2 Size, number of nonzero coefficients and density 49
3.3 Upper bound on largest singular value 51
3.4 Row space 56
3.4.1 Affine interpretation 57
3.4.2 Projective interpretation 58
3.5 Left null space 60
3.5.1 Syzygy analysis 60
3.5.2 Degree of regularity 65
3.6 Null space 66
3.6.1 Link with projective roots 67
3.6.2 Dual vector space 68
3.6.3 Removing multiplicities of affine roots 73
3.6.4 Conditions for existence of particular roots 74

4 Fast Recursive Orthogonalization of the Macaulay matrix 77

4.1 Introduction 77

4.2 Notation 79

4.3 The Orthogonalization Scheme 80

4.4 Computational Complexity 83

4.4.1 SVD 83

4.4.2 Rank revealing QR decomposition 85

4.5 Choosing the numerical tolerance τ 85

4.6 Numerical Experiments 87

4.6.1 Example 1 87

4.6.2 Example 2 88

4.6.3 Example 3 91

4.6.4 Example 4 92

4.6.5 Example 5 93

5 The Canonical Decomposition of $\mathcal{C}_d^n$ 95

5.1 The canonical decomposition of $\mathcal{C}_d^n$ 95

5.1.1 Definition 95

5.1.2 Importance of the canonical decomposition 98

5.1.3 Numerical Computation of the Canonical Decomposition . 99

5.1.4 Numerical Experiment - no perturbations on the coefficients 100

5.1.5 Numerical Experiment - perturbed coefficients 101

5.2 The Reduced Canonical Decomposition of $\mathcal{C}_d^n$ 102

5.2.1 The Reduced Monomials $A^\star(d), B^\star(d)$ and Polynomials $G(d)$ 102

5.2.2 Numerical Computation of $A^\star(d), B^\star(d)$ and $G(d)$ 106

5.2.3 Numerical Experiments 107

5.3 Border Bases 108

6 Applications **113**

6.1 Gröbner basis 113

6.2 Affine root-finding 117

6.3 Ideal Membership problem 125

6.4 Iterative algorithm for finding $l(d)$ and $d^\star$ 126

6.5 Multivariate Polynomial Elimination 129

6.5.1 The Geometry of Polynomial Elimination 130

6.5.2 Algorithm & Numerical Implementation 130

6.5.3 Numerical Experiments 132

6.6 Approximate LCM and GCD 133

6.6.1 Computing the LCM 134

6.6.2 Computing the GCD 136

6.6.3 Choosing the numerical tolerance τ 137

6.6.4 Numerical Experiments 138

6.7 Removing Multiplicities 142

7 Conclusions and Future Work **145**

7.1 Concluding Remarks 145

7.2 Future Research 147

7.2.1 Numerical Analysis 147

7.2.2 Curse of Dimensionality 148

A Numerical Linear Algebra **149**

A.1 Matrix Notation 149

A.2 Vector and Matrix Norms 150

A.3 Four Fundamental Subspaces 151

A.4 Dual vector space 151

A.5 Moore-Penrose pseudoinverse 152

A.6 Condition Number for matrix inversion and least squares 152
A.7 QR Decomposition 154
A.8 Singular Value Decomposition 155
A.9 Principal Angles 156
A.10 Intersection of Subspaces 157
A.11 CS Decomposition 158
A.12 Projectors 159
A.13 Sparse Matrices 160

B Algebraic Geometry **163**
B.1 Polynomials & Monomial Ordering 163
B.2 Homogeneous Polynomials and Coordinates 165
B.3 Ideals 166
B.4 Varieties 166
B.5 Gröbner Basis 167
B.6 Least Common Multiples and Greatest Common Divisors 169

References **171**

Chapter 1

Introduction

Many engineering problems require some kind of mathematical modelling step and multivariate polynomials are a natural modelling tool. This results in problems in which one needs to solve multivariate systems of polynomial equations, divide multivariate polynomials, eliminate variables, compute greatest common divisors, etc. The area of mathematics in which multivariate polynomials are studied is algebraic geometry and has a rich history spanning many centuries [31]. Algebraic geometry started with the task of finding all roots of a univariate polynomial and this has led to some important breakthroughs in mathematics, e.g. the introduction of complex numbers and the development of group theory. By the late 1800s and early 1900s, many of the algebraic concepts that are used in this thesis such as rings, polynomial ideals and the Hilbert Polynomial were known and studied by Kronecker, Noether, Hilbert, Lasker, Macaulay and Sylvester [45, 60, 61, 85]. A milestone was the Ph.D. thesis of Bruno Buchberger [16], who in 1965 introduced the notion of a Gröbner basis. Hironaka independently discovered this concept in 1964 and named it a standard basis [46]. Buchberger's algorithm to compute a Gröbner basis, together with the exponential increase of computing power led to the development of computer algebra. The main focus in this field of research is the design and implementation of symbolical algorithms for the solution of problems in pure mathematics. Two important contributions in this respect were a criterion for detecting unnecessary reductions in Buchberger's algorithm [17] and the insight that Buchberger's algorithm could be interpreted as performing Gaussian elimination on a large sparse matrix [56]. These two important contributions would eventually result in Faugère's F4 and F5 algorithms [37, 38] that symbolically compute Gröbner bases using multiple precision integers. These algorithms are at the time of this writing the state of the art to compute Gröbner bases. With this rapid development of computer algebra, the growth of a numerical nonlinear (or polynomial) algebra was somehow prevented. The

most important numerical method for solving multivariate polynomial systems is numerical polynomial homotopy continuation (NPHC), developed in the 1990s [64,76,75,87]. Although NPHC is often referred to as numerical algebraic geometry, it is not possible to eliminate variables, compute approximate GCDs or perform multivariate divisions nor is there a strong focus on numerical analysis. Indeed, its main application is finding all the affine roots of a system of multivariate polynomial equations. The true 'birth' of the numerical analysis of polynomial algebra is the 1988 paper by Auzinger and Stetter [1]. They presented a direct algorithm for the numerical computation, in floating point arithmetic, of all isolated zeros of a multivariate polynomial system. Its two main ingredients are Gaussian elimination and eigenvector computations. The case of multiple zeros, where the eigenvalue problem is generalized to a kind of Jordan decomposition, is described in [63]. Twenty years of research by Stetter and his collaborators into the numerical analysis of polynomial algebra led to the seminal work 'Numerical Polynomial Algebra' [78]. And although Stetter uses a lot of linear algebra notation, he did not present an extensive linear algebra framework that allows to solve problems with multivariate polynomials. The only exception being the problem of solving a multivariate polynomial system. This brings us to the goal of this thesis:

The goal of this thesis is to establish a numerical linear algebra framework in which problems with multivariate polynomials can be expressed and solved.

The emphasis is hereby on numerical linear algebra, viz. operations on vectors and matrices, and on solving problems: elimination, division, computing approximate LCMs/GCDs, affine root-finding, computing a Gröbner basis, etc.... We will refer to this proposed framework as the Polynomial Numerical Linear Algebra (PNLA) framework. To our knowledge, only one other person, Zhonggang Zeng, is also using only numerical linear algebra to solve different problems such as elimination [92], approximate GCDs [90,94], and finding the multiplicity structure of a multiple root [28, 93]. Where applicable, we will compare his methods with the algorithms that we present in this thesis.

With polynomial models appearing in nearly all areas of scientific computing and coefficients coming in most cases from experimental data, it is justified to make the following assumption.

Assumption 1.1 *We assume throughout the thesis that all coefficients of all multivariate polynomials are real numbers.*

All of the algorithms in this thesis were numerically implemented in MATLAB/OCTAVE and are freely available on request. Numerical experiments were run on a 2.66 GHz quad-core desktop computer with 8 GB RAM.

1.1 Motivating Problems

In order to motivate the need for the PNLA framework, we present 3 different real-life problems and show how they can be stated in terms of multivariate polynomials. More examples of polynomial models are described in [42] with applications in Physics, Chemistry, Engineering, and Computer Science. The first problem we will discuss is that of parametric system identification.

1.1.1 Prediction Error Method System Identification

In this section it is shown that prediction error methods (PEMs) [59] for finding the parameters of Single-Input Single-Output (SISO) Linear Time Invariant (LTI) models is equivalent to finding the roots of a multivariate polynomial system.

PEMs are polynomial system solving

Let the input and output of a system at time t be denoted by $u(t)$ and $y(t)$ respectively and $e(t)$ be a white noise sequence. We will assume that these signals are sampled at discrete time steps. The collected past data up to time N are then denoted by

$$Z^N = \{u(1), y(1), \ldots, u(N), y(N)\}.$$

The linear shift operator q^{-1} is defined such that its application on a signal $k(t)$ is described by

$$q^{-1}\, k(t) = k(t-1).$$

The general form of a SISO LTI model can then be written as

$$A(q)y(t) = \frac{B(q)}{F(q)}\, u(t) + \frac{C(q)}{D(q)}\, e(t), \tag{1.1}$$

where $A(q), B(q), C(q), D(q), F(q)$ are the following polynomials in the shift operator q^{-1}:

$$\begin{aligned}
A(q) &= 1 + \textstyle\sum_{i=1}^{n_a}(a_i\, q^{-i}),\\
B(q) &= \textstyle\sum_{i=1}^{n_b}(b_i\, q^{-n_k-i+1}),\\
C(q) &= 1 + \textstyle\sum_{i=1}^{n_c}(c_i\, q^{-i}),\\
D(q) &= 1 + \textstyle\sum_{i=1}^{n_d}(d_i\, q^{-i}),\\
F(q) &= 1 + \textstyle\sum_{i=1}^{n_f}(f_i\, q^{-i}).
\end{aligned}$$

The number of coefficients of the polynomials $A(q), B(q), C(q), D(q), F(q)$ are n_a, n_b, n_c, n_d, n_f respectively. The degree of $B(q)$ includes a time delay of $n_k > 0$

samples. The basic idea behind PEMs involves the description of the LTI model as a one-step ahead predictor $\hat{y}(t|t-1,\theta)$ of the output $y(t)$. The parameter vector θ contains all coefficients of the polynomials in (1.1). The one-step ahead predictor is given by

$$\hat{y}(t|t-1,\theta) = \left[I - \frac{A(q)D(q)}{C(q)}\right] y(t) + \frac{B(q)D(q)}{C(q)F(q)} u(t).$$

Prediction errors $e(t,\theta)$ are then defined as

$$\begin{aligned} e(t,\theta) &= y(t) - \hat{y}(t|t-1,\theta), \\ &= \frac{A(q)D(q)}{C(q)} y(t) - \frac{B(q)D(q)}{C(q)F(q)}\, u(t). \end{aligned} \tag{1.2}$$

The maximal lag n of (1.2) determines how many times this expression can be written, given Z^N. From rewriting (1.2) as

$$C(q)F(q)e(t,\theta) = A(q)D(q)F(q)y(t) - B(q)D(q)u(t), \tag{1.3}$$

the maximal lag n is found as

$$n = \max(n_a + n_d + n_f, n_k + n_b + n_d - 1, n_f + n_c).$$

Estimates for the model parameters, given Z^N, are then found by minimizing the prediction errors. Or in other words, as solutions of the following optimization problem

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{t=1}^{N} l(e(t,\theta)) \tag{1.4}$$

subject to (1.2) where l refers to a suitable norm. We will assume the quadratic norm $l(e) = e^2/2$. By using Lagrange multipliers $\lambda_1, \ldots, \lambda_{N-n}$, the constraints (1.2) can be added to the cost function

$$\sum_{t=1}^{N} \frac{e(t,\theta)^2}{2N} + \sum_{t=n+1}^{N} \lambda_{t-1} \left[C(q)F(q)e(t,\theta) - A(q)D(q)F(q)y(t) + B(q)D(q)u(t)\right]. \tag{1.5}$$

The cost function (1.5) is clearly polynomial which shows that for the chosen norm PEMs correspond to a polynomial optimization problem. Taking the partial derivatives of (1.5) with respect to every unknown, viz. the model parameters and Lagrange multipliers, and equating this to zero results in a multivariate polynomial system of $2N - n + n_a + n_b + n_c + n_d + n_f$ equations and unknowns. PEMs are therefore mathematically equivalent to solving a multivariate polynomial system. It is however possible to simplify the problem. Examining the cost function (1.5) reveals that the prediction errors $e(t,\theta)$ occur quadratically. Hence, each partial derivative with respect to an unknown $e(t,\theta)$ will result in an equation which is

linear in that $e(t,\theta)$. This means that each of the N prediction errors can be easily eliminated from the polynomial system resulting in $N-n+n_a+n_b+n_c+n_d+n_f$ equations and unknowns. The cost for eliminating N equations and variables however is the increase of the degrees of the remaining polynomials. Equation (1.3) reveals that the degree will increase with n_f+n_c+1 where the $+1$ term is due to the Lagrange multiplier.

Observe that we have established that in the PEM framework one needs to solve a multivariate polynomial system in order to find estimates for the model parameters. Multivariate polynomial systems typically have many solutions, all corresponding with optimal solutions of the optimization problem 1.4. If we can find all solutions of the polynomial system, then we also have the global optimum. Even better would be, of course, to compute only the global optimum.

Output error model identification

In Chapter 6, using our affine root-finding algorithm, we will find the globally optimal parameter estimates from (1.4) for the second order Output Error model

$$y(t) \;=\; \frac{0.2q^{-1}}{(1-1.6\,q^{-1}+.89\,q^{-2})}u(t)+e(t). \tag{1.6}$$

The system is excited with a white-noise input $u(t)$ of 6 samples. This is a very small amount of samples due to the fact that the number of polynomials and variables in the polynomial system scale with the total number of samples N. These samples are obtained from a zero-mean Gaussian distribution with a standard deviation of 10. The system output $y(t)$ is then calculated from (1.6) using zero-mean Gaussian white noise $e(t)$ with a standard deviation of 0.1. Given the observed data $u(1),\ldots,u(6),y(1),\ldots,y(6)$, we estimate the parameters of the following model

$$y(t) \;=\; \frac{b_1\,q^{-1}}{(1-f_1\,q^{-1}+f_2\,q^{-2})}u(t)+e(t).$$

The unknown parameters that need to be estimated in this problem are

$$b_1, f_1, f_2, e(1), e(2), e(3), e(4), \lambda_1, \lambda_2, \lambda_3, \lambda_4.$$

The cost function corresponding with finding all parameters of (1.6) is

$$\begin{aligned}
&\tfrac{1}{12}\,(e(1)^2+e(2)^2+e(3)^2+e(4)^2+e(5)^2+e(6)^2)+\\
&\lambda_1\;(y(3)+f_1\,y(2)+f_2\,y(1)-b_1\,u(2)-e(3)-f_1\,e(2)-f_2\,e(1))+\\
&\lambda_2\;(y(4)+f_1\,y(3)+f_2\,y(2)-b_1\,u(3)-e(4)-f_1\,e(3)-f_2\,e(2))+\\
&\lambda_3\;(y(5)+f_1\,y(4)+f_2\,y(3)-b_1\,u(4)-e(5)-f_1\,e(4)-f_2\,e(3))+\\
&\lambda_4\,(y(6)+f_1\,y(5)+f_2\,y(4)-b_1\,u(5)-e(6)-f_1\,e(5)-f_2\,e(4)).
\end{aligned} \tag{1.7}$$

Setting the partial derivatives of the cost function (1.7) with respect to the prediction errors $e(1), \ldots, e(6)$ to zero results in the following linear expressions:

$$\begin{aligned}
e(1) &= 6\,\lambda_1\,f_2,\\
e(2) &= 6\,\lambda_1\,f_1 + 6\,\lambda_2\,f_2,\\
e(3) &= 6\,\lambda_1 + 6\,\lambda_2 f_1 + 6\,\lambda_3\,f_2,\\
e(4) &= 6\,\lambda_2 + 6\,\lambda_3\,f_1 + 6\,\lambda_4\,f_2,\\
e(5) &= 6\,\lambda_3 + 6\,\lambda_4\,f_1,\\
e(6) &= 6\,\lambda_4.
\end{aligned}$$

By substituting these expressions into the remaining partial derivatives we have effectively eliminated the prediction errors from the problem and obtain the following multivariate polynomial system of 7 polynomials in 7 unknowns

$$\left\{\begin{array}{l}
y(3) + f_1 y(2) + f_2 y(1) - b_1 u(2) - 6\lambda_1 - 6\lambda_2 f_1 - 6\lambda_3 f_2\\
\quad -6\lambda_1 f_1^2 - 6 f_1 \lambda_2 f_2 - 6 f_2^2 \lambda_1 = 0,\\
\\
y(4) + f_1 y(3) + f_2 y(2) - b_1 u(3) - 6\lambda_2 - 6\lambda_3 f_1 - 6\lambda_4 f_2\\
\quad -6\lambda_1 f_1 - 6\lambda_2 f_1^2 - 6 f_1 \lambda_3 f_2 - 6 f_2 \lambda_1 f_1 - 6\lambda_2 f_2^2 = 0,\\
\\
y(5) + f_1 y(4) + f_2 y(3) - b_1 u(4) - 6\lambda_3 - 6\lambda_4 f_1 - 6\lambda_2 f_1\\
\quad -6\lambda_3 f_1^2 - 6 f_1 \lambda_4 f_2 - 6\lambda_1 f_2 - 6 f_1 \lambda_2 f_2 - 6\lambda_3 f_2^2 = 0,\\
\\
y(6) + f_1 y(5) + f_2 y(4) - b_1 u(5) - 6\lambda_4 - 6\lambda_3 f_1 - 6\lambda_4 f_1^2\\
\quad -6\lambda_2 f_2 - 6 f_1 \lambda_3 f_2 - 6\lambda_4 f_2^2 = 0,\\
\\
\lambda_1 y(2) - 6\lambda_1^2 f_1 - 6\lambda_1 \lambda_2 f_2 + \lambda_2 y(3) - 6\lambda_2 \lambda_1 - 6\lambda_2^2 f_1\\
\quad -6\lambda_2 \lambda_3 f_2 + \lambda_3 y(4) - 6\lambda_3 \lambda_2 - 6\lambda_3^2 f_1 - 6\lambda_3 \lambda_4 f_2\\
\quad +\lambda_4 y(5) - 6\lambda_4 \lambda_3 - 6\lambda_4^2 f_1 = 0,\\
\\
\lambda_1 y(1) - 6\lambda_1^2 f_2 + \lambda_2 y(2) - 6\lambda_2 \lambda_1 f_1 - 6\lambda_2^2 f_2 + \lambda_3 y(3)\\
\quad -6\lambda_3 \lambda_1 - 6\lambda_3 \lambda_2 f_1 - 6\lambda_3^2 f_2 + \lambda_4 y(4) - 6\lambda_4 \lambda_2\\
\quad -6\lambda_4 \lambda_3 f_1 - 6\lambda_4^2 f_2 = 0,\\
\\
-\lambda_1 u(2) - \lambda_2 u(3) - \lambda_3 u(4) - \lambda_4 u(5) = 0.
\end{array}\right. \tag{1.8}$$

1.1.2 Maximum Likelihood Estimation of Mixtures of Multinomial Distributions

Finding the maximum likelihood estimates of the model parameters of a mixture of multinomial distributions also corresponds with solving a multivariate polynomial system. The derivation is quite straightforward. As an application of the maximum likelihood estimation of mixtures of multinomial distributions we will discuss the detection of CpG islands in DNA.

Maximum likelihood estimation is polynomial system solving

Let n denote the number of distributions in the mixture and K the total number of possible outcomes in an experiment. Each ith multinomial distribution is characterized by K probabilities $p(k|i)$ with $i = 1 \ldots n$ and $k = 1 \ldots K$. These are assumed to be known. The probability of an observed outcome y_k is then given by

$$p_{y_k}(x) \;=\; x_1\, p(k|1) + \ldots + x_n\, p(k|n) \;=\; \sum_{i=1}^{n} x_i\, p(k|i), \tag{1.9}$$

where $x = (x_1, \ldots, x_n)$ are the unknown mixing probabilities. Data are typically given as a sequence of N observations. When all observations are independent and identically distributed, the data can then be summarized in a data vector $u = (u_1, \ldots, u_K)$. Each u_k is the number of times the outcome y_k is observed in the sequence of N observations. We therefore have that $u_1 + u_2 + \ldots + u_K = N$. The likelihood function is then defined as follows.

Definition 1.1 *Given a mixture of n multinomial distributions and a sequence of N independent and identical distributed samples then the likelihood function $L(x)$ is given by*

$$L(x) = p_{y_1}(x)^{u_1} p_{y_2}(x)^{u_2} \ldots p_{y_K}(x)^{u_K} = \prod_{i=1}^{K} p_{y_i}(x)^{u_i}. \tag{1.10}$$

This likelihood depends on the parameter vector x and data vector u and is therefore called the likelihood function. Observe that it is the assumption of independent and identical distributed observations that allows us to factorize the likelihood. Any reordering of the observations leads to the same data vector u and has therefore no effect on the likelihood function. Multiplying probabilities leads to very small numbers, which could result in numerical underflow. By taking the logarithm of (1.10) the expression is changed to

$$l(x) = \log L(x) = \sum_{i=1}^{K} u_i \log p_{y_i}(x),$$

which effectively transforms the product of probabilities into a sum. This avoids the occurrence of any numerical underflow when multiplying probabilities. The maximum log-likelihood estimate of x is the solution of the following optimization problem

$$\hat{x} = \underset{x}{\operatorname{argmax}}\, l(x) \tag{1.11}$$

which is equivalent with maximizing $L(x)$ since the logarithm is a monotonic function. The optimization problem (1.11) is solved by taking the partial derivatives of $l(x)$ with respect to each x_i and setting these equal to zero. This results in the following system of n rational equations in n unknowns

$$\left\{ \begin{array}{ccccc} \frac{\partial l(x)}{\partial x_1} & = & \sum_i \frac{u_i}{p_{y_i}} \frac{\partial p_{y_i}}{\partial x_1} & = & 0, \\ \vdots & & & & \\ \frac{\partial l(x)}{\partial x_n} & = & \sum_i \frac{u_i}{p_{y_i}} \frac{\partial p_{y_i}}{\partial x_n} & = & 0. \end{array} \right. \tag{1.12}$$

These are rational equations since each term contains a linear polynomial of the form (1.9) in the denominator. Therefore, in order to find the solutions of (1.12) all terms for each equation need to be put onto a common denominator. To improve the readability the dependencies of p_k on x were dropped in the notation. Like for the system identification problem, such a polynomial system will have many solutions. The global optimum can then be found by evaluating the log-likelihood in all solutions that satisfy the constraints $0 \leq x_1, \ldots, x_n \leq 1$. Also observe that it is in fact possible to eliminate 1 unknown. Using the relation $x_1 + \ldots + x_n = 1$, it is possible to reduce the number of equations and unknowns to $n - 1$.

Detection of CpG islands in DNA

One application of a mixture model is the detection of CpG islands in DNA. CpG islands are genomic regions that contain a high frequency of sites where a cytosine (C) base in the DNA sequence is followed by a guanine (G) [13]. When in the human genome the CG dinucleotide (often written as CpG) occurs, the C nucleotide is typically chemically modified by methylation. This methyl-C in its turn has a high probability of turning into a T, with the consequence that CpG dinucleotides are more rare in the genome data than would be expected. Regions of DNA that contain a high frequency of CpG would therefore indicate that there is some selective pressure to keep them. This explains why CpG islands are usually found around the promotors of many genes. They are typically a few hundred to a few thousand bases long. In order to find a solution to the given problem we make the following simplification: instead of focusing specifically on the occurrence of CpG's we count the occurrences of C and G instead. In this case, the total number of possible outcomes K is 4, the 4 building blocks of DNA: $\{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$. A mixture model of the DNA sequence is set up that mixes 3 distributions. Each one of these

Table 1.1: probabilities for each of the distributions.

DNA Type	A	C	G	T
CG rich	0.15	0.33	0.36	0.16
CG poor	0.27	0.24	0.23	0.26
CG neutral	0.25	0.25	0.25	0.25

distributions represents a certain type of DNA: CG rich, CG poor and CG neutral. The first type, CG rich, stands for DNA that is rich in both the C and G bases. Therefore, the C and G bases sampled from this distribution will have higher probabilities to occur relative to those for A and G. In a similar fashion the CG poor and CG neutral type are characterized by specific probabilities. The complete model is summarized in Table 1.1 and is obtained from [34] and [69]. Now suppose that the following sequence of DNA is observed

CTCACGTGATGAGAGCATTCTCAGA
CCGTGACGCGTGTAGCAGCGGCTCA.

These observations can be summarized in the data vector $u = (11, 14, 15, 10)$. Deciding whether this particular segment came from a CpG island is based on the occurrences of both the C and G bases. This information is encoded in the mixing probabilities: x_1, x_2 and x_3. These are the probabilities with which samples are drawn from either the CG rich, poor or neutral distribution respectively and comparing them relative to one another will allow us to answer the given problem. For any general mixture model of k mixtures, the probability of making a certain observation y is

$$p(y) \;=\; \sum_{i=1}^{k} x_i \, p(y \,|\, i),$$

where $p(y|i)$ is the probability to observe y given that it is sampled from distribution i. The probabilities of observing each of the bases A to T are therefore given by

$$\begin{aligned}
p(A) &= 0.15\,x_1 + 0.27\,x_2 + 0.25\,x_3, \\
p(C) &= 0.33\,x_1 + 0.24\,x_2 + 0.25\,x_3, \\
p(G) &= 0.36\,x_1 + 0.23\,x_2 + 0.25\,x_3, \\
p(T) &= 0.16\,x_1 + 0.26\,x_2 + 0.25\,x_3,
\end{aligned}$$

which can be reduced to

$$\begin{aligned} p(A) &= -0.10\,x_1 + 0.02\,x_2 + 0.25, \\ p(C) &= +0.08\,x_1 - 0.01\,x_2 + 0.25, \\ p(G) &= +0.11\,x_1 - 0.02\,x_2 + 0.25, \\ p(T) &= -0.09\,x_1 + 0.01\,x_2 + 0.25, \end{aligned}$$

since $x_1 + x_2 + x_3 = 1$. Each probability is described by a first order polynomial. The maximum likelihood estimators for x_1, x_2 and x_3 are found from the following optimization problem

$$\underset{x_1,x_2,x_3}{\operatorname{argmax}}\, l(x_1, x_2, x_3)$$

where the log-likelihood is given by

$$l(x_1, x_2, x_3) = 11 \log p(A) + 14 \log p(C) + 15 \log p(G) + 10 \log p(T).$$

The polynomial system that corresponds with (1.12) for this problem is therefore

$$\begin{cases} -0.0289\,x_1 + 0.0047\,x_2 + 0.00396\,x_1^3 - 0.0000136\,x_2^3 + 0.0120 - 0.00131\,x_1^2 \\ +\,0.000378 x_1\,x_2 - 0.0000357\,x_2^2 - 0.00183\,x_1^2\,x_2 + 0.000276\,x_1\,x_2^2 = 0, \\ \\ 0.0047\,x_1 - 0.0008\,x_2 - 0.00062\,x_1^3 + 0.000002\,x_2^3 - 0.00187 + 0.00017\,x_1^2 \\ -0.000056\,x_1\,x_2 + 0.000006\,x_2^2 + 0.00028\,x_1^2\,x_2 - 0.000041\,x_1\,x_2^2 = 0. \end{cases}$$

In Chapter 6, we will solve this polynomial system, find the estimates for x_1, x_2, x_3 and decide whether the observed DNA sequence came from a CpG island.

1.1.3 Blind Image Deconvolution

In this section we will show how blind image deconvolution can be achieved by the computation of a multivariate polynomial greatest common divisor (GCD). This approach was first developed in [71], from which we give the following outline of the problem. In many problems including communication, satellite imaging, and synthetic aperture radar (SAR), the output observations consist of a desired input that has been distorted by a blurring function. The output is typically described as a convolution of an input data stream and the channel impulse response, both of which are unknown. Similarly, in ordinary blurred images, the final picture can be represented as a convolution between the desired picture and a blurring function that results from camera motion and/or slow shutter speed. Blind identification then consists of determining the input function and blurring function from the output observation. Let $p(i, j)$ represent the desired image, $d(i, j)$ the blurring

function, and $n(i,j)$ additive noise. If $f(i,j)$ represents the noisy blurred image we can write

$$f(i,j) \;=\; p(i,j) \,*\, d(i,j) + n(i,j),$$

where $*$ stands for the convolution operator. This expression can be written, using 2D z-transforms as the polynomial

$$F(z_1,z_2) \;=\; P(z_1,z_2)\,D(z_1,z_2) + N(z_1,z_2). \tag{1.13}$$

It is clear from (1.13) that even in the absence of noise, knowing the output $F(z_1,z_2)$ alone is not sufficient to obtain the original image or blurring function. In practice, it is either too costly to obtain a priori information about the imaged scene and the distortion (as in astronomy or x-ray imaging) or it is simply impossible (as in real-time video conferencing). Let us first assume there is no additive noise and that there are two distinct blurring functions $D_1(z_1,z_2)$ and $D_2(z_1,z_2)$. Then we have two blurred outputs

$$F_1(z_1,z_2) = P(z_1,z_2)\,D_1(z_1,z_2) \;\text{ and }\; F_2(z_1,z_2) = P(z_1,z_2)\,D_2(z_1,z_2).$$

Suppose now that the two blurring functions are relatively co-prime, then the desired image $P(z_1,z_2)$ can be retrieved as the GCD of $F_1(z_1,z_2)$ and $F_2(z_1,z_2)$. Or, in other words, if

$$\mathrm{GCD}(D_1(z_1,z_2), D_2(z_1,z_2)) \;=\; 1,$$

then

$$P(z_1,z_2) \;=\; \mathrm{GCD}(F_1(z_1,z_2), F_2(z_1,z_2)).$$

The restriction that $D_1(z_1,z_2), D_2(z_1,z_2)$ are co-prime means that they do not need to be free of common zeros. Indeed, take for example $D_1(z_1,z_2) = z_1$ and $D_2(z_1,z_2) = z_2$. They both share the common zero $(0,0)$ but are also co-prime. Although the above formulation requires two distinct blurred images, it is important to realize that this information can often be retrieved from a single image. Assuming that the blurring function $D(z_1,z_2)$ has a much smaller support size (degree) compared to the image itself, then it is possible to restrict our attention to two non-overlapping regions R_1 and R_2 that are large enough to contain the blurring function and are sufficiently apart. From these two regions, the blurring function $D(z_1,z_2)$ can then be estimated as

$$D(z_1,z_2) \;=\; \mathrm{GCD}(F_1(z_1,z_2), F_2(z_1,z_2)),$$

with

$$F_1(z_1,z_2) \in R_1 \text{ and } F_2(z_1,z_2) \in R_2.$$

It is clear that computing the GCD symbolically in the presence of noise will surely fail. Indeed, introducing a tiny perturbation to $F_1(z_1,z_2)$ or $F_2(z_1,z_2)$ would ensure that the retrieved image $\mathrm{GCD}(F_1(z_1,z_2), F_2(z_1,z_2)) = 1$. What is therefore required is the computation of an approximate GCD, which allows for the presence of noise. There are several formulations of approximate GCDs [12, 19, 36, 51, 68, 73, 94]. A common definition is the following.

Definition 1.2 *([12]) A polynomial g is said to be an ϵ-divisor of u and v if there exist polynomials $\hat{u}$ and $\hat{v}$ of degree n and m respectively, such that $||u - \hat{u}||_2 \leq \epsilon\,||u||_2$, $||v - \hat{v}||_2 \leq \epsilon\,||v||_2$ and g divides $\hat{u}$ and $\hat{v}$ and is of maximal degree.*

Or in other words, one allows for perturbations on u and v such that the perturbed polynomials $\hat{u}, \hat{v}$ have an exact GCD of maximal degree. We will provide our own formulation in Chapter 6 and explain how it is related to Definition 1.2. As an application of our approximate GCD algorithm we will apply blind deconvolution on two noisy images to reconstruct the desired image displayed in Figure 1.1 as good as possible.

Figure 1.1: **The desired image $p(i, j)$.**

The two filters applied to $P(z_1, z_2)$ are the low-pass filter

$$D_1(z_1, z_2) \;=\; 0.4694 + 0.0327\,z_1 + 0.5500\,z_2 + 0.4621\,z_1^2 - 0.2077\,z_1\,z_2 - 0.3066\,z_2^2,$$

and the high-pass filter

$$D_2(z_1, z_2) \;=\; -0.3318 + 1.1431\,z_1 - 1.9563\,z_2 + 0.4470\,z_1^2 + 0.9350\,z_1\,z_2 + 0.7630\,z_2^2.$$

Random uniformly distributed noise is added to both $P(z_1, z_2)\,D_1(z_1, z_2)$ and $P(z_1, z_2)\,D_2(z_1, z_2)$ to obtain

$$F_1(z_1, z_2) \;=\; P(z_1, z_2)\,D_1(z_1, z_2) + N_1(z_1, z_2) \text{ (Figure 1.2)}$$

and

$$F_2(z_1, z_2) \;=\; P(z_1, z_2)\,D_2(z_1, z_2) + N_2(z_1, z_2) \text{ (Figure 1.3)}$$

with signal-to-noise ratios of 46dB for both images.

We will show in Chapter 6 that our approximate GCD algorithm can indeed retrieve an image $\hat{p}(i, j)$ that lies close to the original $p(i, j)$.

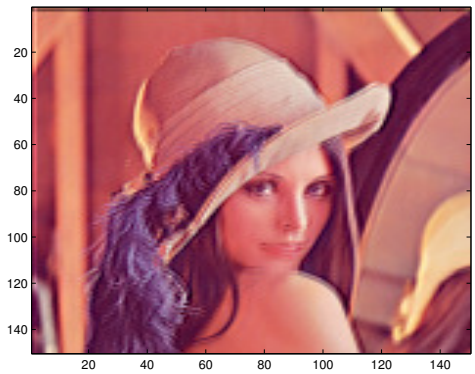

Figure 1.2: **The filtered and noisy image $f_1(i, j)$.**

Figure 1.3: **The filtered and noisy image $f_2(i, j)$.**

1.2 Contributions

In this section, we give an overview of the main contributions in this thesis. These contributions will be related to the appropriate chapters in the next section.

1.2.1 Polynomial Numerical Linear Algebra Framework

We develop a numerical linear algebra framework and numerical algorithms to solve the following problems:

1. polynomial addition, multiplication, division,
2. the computation of a Gröbner basis,
3. the computation of an approximate least common multiple (LCM) and greatest common divisor (GCD) of two multivariate polynomials,
4. the computation of all affine roots of a system of multivariate polynomials,
5. solving the ideal membership problem,
6. finding all syzygies and the degree of regularity,
7. multivariate polynomial elimination,
8. removing multiplicities of roots.

Furthermore, we show that, contrary to the symbolical methods in computer algebra, our numerical methods do not require an explicit computation of a Gröbner basis to solve problems 4 up to 8.

1.2.2 Bounds for the singular values and condition number of the multiplication matrix and Macaulay matrix

We derive bounds for the largest and smallest singular value and the condition number of the multiplication matrix in terms of the coefficients of the polynomials. Similarly, a bound is derived for the largest singular value of the Macaulay matrix. These bounds are then also interpreted as being bounds for the product and sum of products of multivariate polynomials. Furthermore, we are able to bound the perturbations of the singular values of the Macaulay matrix when its entries are perturbed by noise.

1.2.3 Recursive orthogonalization algorithm for the Macaulay matrix

The computation of orthogonal bases for the row space and null space of the Macaulay matrix is a central step in all algorithms of this thesis. The polynomial growth of the dimensions of the Macaulay matrix make this infeasible when the number of variables is large. We derive a recursive orthogonalization algorithm that addresses this issue by exploiting both the sparsity and the structure of the Macaulay matrix. Two implementations are provided: one using the SVD and one using a sparse rank-revealing QR decomposition.

1.2.4 Interpretations and the introduction of the canonical decomposition

The row space, null space and left null space of the Macaulay matrix are interpreted in terms of multivariate polynomials. These interpretations are valuable for all algorithms and bounds that are derived in this thesis. Since our framework uses only linear algebra, it is also possible to interpret the problems that are addressed in this thesis geometrically. For example, computing the quotient and remainder during polynomial division corresponds geometrically with oblique projections of a vector onto subspaces. In addition to these interpretations, we introduce the notion of the canonical and reduced canonical decomposition of the vector space $\mathcal{C}_d^n$.

1.3 Chapter overview of the thesis

We now present an overview of each of the chapters in relation to our contributions and mention for each chapter relevant publications. This overview is also represented visually in Figure 1.4.

Chapter 2: Basic Operations on Polynomials
This chapter lays the foundation of the PNLA framework by describing 3 basic operations on multivariate polynomials. These operations are addition, multiplication and division and are shown to correspond with vector addition, matrix-vector multiplication and vector decomposition respectively. Bounds for the largest and smallest singular value and the condition number of the multiplication matrix in terms of the coefficients of the polynomials are derived and the divisor matrix is introduced. Through this divisor matrix, the nonuniqueness of the quotient and remainder of polynomial division is explained and a geometric interpretation is given to polynomial division. We present our division algorithm

and demonstrate its effectiveness by means of numerical experiments. Finally, we discuss how the divisions of Buchberger's Algorithm, to compute a Gröbner basis, can be implemented using matrix reductions.

Publications related to this chapter: [6, 8].

Chapter 3: Macaulay matrix
In this chapter, we introduce the most important matrix of this thesis, the Macaulay matrix. We give expressions for its size and derive its density and an upper bound for its largest singular value. This upper bound is then applied to bound a sum of products of multivariate polynomials and to bound perturbations on the singular values of the Macaulay matrix. Furthermore, 3 important fundamental subspaces associated with the Macaulay matrix are discussed in detail: its row space, left null space and null space. We interpret the row space of the Macaulay matrix and discuss the related ideal membership problem of algebraic geometry. The link between the left null space and the notion of polynomial syzygies is revealed. Analyzing the growth of the left null space results in the important notion of the degree of regularity. We discuss the right null space in terms of the projective roots of the polynomial system and show how multiplicities of roots can be exchanged for extra polynomials. We also proof necessary conditions on the existence of zero roots and roots at infinity.

Publications related to this chapter: [7].

Chapter 4: Fast recursive orthogonalization of the Macaulay matrix
Orthogonal bases for both the row space and right null space of the Macaulay matrix play an important part in all of the algorithms developed in this thesis. A major bottleneck in the execution of these algorithms is the combinatorial growth of the Macaulay matrix. Fortunately, as discussed in the previous chapter, the matrix is very structured and very sparse. In this chapter we exploit both this structure and sparsity in developing a fast recursive orthogonalization scheme. This scheme allow us to update orthogonal bases for both the row space and right null space of the Macaulay matrix. Two implementations of the recursive scheme are discussed: one using a sparse rank-revealing QR decomposition and one using a full SVD. The computational complexity for both implementations is compared. We show through numerical experiments how the sparse implementation is far superior over the full SVD: total run time is 15 up to 119 times faster and a factor of 12 up to 500 times less storage space is required.

Publications related to this chapter: [5].

Chapter 5: The Canonical Decomposition of $\mathcal{C}_d^n$
In this chapter, we introduce both the canonical and reduced canonical decomposition and the notion of a pure power. These concepts are directly linked with

both a Gröbner basis and the number of roots of a polynomial system and will be needed when discussing several applications in Chapter 6. An algorithm is presented that produces these decompositions through the repetitive computation of intersections of subspaces. In addition, the effect of noise on the coefficients of the polynomials on the resulting canonical decompositions is investigated. It is shown that computing canonical decompositions is in fact an ill-posed problem. We also discuss how border bases are used to deal with this ill-posedness and provide an algorithm to compute them. The occurrence of pure powers in the reduced canonical decomposition will play a key role in finding stop criteria for the recursive algorithms in Chapter 6.

Publications related to this chapter: [7].

Chapter 6: Applications
In this chapter, we discuss 7 different applications and corresponding algorithms to solve them in our PNLA framework:

- **numerical computation of a Gröbner basis**: we reveal the relation between a Gröbner basis and the reduced canonical decomposition for the zero-dimensional case. This leads to a simple stop-criterion for our recursive Gröbner basis algorithm,
- **affine root-finding**: We show how the particular Vandermonde structure of the null space of the Macaulay matrix allows to compute all affine roots from an eigenvalue problem. A stop-criterion in terms of pure powers in the reduced canonical decomposition is also derived. The root-finding algorithm is then applied on both the system identification and maximum likelihood estimation problem from Section 1.1,
- **ideal membership problem**: we show how the occurrence of a Gröbner basis in the row space of the Macaulay matrix is essential for solving the ideal membership problem. We derive an expression for the degree at which the problem can be solved,
- **syzygy analysis**: we present a recursive algorithm that determines all syzygies and as a result determines the degree of regularity,
- **multivariate polynomial elimination**: we discuss the geometric interpretation of multivariate polynomial elimination and provide a numerical elimination algorithm,
- **approximate LCM/GCD**: we introduce our own geometric definition of an approximate LCM and GCD, together with corresponding algorithms. In addition, we illustrate the connection of our definition with the commonly used ϵ-GCD from the literature. Our approximate GCD algorithm is finally applied to the blind image deconvolution problem from Section 1.1,

- **radical ideal**: we develop an algorithm, based on a well-known theorem from algebraic geometry, that removes the multiplicities of all affine roots by adding extra polynomials to the polynomial system.

Publications related to this chapter: [3, 4, 6, 8, 33].

Chapter 7: Conclusion and Future Work
In this chapter, we summarize the thesis and give an overview of future research.

Appendix A: Numerical Linear Algebra
This appendix provides a short overview of all the numerical linear algebra concepts that are used in the thesis.

Appendix B: Algebraic Geometry
This appendix provides a short overview of all the algebraic geometry concepts that are used in the thesis.

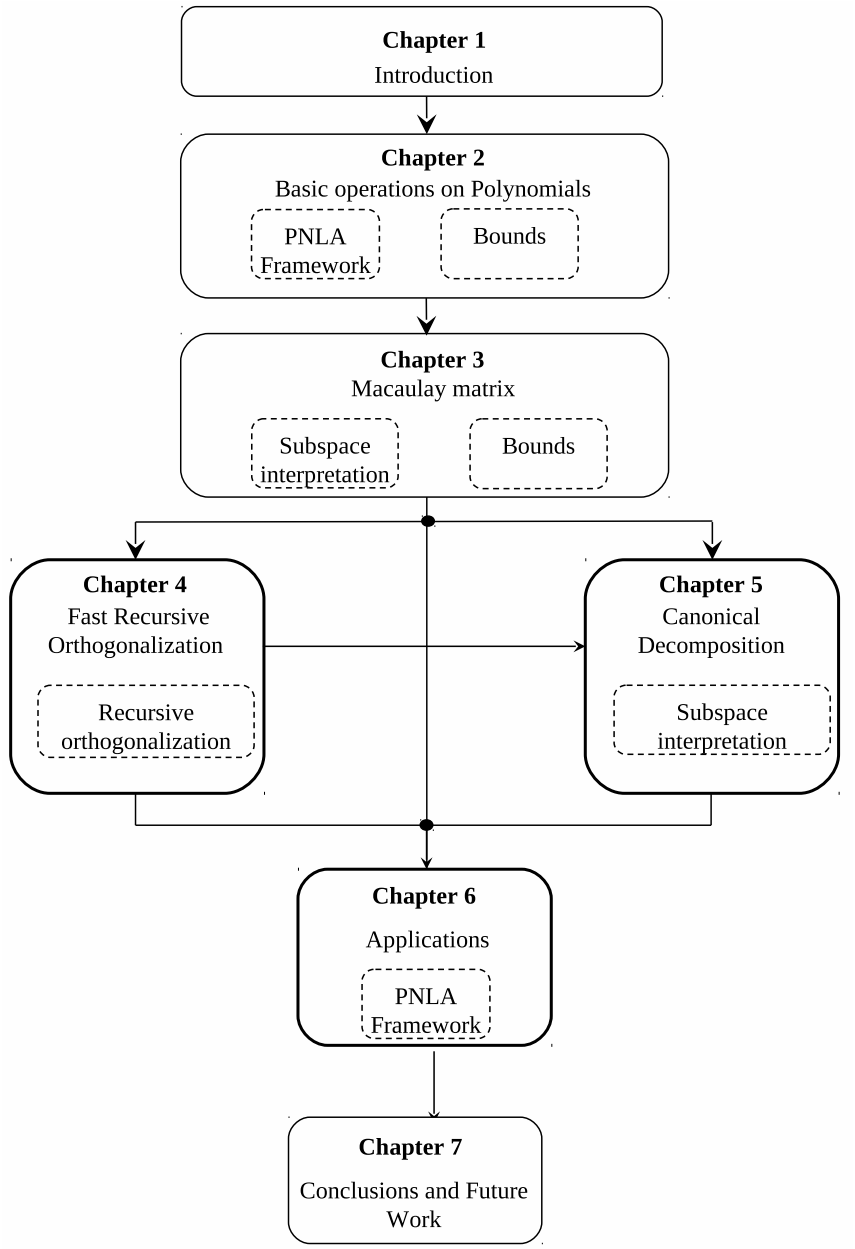

Figure 1.4: **An overview of the chapters and their interconnection. Contributions are mentioned for each of the chapters in dashed boxes.**

Chapter 2

Basic Operations on Polynomials

In this chapter, we will describe how the three basic operations of addition, multiplication and division of multivariate polynomials are described in the PNLA framework. In addition, bounds are derived for the largest and smallest singular value and the condition number of the multiplication matrix. The largest singular value of the multiplication matrix will bound the product of two multivariate polynomials and knowledge on its condition number will be crucial for the computation of approximate LCMs and GCDs in Chapter 6. In addition, a geometrical interpretation is given to multivariate polynomial division.

This chapter lays the foundation of the PNLA framework. Before discussing any of the three basic operations on polynomials, we first explain how multivariate polynomials are represented as vectors. It is also highly recommended to read Appendix A and B to refresh some basic knowledge on linear algebra and algebraic geometry.

2.1 Multivariate polynomials as vectors

The ring of multivariate polynomials in n variables is denoted by $\mathcal{C}^n$. It is easy to show that the subset of $\mathcal{C}^n$, containing all multivariate polynomials of total degrees from 0 up to d forms a vector space. We will denote this vector space by $\mathcal{C}_d^n$. Throughout this thesis we will use a monomial basis as a basis for $\mathcal{C}_d^n$. This monomial basis, together with a monomial ordering allows us then to represent a multivariate polynomial $f = \sum_\alpha f_\alpha x^\alpha$ by its coefficient vector.

Details on monomial orderings and the degree negative lexicographic ordering used throughout the thesis can be found in Appendix B Section B.1. The representation of a multivariate polynomial by a vector will be obtained by simply ordering the coefficients in a row vector according to the specified monomial ordering. The following example illustrates this for the degree negative lexicographic ordering.

Example 2.1 *The polynomial* $f = 2 + 3x_1 - 4x_2 + x_1x_2 - 8x_1x_3 - 7x_2^2 + 3x_2x_3$ *in* C_3^2 *can be written as*

$$f = \begin{pmatrix} 2 & 3 & -4 & 0 & 0 & 1 & -8 & -7 & 3 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \\ x_1^2 \\ x_1x_2 \\ x_1x_3 \\ x_2^2 \\ x_2x_3 \\ x_3^2 \end{pmatrix}.$$

From here on, we will work implicitly with the monomial basis and therefore represent a multivariate polynomial f*, by abuse of notation, solely by its coefficient vector*

$$f = \begin{array}{c} \begin{array}{cccccccccc} 1 & x_1 & x_2 & x_3 & x_1^2 & x_1x_2 & x_1x_3 & x_2^2 & x_2x_3 & x_3^2 \end{array} \\ \begin{pmatrix} 2 & 3 & -4 & 0 & 0 & 1 & -8 & -7 & 3 & 0 \end{pmatrix} \end{array}.$$

By convention, a coefficient vector will always be a row vector.

With this one-to-one correspondence between a multivariate polynomial f and its coefficient vector, the following relations hold:

$$\begin{aligned} ||f||_1 &= \textstyle\sum_\alpha |f_\alpha|, \\ ||f||_2 &= \sqrt{\textstyle\sum_\alpha |f_\alpha|^2}. \end{aligned}$$

These two norms can be calculated in double precision with high relative accuracy.

2.2 Multivariate polynomial Addition

The addition of two multivariate polynomials f_1, f_2 of degrees d_1, d_2 respectively is rather straightforward. This simply corresponds to vector addition. When the total degrees are different, then care needs to be taken to do the addition in the vector space $\mathcal{C}_d^n$ with $d = \max(d_1, d_2)$.

Example 2.2 *Let* $f_1 = -2+5x_1+9x_1x_2-4x_2^2$ *with* $d_1 = 2$ *and* $f_2 = 9+x_1+5x_2$ *with* $d_2 = 1$*, then their corresponding coefficient vectors in* $\mathcal{C}_2^2$ *are*

$$\begin{aligned} f_1 &= (\ -2 \quad 5 \quad 0 \quad 0 \quad 9 \quad -4\ ), \\ f_2 &= (\ \ 9 \quad 1 \quad 5 \quad 0 \quad 0 \quad\ \ 0\ ). \end{aligned}$$

The addition of the two coefficient vectors results in

$$f_1 + f_2 = \begin{pmatrix} 7 & 6 & 5 & 0 & 9 & -4 \end{pmatrix},$$

which corresponds with the polynomial

$$7 + 6\,x_1 + 5\,x_2 + 9\,x_1x_2 - 4\,x_2^2.$$

2.3 Multivariate Polynomial Multiplication

Next, we describe the multiplication of two multivariate polynomials in the PNLA framework. This will turn out to be a vector matrix multiplication and generalizes the convolution operation to the multivariate case. A generalization of the multiplication operator to multiple polynomials will play a crucial role in the remaining chapters.

2.3.1 Multiplication matrix

Given two polynomials $h, f \in \mathcal{C}_d^n$, then their product hf does not lie in $\mathcal{C}_d^n$ anymore. It is easy to derive that polynomial multiplication can be written in the PNLA framework as a vector matrix product. Supposing $\deg(h) = m$ we can write

$$\begin{aligned} h\,f &= (h_0 \ + \ h_1\,x_1 \ + h_2\,x_2 \ + \ldots \ + h_k\,x_n^m)\ f \\ &= h_0\,f \ + h_1\,x_1\,f \ + h_2\ x_2\,f \ + \ldots + h_k\,x_n^m\,f. \end{aligned}$$

This can be written as the following vector matrix product

$$h\,f = \begin{pmatrix} h_0 & h_1 & \ldots & h_m \end{pmatrix} \begin{pmatrix} f \\ x_1\,f \\ x_2\,f \\ \vdots \\ x_n^m\,f \end{pmatrix}, \tag{2.1}$$

where each row of the matrix in the right hand side of (2.1) is the coefficient vector of $f, x_1f, x_2f, \ldots, x_n^m f$ respectively and x_n^m is the leading monomial of h, also

denoted $\mathrm{LM}(h)$. The multiplication of f with a monomial results in all coefficients of f being shifted to the right in its corresponding coefficient vector. Therefore the matrix that is built up from the coefficients of f in expression (2.1) is a quasi-Toeplitz matrix. We call it a quasi-Toeplitz matrix because the coefficients will not lie on diagonals but close to the diagonal. So we use the word quasi-Toeplitz in the sense of almost or nearly Toeplitz. In the univariate case ($n = 1$) this multiplication matrix corresponds to the discrete convolution operator and is predominantly used in linear systems theory. The polynomial f is then interpreted as the impulse response of a linear time-invariant system and h as the input signal. For this univariate case, assuming $\deg(f) = n$, then writing out (2.1) results in

$$h\,f = \begin{pmatrix} h_0 & h_1 & \dots & h_m \end{pmatrix} \begin{pmatrix} f_0 & f_1 & f_2 & \dots & f_n & 0 & 0 & \dots & 0 \\ 0 & f_0 & f_1 & f_2 & \dots & f_n & 0 & \dots & 0 \\ 0 & 0 & f_0 & f_1 & f_2 & \dots & f_n & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & f_0 & f_1 & f_2 & \dots & f_n \end{pmatrix},$$

where the multiplication operator now is a Toeplitz matrix. We now formally define the multiplication operator of a given polynomial.

Definition 2.1 *Let $f \in \mathcal{C}^n$ with $deg(f) = d_0$, then its multiplication matrix of degree $d \geq d_0$ is the matrix containing the coefficients of*

$$M_f(d) \;=\; \begin{pmatrix} f \\ x_1 f \\ x_2 f \\ \vdots \\ x_n^{d-d_0} f \end{pmatrix} \tag{2.2}$$

where f is multiplied with all n-variate monomials from degree 0 up to $d - d_0$.

The total number of rows of the multiplication matrix is $\binom{d-d_0+n}{n}$ and its total number of columns is $\binom{d+n}{n}$. This means that $M_f(d)$ will always be an underdetermined matrix. We also have that

$$\mathrm{row}(M_f(d)) \;= \{hf \mid h, f \in \mathcal{C}^n : \deg(h) \leq d - \deg(f_i)\}. \tag{2.3}$$

Or in other words: its row space contains all polynomials hf with a maximal total degree of d. We will use the symbol $\mathcal{M}_f$ to denote $\mathrm{row}(M_f(d))$. The following example illustrates the multiplication of two polynomials in $\mathcal{C}_2^2$.

Example 2.3 *Let $h = x_1^2 + 2x_2 - 9$ and $f = x_1x_2 - x_2$. The leading monomial of h is x_1^2. The multiplication $h\,f$ is then given by*

$$h\,M_f(4) = = \begin{pmatrix} -9 & 0 & 2 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} f \\ x_1 f \\ x_2 f \\ x_1^2 f \\ x_1x_2 f \\ x_2^2 f \end{pmatrix},$$

where the multiplication operator $M_f(4)$ is

$$\begin{array}{c} \begin{array}{cccccccccccccc} 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 & x_1^3 & x_1^2x_2 & x_1x_2^2 & x_2^3 & x_1^4 & x_1^3x_2 & x_1^2x_2^2 & x_1x_1^3 & x_2^4 \end{array} \\ \begin{pmatrix} 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}.$$

Left-multiplying this matrix with the coefficient vector of h results in the vector

$$\begin{array}{c} \begin{array}{cccccccccccccc} 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 & x_1^3 & x_1^2x_2 & x_1x_2^2 & x_2^3 & x_1^4 & x_1^3x_2 & x_1^2x_2^2 & x_1x_1^3 & x_2^4 \end{array} \\ \begin{pmatrix} 0 & 0 & 9 & 0 & -9 & -2 & 0 & -1 & 2 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}, \end{array}$$

which is indeed the coefficient vector of $h\,f$.

The description of multiplication of multivariate polynomials in this linear algebra framework therefore leads in a natural way to the generalization of the convolution operation to the multivariate case [15, 66]. In the same way, multivariate polynomial division will generalize the deconvolution operation. Next, we derive an expression for the largest and smallest singular value and the condition number of the multiplication matrix.

2.3.2 Condition number

In this section, we derive bounds on the largest and smallest singular value and on the condition number of a multiplication matrix. The upper bound on the largest singular value will be a bound on the product of two multivariate polynomials and the condition number will be important for the computation of an approximate LCM and GCD. For a $p \times q$ multiplication matrix $M_{f_1}(d) = (m_{ij})$ we define the absolute row sum as

$$r_i = \sum_{1 \le j \le q} |m_{ij}|,$$

and the absolute column sum as

$$c_j = \sum_{1 \le i \le p} |m_{ij}|.$$

These two sums will be crucial for deriving the bounds. First we prove the following lemma in which we bound the absolute column sum of the multiplication matrix $M_{f_1}(d)$ in terms of the coefficients of f_1. In the following, $\deg(f)$ stands for the degree of the multivariate polynomial f.

Lemma 2.1 *Let $f_1 \in \mathcal{C}^n$ with*

- *$deg(f_1) = d_1$,*
- *$d_m = \min_{a_\alpha \neq 0} deg(a_\alpha x^\alpha)$,*
- *d_l the degree of the LCM of all monomials of f_1 with nonzero coefficients*

and $M_{f_1}(d) = (m_{ij})$ the $p \times q$ multiplication matrix of degree d. Then

$$\max_{1 \le j \le q} c_j \ \le ||f_1||_1$$

where equality is guaranteed from $d \ge d_l - d_m + d_1$.

Proof. The structure of the multiplication matrix ensures that each column can contain all coefficients of f_1 at most once. Hence the largest c_j that can be obtained is $||f_1||_1$. This happens when each nonzero coefficient of f_1 is shifted to the least common multiple of all its monomials with nonzero coefficients. The degree for which each of these monomials is shifted to its LCM is $d = d_l - d_m + d_1$. □

Example 2.4 *Suppose*

$$f_1 = 2\,x_2\,x_4 + 2\,x_1\,x_3 + x_2^2 - x_3,$$

then $d_1 = 2$ and $d_m = deg(x_3) = 1$. The least common multiple of all monomials with nonzero coefficients is $x_1\,x_2^2\,x_3\,x_4$ and hence $d_l = 5$. Using Lemma 2.1 we then have that

$$\max_{1 \le j \le q} c_j = ||f_1||_1 = 6$$

for $d \ge 5 - 1 + 2 = 6$.

We now prove the following upper bound on the largest singular value of $M_{f_1}(d)$.

Theorem 2.1 *Let $f_1 \in \mathcal{C}^n$ and $M_{f_1}(d)$ its corresponding $p \times q$ multiplication matrix of degree d. Then its largest singular value σ_1 is bounded by*

$$\sigma_1 \leq ||f_1||_1.$$

Proof. Schur [74] provided the following upper bound on the largest singular value

$$\sigma_1^2 \leq \max_{1 \leq i \leq p, 1 \leq j \leq q} r_i \, c_j.$$

Lemma 2.1 ensures that the maximal c_j is $||f_1||_1$. Each row of the multiplication matrix contains the same coefficients and therefore $r_i = ||f_1||_1$ for any row i. From this it follows that $\sigma_1 \leq ||f_1||_1$. □

Theorem 2.1 also provides an upper bound on the 2-norm of the product of two polynomials which is better in practice than the bound given in [10, p. 222].

Corollary 2.1 *Let $f_1, f_2 \in \mathcal{C}^n$ with degrees d_1, d_2 respectively, then the 2-norm of their product is bounded from above by*

$$||f_1 \, f_2||_2 \leq min\left(||f_1||_1 \, ||f_2||_2, ||f_2||_1 \, ||f_1||_2\right).$$

Proof. The commutativity of multiplying multivariate polynomials implies that the product $f_1 \, f_2$ can be computed using the multiplication matrix as either

$$f_1 \, f_2 \;=\; f_1 \, M_{f_2}(d_1 + d_2),$$

or

$$f_1 \, f_2 \;=\; f_2 \, M_{f_1}(d_1 + d_2).$$

Theorem 2.1 bounds these vectors in 2-norm from above by $||f_1||_2 \, ||f_2||_1$ and $||f_1||_1 \, ||f_2||_2$ respectively. □

It is also possible to derive a lower bound on the smallest singular value of the multiplication matrix.

Theorem 2.2 *Let $f_1 \in \mathcal{C}^n$ and $M_{f_1}(d) = (m_{ij})$ its corresponding $p \times q$ multiplication matrix of degree d. Then its smallest singular value σ_{min} is bounded from below by*

$$\sigma_{min} \geq 2|m_{00}| - ||f_1||_1.$$

PROOF. Johnson [48] provided the following bound for the smallest singular value of a $p \times q$ matrix with $p \leq q$

$$\sigma_{\min} \geq \min_{1 \leq i \leq p} \left\{ |m_{ii}| - \frac{1}{2}(\sum_{j \neq i}^{p} |m_{ij}| + \sum_{j \neq i}^{p} |m_{ji}|) \right\}. \tag{2.4}$$

The structure of the multiplication matrix ensures that every m_{ii} equals m_{00} for all rows i. We can therefore rewrite (2.4) as

$$\sigma_{\min} \geq |m_{00}| - \frac{1}{2} \max_{1 \leq i \leq p} \left\{ \sum_{j \neq i}^{p} |m_{ij}| + \sum_{j \neq i}^{p} |m_{ji}| \right\}. \tag{2.5}$$

The maximal absolute column and row sum of the leftmost $p \times p$ block of $M_{f_1}(d)$ is bounded by

$$\sum_{j \neq i}^{p} |m_{ij}| + \sum_{j \neq i}^{p} |m_{ji}| \leq 2\,||f_1||_1 - 2\,|m_{00}|.$$

The $-2\,|m_{00}|$ term comes from the fact that the diagonal element m_{00} cannot be counted in these sums. Using this bound for the maximal absolute column and row sum into (2.5) results in

$$\begin{aligned} \sigma_{\min} &\geq |m_{00}| - \tfrac{1}{2} \max_{1 \leq i \leq p} \left\{ \textstyle\sum_{j \neq i}^{p} |m_{ij}| + \sum_{j \neq i}^{p} |m_{ji}| \right\}, \\ &\geq |m_{00}| - \tfrac{1}{2}\,(2\,||f_1||_1 - 2\,|m_{00}|), \\ &\geq 2\,|m_{00}| - ||f_1||_1, \end{aligned}$$

which concludes the proof. □

This lower bound is trivial when $2|m_{00}| \leq ||f_1||_1$. From Theorem 2.1 and Theorem 2.2 the following upper bound for the condition number of $M_{f_1}(d)$ can be derived.

Corollary 2.2 *Let $f_1 \in \mathcal{C}^n$ and $M_{f_1}(d)$ its corresponding $p \times q$ multiplication matrix of degree d, then its condition number $\kappa_1(d)$ is bounded from above for the nontrivial case by*

$$\kappa_1(d) \leq \frac{||f_1||_1}{2|m_{00}| - ||f_1||_1}. \tag{2.6}$$

This upper bound can be quite an overestimation. In practice, the condition number grows very slowly as a function of the degree d.

Example 2.5 *Let*

$$f_1 = 55 + 9\,x_1 + 8\,x_2 + 7\,x_3 + 6\,x_1^2 + 5\,x_1\,x_2 + 4\,x_1\,x_3 + 3\,x_2^2 + 2\,x_2\,x_3 + x_3^2,$$

then $||f_1||_1 = 100$ *and* $m_{00} = 55$ *and therefore*

$$\kappa_1 \leq \frac{100}{2 \times 55 - 100} = 10.$$

Figure 2.1 displays the graphs of both the upper bound and the condition number as a function of the degree d. The condition number starts from 1, since for $d = d_1, \sigma_1 = \sigma_{min} = ||f_1||_2$, *and then grows sublinearly. Although the condition number seems to converge to 3 in the limit for large d, this is not the case.*

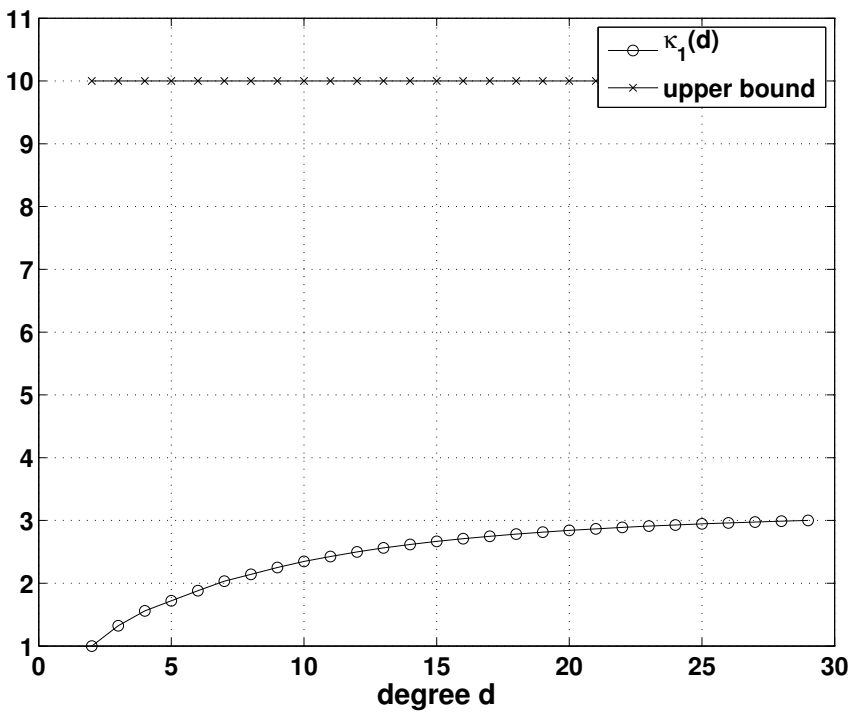

Figure 2.1: **The condition number and its upper bound for** $M_{f_1}(d)$ **in Example 2.5 as functions of the degree** d**.**

The observation that the condition number of $M_f(d)$ grows slowly will be important for the computation of an approximate GCD in Chapter 6. There, a least-squares solution $\hat{h}$ of the overdetermined linear system

$$M_f(d)^T\, h^T = l^T$$

will be computed. A small condition number of $M_f(d)$ then ensures a high relative accuracy for the computed solution $\hat{h}$.

2.4 Multivariate Polynomial Division

Everybody is familiar with the polynomial division for the univariate case. It is therefore quite surprising that this was generalized to the multivariate case only 40 years ago [22]. In contrast to addition and multiplication, the division of multivariate polynomials is a bit more involved. Similar to multiplication, a specific matrix operator for division can be defined. However, the operation of dividing a multivariate polynomial by a set of multivariate divisors will not correspond to a vector matrix product. We will show in this section that the division of multivariate polynomials corresponds to a vector decomposition using oblique projections. We start with the formal definition in which $\mathrm{LM}(p)$ denotes the leading monomial of a multivariate polynomial p with respect to a specified monomial ordering.

Definition 2.2 *([22, p. 64]) Fix any monomial order $>$ on $\mathcal{C}^n$ and let $F = (f_1, \ldots, f_s)$ be a s-tuple of polynomials in $\mathcal{C}_d^n$. Then every $p \in \mathcal{C}^n$ can be written as*

$$p \;=\; h_1 f_1 + \ldots + h_s f_s + r \tag{2.7}$$

where $h_1, \ldots, h_s, r \in \mathcal{C}_d^n$. For each i, $h_i f_i = 0$ or $LM(p) \geq LM(h_i f_i)$, and either $r = 0$ or r is a linear combination of monomials, none of which is divisible by any of $LM(f_1), \ldots, LM(f_s)$.

The generalization lies obviously in extending the polynomials p and f in the univariate case to elements of $\mathcal{C}^n$ and sets of divisors F. The constraint on the remainder term for the univariate case, $\deg(r) < \deg(f)$, is also generalized. The biggest consequence of this new constraint is that the remainder can have a degree which is strictly higher than any of the divisors f_i. The division of multivariate polynomials has an interesting property that is not present in the univariate case: both the quotient and remainder depend on the order in which p is divided by the polynomials of F. The following example illustrates this nonuniqueness of both quotient and remainder.

Example 2.6 *([22, p. 67]) Dividing $p = x_1 x_2^2 - x_1$ first by $f_1 = x_1 x_2 + 1$ and then by $f_2 = x_2^2 - 1$ results in*

$$x_1 x_2^2 - x_1 \;=\; x_2\,(x_1 x_2 + 1) + 0\,(x_2^2 - 1) + (-x_1 - x_2).$$

The quotient is hence $x_1 x_2^2 + x_2$ and the remainder is $-x_1 - x_2$. If we now divide p, on the other hand, first by $f_2 = x_2^2 - 1$ and then by $f_1 = x_1 x_2 + 1$ we have

$$x_1 x_2^2 - x_1 \;=\; x_1\,(x_2^2 - 1) + 0\,(x_1 x_2 + 1) + 0.$$

The quotient is now $x_1x_2^2 - x_1$ and the remainder is 0.

We will now show how multivariate division is described in the PNLA framework by oblique projections. The nonuniqueness of both the quotient and the remainder will be linked to a rank-deficiency of a particular divisor matrix, which we define in the next section.

2.4.1 Divisor matrix

Definition 2.2 requires that we are able to describe, for a given polynomial $p \in \mathcal{C}^n$, a sum of the form $h_1f_1 + \ldots + h_sf_s$ where $h_1, \ldots, h_s, f_1, \ldots, f_s \in \mathcal{C}^n$ and where for each h_if_i $(i = 1, \ldots, s)$ the condition $\mathrm{LM}(p) \geq \mathrm{LM}(h_if_i)$ applies. The row space of the divisor matrix D will describe such a sum.

Definition 2.3 *Given a set of polynomials $f_1, \ldots, f_s \in \mathcal{C}^n$, each of degree $d_i\,(i = 1 \ldots s)$ and a polynomial $p \in \mathcal{C}^n$ of degree d, then the divisor matrix D is given by*

$$D = \begin{pmatrix} f_1 \\ x_1f_1 \\ x_2f_1 \\ \vdots \\ x_n^{k_1}f_1 \\ f_2 \\ x_1f_2 \\ \vdots \\ x_n^{k_2}f_2 \\ \vdots \\ x_n^{k_s}f_s \end{pmatrix}, \tag{2.8}$$

where each polynomial f_i is multiplied with all monomials x^{α_i} from degree 0 up to degree $k_i = deg(p) - deg(f_i)$ such that $LM(x^{\alpha_i}f_i) \leq LM(p)$.

Notice the similarity of the divisor matrix with the multiplication matrix. It can be constructed by stacking the multiplication matrices $M_{f_i}(d)$ on top of one another, after which rows for which $\mathrm{LM}(x^{\alpha_i}f_i) > \mathrm{LM}(p)$ are removed. Hence, we have that

$$\mathrm{row}(D) = \{q = \sum_{i=1}^{s} h_if_i \,|\, \deg(q) \leq \deg(p) \text{ and } \mathrm{LM}(h_if_i) \leq \mathrm{LM}(p)\, \forall i = 1, \ldots, s\}.$$

The row space of D will be denoted $\mathcal{D}$. It is clear that $\mathcal{D} \subset \mathcal{C}_d^n$ and that $\dim(\mathcal{D}) = \text{rank}(D)$. Each column of D contains the coefficients of a certain monomial and hence the number of columns of D, $\#\text{col}(D)$, corresponds with $\dim(\mathcal{C}_d^n)$. This divisor matrix will be the key to generalize multivariate polynomial division in terms of linear algebra.

Example 2.7 *The divisor matrix D for dividing $p = x_1x_2^2 - x_1$ by $f_1 = x_1x_2 + 1$ and $f_2 = x_2^2 - 1$ is*

$$D = \begin{array}{c} \\ f_1 \\ x_1\, f_1 \\ x_2\, f_1 \\ f_2 \\ x_1\, f_2 \end{array} \begin{array}{c} \begin{array}{cccccccccc} 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 & x_1^3 & x_1^2x_2 & x_1x_2^2 & x_2^3 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}.$$

Observe that this divisor matrix has no row corresponding with $x_2\, f_2$ since $LM(x_2\, f_2) = x_2^3 > LM(p)$.

By definition, the $\sum_{i=1}^{s} h_i f_i$ terms of (2.7) are described by the row space $\mathcal{D}$ of the divisor matrix. This allows us to rewrite (2.7) as the vector equation

$$p \;=\; h\, D + r,$$

which leads to the insight: multivariate polynomial division corresponds to a vector decomposition. The vector p is decomposed into $h\, D$, which lies in $\mathcal{D}$, and into r. Since p can be any element of $\mathcal{C}_d^n$ and $\mathcal{D}$ is a subspace of $\mathcal{C}_d^n$, it therefore follows that there exists a vector space $\mathcal{R}$ such that $\mathcal{D} \oplus \mathcal{R} = \mathcal{C}_d^n$. In general, there are many other subspaces $\mathcal{R}$ which are the complement of $\mathcal{D}$. The most useful $\mathcal{R}$ for multivariate polynomial division will be the vector space which is isomorphic with the quotient space $\mathcal{C}/\mathcal{D}$.

2.4.2 Quotient Space

Having defined the vector space $\mathcal{D}$, we now proceed to describe the vector space of remainder terms. We start with considering the following relationship, denoted by $\sim$, in $\mathcal{C}_d^n$:

$$\forall\, p, r \in \mathcal{C}_d^n : p \sim r \Leftrightarrow p - r \in \mathcal{D}.$$

It is easily shown that $\sim$ is a equivalence relationship and therefore partitions $\mathcal{C}_d^n$. Each of these partitions is an equivalence class

$$[p]_{\mathcal{D}} \;=\; \{r \in \mathcal{C}_d^n : p - r \in \mathcal{D}\}.$$

Since $p - r \in \mathcal{D}$, we can also write it as $p - r = h\,D$ and therefore

$$p \;=\; h\,D + r.$$

The addition of the constraint that either $r = 0$, or r is a linear combination of monomials, none of which is divisible by any of $\mathrm{LM}(f_1), \ldots, \mathrm{LM}(f_s)$ allows then for the interpretation of the elements of the equivalence class as the remainder terms. The set of all the equivalence classes $[p]_{\mathcal{D}}$ is denoted by $\mathcal{C}/\mathcal{D}$ and is also a vector space. In fact, one can find a vector space $\mathcal{R} \subset \mathcal{C}_d^n$, isomorphic with $\mathcal{C}/\mathcal{D}$, such that $\mathcal{D} \oplus \mathcal{R} = \mathcal{C}_d^n$. This implies that

$$\begin{aligned}
\dim(\mathcal{R}) &= \dim(\mathcal{C}/\mathcal{D}), \\
&= \dim(\mathcal{C}_d^n) - \dim(\mathcal{D}), \\
&= \tbinom{d+n}{n} - \mathrm{rank}(D), \\
&= \dim(\mathrm{null}(D)),
\end{aligned}$$

which allows to determine the dimension of $\mathcal{R}$ in a straightforward manner. All information on the dimensions of both $\mathcal{D}$ and $\mathcal{R}$ can be obtained from the rank of D. Since $\mathcal{R}$ is a finite-dimensional vector space, a monomial basis for $\mathcal{R}$ can be formally defined.

Definition 2.4 *Any set of monomials which forms a basis of a vector space $\mathcal{R}$ such that $\mathcal{R} \cong \mathcal{C}/\mathcal{D}$ and $\mathcal{R} \subset \mathcal{C}_d^n$ is called an R basis. The corresponding canonical basis of $\mathcal{R}$ in $\mathcal{C}_d^n$ is denoted R such that $\mathcal{R} = row(R)$.*

Since $\mathcal{R} \subset \mathcal{C}_d^n$, the canonical basis R needs to be a monomial basis. These basis monomials are in fact representatives of the equivalence classes of a basis for $\mathcal{C}/\mathcal{D}$. Although a polynomial basis could be chosen for R, this would make it significantly harder to require that every monomial of this basis should not be divisible by any of the leading monomials of $f_1, \ldots, f_s$. This will turn out to be easy for a monomial basis of R. Finding the basis monomials of $\mathcal{R}$ is equivalent with looking for a set of columns of the divisor matrix D which are linearly dependent with respect to all other columns. We will prove this when discussing the nonuniqueness of the remainder. Since $\dim(\mathcal{C}/\mathcal{D}) = \dim(\mathrm{null}(D))$, it must be possible to find $\binom{d+n}{n} - r$ linearly dependent columns with $r = \mathrm{rank}(D)$. In the univariate case, D is by construction of full row rank and hence $r = d - d_0 + 1$. The number of linearly dependent columns is then $(d+1) - (d - d_0 + 1) = d_0$. This is in fact linked with the fundamental theorem of algebra which states that an univariate polynomial of degree d_0 over the complex field has d_0 solutions. In the multivariate case, things are a bit more complicated. D is then neither of full row rank nor of full column rank. This will explain the nonuniqueness of both the quotients and remainder, as we will show next.

2.4.3 Nonuniqueness of quotient

Suppose the rank of the matrix D is r. In general, the matrix will not be of full row rank and there will be maximally $\binom{p}{r}$ possibilities of choosing r linearly independent rows. In practice, a basis for the row space of D is required for calculating the decomposition of p into $\sum_i h_i f_i$ terms. Therefore depending on which rows are chosen as a basis for $\mathcal{D}$, several decompositions are possible. Checking whether the quotient is unique hence involves a rank test of D. This fact is summarized in the following lemma.

Lemma 2.2 *The quotient h D when dividing p by a set of divisors F is unique, or in other words does not depend on the order of the divisors, when the corresponding divisor matrix D is of full row rank.*

The definition of multivariate polynomial division does not specify any constraints on how to choose a basis for $\mathcal{D}$. In Subsection 2.4.6, where we discuss the implementation of our division algorithm, it is explained how such a basis is chosen using a sparse rank-revealing QR decomposition. Choosing a basis for $\mathcal{R}$ is constrained by the definition of multivariate polynomial division, but not in such a way that only one possible basis is left.

2.4.4 Nonuniqueness of remainder

In the previous section we saw that the nonuniqueness of the quotient is due to the divisor matrix D not being of full row rank. In the same way, the nonuniqueness of the remainder is due to the divisor matrix D not being of full column rank. However, not every maximal set of linearly dependent columns of D, with respect to the remaining columns, is a possible candidate as a basis for the remainder space $\mathcal{R}$. Only the maximal sets of linearly dependent columns that satisfy the constraint that none of the corresponding monomials are divisible by any of $\mathrm{LM}(f_1), \ldots, \mathrm{LM}(f_s)$ are allowed. This constraint, in general, is not sufficient to reduce the number of possible bases of $\mathcal{R}$ to only one. The following simple example illustrates.

Example 2.8 *Suppose*

$$p = 9x_2^2 - x_1x_2 - 5x_2 + 6$$

is divided by

$$F = \{f_1 = x_2 - 3, f_2 = x_1x_2 - 2x_2\}.$$

Since LM(p) = x_2^2 one needs to construct the following divisor matrix

$$D = \begin{array}{c} \\ f_1 \\ x_1\, f_1 \\ x_2\, f_1 \\ f_2 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 \end{array} \\ \begin{pmatrix} -3 & 0 & 1 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 & 1 & 0 \\ 0 & 0 & -3 & 0 & 0 & 1 \\ 0 & 0 & -2 & 0 & 1 & 0 \end{pmatrix} \end{array}.$$

The null column corresponding with the monomial x_1^2 will surely be linearly dependent with respect to all other columns. The rank of D is 4 and any other column of D could be chosen as the second linearly dependent column. This gives the following set of possible bases for $\mathcal{R}$:

$$\{1, x_1^2\}, \{x_1, x_1^2\}, \{x_2, x_1^2\}, \{x_1^2, x_1x_2\}, \{x_1^2, x_2^2\}.$$

The leading monomials of f_1 and f_2 are, according to the degree negative lexicographic ordering, x_1x_2 and x_2 respectively. Therefore the set of possible bases for $\mathcal{R}$ is reduced to $\{\{1, x_1^2\}, \{x_1, x_1^2\}\}$ since neither 1 nor x_1 are divisible by x_1x_2 or x_2. Since D is of full row rank, this implies that the quotient is unique. The matrix R corresponding with the R basis $\{1, x_1^2\}$ is

$$R = \begin{array}{c} \begin{array}{cccccc} 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}.$$

The row space of R is indeed such that $\mathcal{R} \oplus \mathcal{D} = \mathcal{C}_2^2$.

From Example 2.8 it is clear that not every set of linearly dependent columns corresponds with an R basis that is suitable to describe multivariate polynomial division. The encoding of the graded monomial ordering in the columns of the divisor matrix allows us to find a suitable basis for $\mathcal{R}$ that satisfies the constraint that none of its monomials is divisible by any $\mathrm{LM}(f_i)$ $(i = 1 \ldots s)$. The key idea is to check each column for linear dependence with respect to all columns to its right, starting from the rightmost column. Before stating the main theorem, we first introduce some notation and prove a needed lemma. In what follows, a monomial will be called linearly (in)dependent when its corresponding column of the divisor matrix D is linearly (in)dependent with respect to another set of columns. Suppose the divisor matrix D has q columns. Then each column of D corresponds with a monomial $m_1, \ldots, m_q$ with $m_1 < m_2 < \ldots < m_q$ according to the monomial ordering. Suppose now that $\mathrm{rank}(D) = r$ and therefore $c = q - r$ linearly dependent monomials can be found. We now introduce the following high-level algorithm that finds a special set of linearly dependent monomials. In this algorithm each monomial label stands for a column vector of the divisor matrix D.

Algorithm 2.1 *Find a maximal set of linearly dependent monomials*
Input*: divisor matrix* D
Output*: a maximal set of linearly dependent monomials* l

```
l ← ∅
if m_q = 0 then
   l ← [l , m_q]
end if
for i = q − 1 : −1 : 1 do
   if m_i linearly dependent with respect to {m_{i+1}, ..., m_q} then
      l ← [l , m_i]
   end if
end for
```

Algorithm 2.1 finds a maximal set of monomials l which are linearly dependent with respect to all monomials to their right. We will label these c monomials of l as $l_1, \ldots, l_c$ such that $l_c < \ldots < l_2 < l_1$ according to the monomial ordering. The matrix D can then be visually represented as

$$D = \begin{array}{c} \begin{array}{ccccccccc} m_1 & \ldots & l_c & \ldots & l_k & \ldots & l_1 & \ldots & m_q \end{array} \\ \begin{pmatrix} & & \times & & \times & & \times & & \\ \ldots & \ldots & \times & \ldots & \times & \ldots & \times & \ldots & \ldots \\ & & \times & & \times & & \times & & \end{pmatrix} \end{array}.$$

Example 2.9 *We revisit the divisor matrix of Example 2.8 and apply Algorithm 2.1. For this simple example, checking the linear dependence was done using the SVD-based 'rank' command in MATLAB. A monomial* m_i *was considered to be linearly dependent as soon as the rank did not increase when adding its column to the matrix containing* $\{m_{i+1}, \ldots, m_q\}$*. It is easy to verify that this results in the following linearly dependent monomials:* $l_1 = x_1^2, l_2 = 1$*.*

The previous example indicates that Algorithm 2.1 produces the standard monomials of lowest total degree for the degree negative lexicographic ordering. We now prove this in the following lemma.

Lemma 2.3 *Given a divisor matrix* D *of rank* r *and the linearly dependent monomials* $l_1, \ldots, l_c$ *found from Algorithm 2.1. Then any other set of* c *linearly dependent monomials* $l'_1, \ldots, l'_c$ *with* $l'_1 > l'_2 > \ldots > l'_c$ *satisfies the following conditions:* $l'_1 \geq l_1, l'_2 \geq l_2, \ldots, l'_c \geq l_c$*.*

Proof. Let $\{l_k, \ldots, m_q\}$ denote the set of all monomials from l_k up to m_q for a certain $k \in \{1, \ldots, c\}$ and let q_1 denote the cardinality of $\{l_k, \ldots, m_q\}$. From Algorithm 2.1 we know that $\{l_k, \ldots, m_q\}$ contains k linearly dependent monomials and $q_1 - k$ linearly independent monomials. We now choose the largest k such

that $l'_k < l_k$. $\{l_k, \ldots, m_q\}$ will then contain at most $k-1$ l' monomials, which implies that there are at least $q_1 - k + 1$ linearly independent monomials in $\{l_k, \ldots, m_q\}$. This contradicts the fact that there are exactly $q_1 - k$ linearly independent monomials in $\{l_k, \ldots, m_q\}$. □

This lemma states that the R basis which is found from Algorithm 2.1 is of minimal total degree according to the degree negative lexicographic ordering. We can now prove the main theorem.

Theorem 2.3 *Consider a divisor matrix D. Then a suitable monomial basis for $\mathcal{R}$ is found by Algorithm 2.1. None of the monomials corresponding with the linearly dependent columns found in this way are divisible by any of the leading monomials of $f_1, \ldots, f_s$ and therefore serve as a basis for the vector space of remainder terms $\mathcal{R}$.*

PROOF. Since $\mathcal{D} \oplus \mathcal{R} = \mathcal{C}_d^n$, any multivariate polynomial $p \in \mathcal{C}_d^n$ can be decomposed into $\sum_{i=1}^{s} h_i f_i \in \mathcal{D}$, spanned by a maximal set of linearly independent rows of D, and $r \in \mathcal{R}$, spanned by the monomials $l_1, \ldots, l_c$ found from Algorithm 2.1. We can therefore write

$$p \;=\; \sum_{i=1}^{s} h_i \, f_i + r \quad \text{with } r = \sum_{i=1}^{c} a_i \, l_i \quad (a_i \in \mathbb{C}). \tag{2.9}$$

Suppose now that at least one of the monomials $l_1, \ldots, l_c$ is divisible by a leading monomial of one of the polynomials $f_1, \ldots, f_s$, say f_j. Let l_k be the monomial of highest degree which is divisible by $\mathrm{LM}(f_j)$. This implies that the division of $r - \sum_{i=1}^{k-1} a_i \, l_i$ by f_j can be written as

$$r - \sum_{i=1}^{k-1} a_i \, l_i \;=\; g f_j + r' \tag{2.10}$$

where $r' \neq 0$ and due to the definition (2.2) none of the monomials of r' are divisible by $\mathrm{LM}(f_j)$. In addition, all monomials $r'_1, \ldots, r'_t$ of r' satisfy $r'_i < l_k$ [22, p. 64-66]. By substituting (2.10) into (2.9) we have

$$\begin{aligned} p &= \textstyle\sum_{i=1}^{s} h_i \, f_i + r \\ &= \textstyle\sum_{i=1}^{s} h_i \, f_i + \sum_{i=1}^{k-1} a_i \, l_i + g f_j + r' \\ &= \textstyle\sum_{i=1}^{s} h'_i \, f_i + \sum_{i=1}^{k-1} a_i \, l_i + r'. \end{aligned}$$

From this last equation one can see that r' needs to contain $c - k + 1$ monomials none of which are divisible by any of the leading monomials of $f_1, \ldots, f_s$. If $\mathrm{LM}(r')$ is not divisible by any of the leading monomials of $f_1, \ldots, f_s$ then $\mathrm{LM}(r')$ is the new linearly dependent monomial l'_k. However, $l'_k < l_k$, which is a contradiction

according to Lemma 2.3. If LM(r') is divisible by any of the leading monomials of $f_1, \ldots, f_s$, then the division procedure as in (2.10) can be repeated, leading to the same contradiction. □

The duality between the linearly dependent columns of D and the linearly independent rows of its null space $K = \text{null}(D)$ implies the following corollary of Theorem 2.3.

Corollary 2.3 *A monomial basis for $\mathcal{R}$ can also be found from checking the rows of the null space of D for linear independence from top to bottom. None of the monomials corresponding with the linearly independent rows are divisible by any of the leading monomials of $f_1, \ldots, f_s$.*

Corollary 2.3 will be useful when discussing a practical implementation. In algebraic geometry, the nonuniqueness of the remainder corresponds with the remainder being dependent on the order of the divisors $f_1, \ldots, f_s$. This is normally solved by computing the remainder of p being divided by a Gröbner basis instead. The difference between the Gröbner basis method and the algorithm described in this manuscript is discussed in Section 2.4.8. The R basis, which is found from Theorem 2.3 is also unique, since changing the order of the divisors (rows) will not affect the linear dependence of the columns in Algorithm 2.1.

2.4.5 The Geometry of Polynomial Division

Having discussed the divisor matrix D and a canonical basis R for the quotient space, it is now possible to interpret multivariate polynomial division geometrically. Since

$$p = \sum_{i=1}^{s} h_i f_i + r,$$

with $\sum_{i=1}^{s} h_i f_i \in \mathcal{D}$ and $r \in \mathcal{R}$, finding the $\sum h_i f_i$ terms is then equivalent to projecting p along $\mathcal{R}$ onto $\mathcal{D}$. The remainder r can then simply be found as $p - \sum_{i=1}^{s} h_i f_i$ or as the projection of p along $\mathcal{D}$ onto $\mathcal{R}$. Figure 2.2 represents this in three-dimensional Euclidean space. The whole three-dimensional Euclidean space represents $\mathcal{C}_d^n$, the plane represents $\mathcal{D}$ and the long oblique line pointing to the left represents $\mathcal{R}$. Since $\mathcal{R}$ does not lie in $\mathcal{D}$ it is clear that $\mathcal{D} \oplus \mathcal{R} = \mathcal{C}_d^n$. The oblique projection of p along $\mathcal{R}$ onto $\mathcal{D}$ is given by the following expression

$$\sum_{i=1}^{s} h_i f_i \;=\; p\,(I - R^T R)\,\left[D(I - R^T R)\right]^{\dagger}\, D. \tag{2.11}$$

More information on projector operators can be found in Appendix A Section A.12. Expression (2.11) assumes that the basis for the vector spaces $\mathcal{D}$ and $\mathcal{R}$ are given by the rows of D and R respectively and that R is orthogonal.

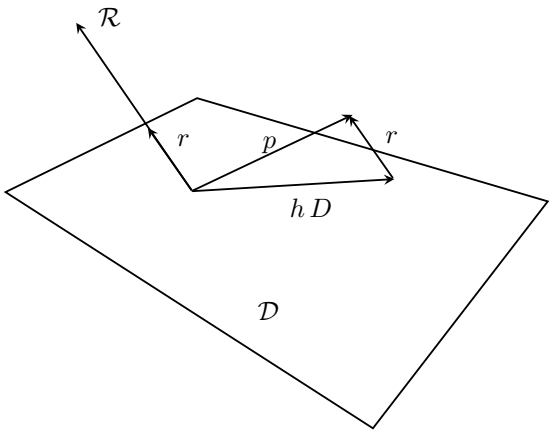

Figure 2.2: **The quotient hD of dividing p by $F = \{f_1, \ldots, f_s\}$ is found by projecting p along $\mathcal{R}$ onto $\mathcal{D}$.**

2.4.6 Algorithm & Numerical Implementation

In this section, a high-level algorithm and numerical implementation are presented for doing multivariate polynomial division. The outline of the algorithm is given in Algorithm 2.2. This is a high-level description since implementation details are not given. The most important object in the algorithm is the divisor matrix D. From this matrix a basis for $\mathcal{D}$ and $\mathcal{R}$ are determined. The $\sum_i^s h_i f_i$ terms are then found from projecting p along $\mathcal{R}$ onto $\mathcal{D}$. The remainder is then found as $r = p - \sum_i^s h_i f_i$. The quotients h_i can easily be retrieved from solving the linear system $hD = \sum_i^s h_i f_i$.

Algorithm 2.2 *Multivariate Polynomial Division*
***Input**: polynomials $f_1, \ldots, f_s, p \in \mathcal{C}_d^n$*
***Output**: $h_1, \ldots, h_s, r$ such that $p = \sum_i^s h_i f_i + r$*
$D \leftarrow$ *Divisor matrix of* $f_1, \ldots, f_s$
$A \leftarrow$ *basis of vector space* $\mathcal{D}$ *determined from* D
$B \leftarrow$ *monomial basis of vector space of remainders* $\mathcal{R}$ *determined from* D
$\sum_i^s h_i f_i \leftarrow$ *project* p *along* $\mathcal{R}$ *onto* $\mathcal{D}$
$r \leftarrow p - \sum_i^s h_i f_i$
$h = \big(h_1, \ldots, h_s\big) \leftarrow$ *solve* $hD = \sum_i^s h_i f_i$

Our implementation of Algorithm 2.2 consists of three main steps:

1. the rank of D is determined, together with a basis for $\mathcal{D}$,
2. the R basis is determined,
3. the oblique projection is computed.

The first step, the rank determination, can be done by either an SVD or a rank-revealing QR decomposition. When a sparse matrix data structure is used however, only the rank-revealing QR is an option. Our numerical implementation of steps 2 and 3 is done via 3 QR decompositions. Furthermore, the use of orthogonal matrix factorizations guarantees the numerical backward stability of the implementation. The third QR decomposition will dominate the cost of the method, which is $O((q+1)q^2)$ where q is the number of columns of D. Also observe that q grows as $O(d^n)$ where $d = \deg(p)$ and n is the number of variables. In addition, D also typically has a large amount of zero elements. Using a sparse matrix data structure is therefore a natural choice. We will discuss the implementation in detail for the sparse case, which uses the multifrontal multithreaded rank-revealing QR decomposition [24]. This sparse QR decomposition uses by default a numerical tolerance of $\tau = 20\ (q+s)\ \epsilon\ \max_j ||D_{*j}||_2$ where ϵ is the machine roundoff (about 10^{-16} since only a double-precision implementation of the sparse QR factorization is available). $\max_j ||D_{*j}||_2$ is the largest 2-norm of any row of D and D is s-by-q. The rank of D, a basis for its row space $\mathcal{D}$ and a basis for its null space K are determined from the following QR factorization

$$D^T\ P_d\ =\ Q_d\ R_d,$$

where P_d corresponds with a column permutation that reduces fill-in and allows to determine the rank. The estimate for the rank r is given by the number of nonzero diagonal elements of R_d. The r leftmost columns of $D^T\ P_d$ span $\mathcal{D}$. An orthogonal basis for the null space K is given by the remaining columns of Q_d. This first QR decomposition is a critical step in the implementation. Indeed, an ill-defined numerical rank indicates an inherent difficulty to determine the dimensions of $\mathcal{D}$ and $\mathcal{R}$. Now, Corollary (2.3) is used to find the R basis. K is per definition of full column-rank, say $\dim(K) = c$, and from a second sparse QR decomposition

$$K^T\ P_k\ =\ Q_k\ R_k$$

the linearly independent rows of K are found as the leftmost c columns of $K^T\ P_k$. In fact, the factors Q_k and R_k do not need to be kept. The column permutation will work from the leftmost column of K^T to the right, which corresponds with checking the rows of K for linear independence from top to bottom. Corollary 2.3 then ensures a correct R basis for multivariate polynomial division is found. From this, a canonical basis R for $\mathcal{R}$ can be constructed. With the first two steps completed one can now use (2.11) to find the projection of p onto $\mathcal{D}$ along $\mathcal{R}$. It is possible to simplify (2.11). The first two factors of its right-hand side are

$$p\,(I - R^T\,R) \tag{2.12}$$

and the third factor is

$$\left[D\,(I - R^T\,R)\right]^\dagger. \tag{2.13}$$

The first step in simplifying (2.11) is the calculation of the LQ factorization of

$$\begin{pmatrix} R \\ D \\ p \end{pmatrix} \;=\; L\,Q \;=\; \begin{pmatrix} L_R \\ L_D \\ L_p \end{pmatrix}\; Q, \tag{2.14}$$

where L is lower triangular and Q orthogonal. This allows us to write

$$\begin{aligned} R &= L_R\,Q, \\ D &= L_D\,Q, \\ p &= L_p\,Q. \end{aligned}$$

Since R is a canonical basis, all rows of R are orthonormal and will be contained in Q without any change. Hence, L_R will always be a unit matrix embedded into a rectangular structure

$$L_R \;=\; \begin{pmatrix} I_c & O \end{pmatrix}$$

where $c = \dim(\mathcal{R})$. This implies that $L_R\,L_R^T = I_c$. The next step is to replace p and R in (2.12) by their respective LQ decompositions

$$\begin{aligned} p\,(I_q - R^T\,R) &= L_p\,Q\,(I_q - Q^T\,L_R^T\,L_R\,Q), \\ &= L_p\,Q\,Q^T\,(I_q - L_R^T\,L_R)\,Q, \\ &= L_p\,(I_q - L_R^T\,L_R)\,Q. \end{aligned} \tag{2.15}$$

The simplifications are possible since $L_R\,L_R^T = I_c$ and $Q\,Q^T = I_q$. Applying the same strategy of replacing $\mathcal{D}$ and R by their respective LQ decompositions in (2.13) results in

$$D\,(I - R^T\,R) \;=\; L_D\,(I_q - L_R^T\,L_R)\,Q. \tag{2.16}$$

From here on, W denotes the common factor $(I_q - L_R^T\,L_R)$. Using (2.15) and (2.16) in (2.11) we obtain

$$\begin{aligned} \sum_{i=1}^{s} h_i f_i &= p\,(I - R^T R)\,\left[D(I - R^T R)\right]^\dagger\,D, \\ &= L_p\,W\,Q\,(L_D W Q)^\dagger\,D, \\ &= L_p\,W\,Q\,Q^\dagger\,(L_D W)^\dagger\,D, \\ &= L_p\,W\,(L_D W)^\dagger\,D, \end{aligned} \tag{2.17}$$

which requires the calculation of only 1 matrix pseudo-inverse. Exploiting the structure of W allows to further simplify this expression. Since $W = (I_q - L_R^T L_R)$ and L_R is a unit matrix embedded in a rectangular structure it follows that

$$W = \begin{pmatrix} 0 & 0 \\ 0 & I_r \end{pmatrix},$$

where $r = q - c$ is the rank of D. Partitioning L_p into

$$L_p = \begin{pmatrix} L_{p_1} & L_{p_2} \end{pmatrix},$$

where L_{p_2} are the r rightmost columns and likewise L_D into

$$L_D = \begin{pmatrix} L_{D_1} & L_{D_2} \end{pmatrix}$$

simplifies (2.17) to $L_{p_2} L_{D_2}^{\dagger} D$. We can therefore write the oblique projection of p along $\mathcal{R}$ on $\mathcal{D}$ as

$$\sum_{i=1}^{s} h_i f_i = L_{p_2} L_{D_2}^{\dagger} D. \tag{2.18}$$

In this final expression the orthogonal matrix Q of (2.14) does not appear and it is therefore not necessary to calculate it explicitly. L_{D_2} is a square matrix and when it is of full column rank, $L_{D_2}^{\dagger} = L_{D_2}^{-1}$. The factor $L_{p_2} L_{D_2}^{\dagger}$ in (2.18) specifies the linear combination of rows of D and therefore the decomposition of p into the $\sum_{i=1}^{s} h_i f_i$ terms. The remainder is then easily found as $r = p - \sum_{i=1}^{s} h_i f_i$.

2.4.7 Numerical Experiments

In this example we will replace x_1, x_2, x_3 with x, y, z respectively and divide the polynomial $p = -5 + 2x + y^2 + z^2 + 8xy^2$ by

$$F = \begin{cases} f_1 = -4 + x^2 + y^2 + z^2, \\ f_2 = -5 + x^2 + 2y^2, \\ f_3 = -1 + xz. \end{cases}$$

The leading monomial of p according to the degree negative lexicographic ordering is xy^2. The divisor matrix D is the following 5 by 20 matrix

$$D = \begin{pmatrix} f_1 \\ f_2 \\ xf_2 \\ f_3 \\ xf_3 \end{pmatrix}$$

The numerical tolerance for this example is $\tau = 5.468 \times 10^{-13}$. The rank is estimated to be 5 and therefore $\dim(\mathcal{R}) = 15$. The monomial basis for $\mathcal{R}$ is

$$\{1, x, y, z, x^2, xy, yz, x^3, x^2y, xyz, xz^2, y^3, y^2yz^2, z^3\}.$$

The factor $L_{p2}\,L_{D2}^{\dagger}$ equals $\begin{pmatrix}1.0 & 0 & 4.0 & 0 & 0\end{pmatrix}$ and $\sum_i h_i f_i$ is therefore

$$\sum_i h_i f_i \;=\; 1.0\,f_1 + 4.0\,x\,f_2 \;=\; -4.0 - 20.0\,x + 1.0\,x^2 + 1.0\,y^2 + 1.0\,z^2 + 4.0\,x^3 + 8.0\,xy^2.$$

The remainder term r is easily found from the vector difference

$$r \;=\; p - \sum_i h_i f_i = -1.0 + 22.0\,x - 1.0\,x^2 + 0.0\,y^2 + 0.0\,z^2 - 4.0\,x^3.$$

The total running time for computing both the quotients and the remainder was 0.011 seconds. The forward errors for both the $\sum_i h_i f_i$ terms and r are bounded from above by 10^{-15}. Observe that, unlike in the univariate case, the leading monomial of the remainder is x^3 and has a larger degree than any of the divisors.

We now perturb the coefficients of the divisors with noise of order 10^{-6} and divide $p = -5 + 2x + y^2 + z^2 + 8xy^2$ by

$$\tilde{F} \;=\; \begin{cases} f_1 &=& -4.000001 + 0.000001\,y + x^2 + y^2 + z^2, \\ f_2 &=& -5.000001 + x^2 + 2y^2, \\ f_3 &=& -1 + 0.000001\,x^2 + xz. \end{cases}$$

The noise introduced two extra terms: $10^{-6}\,y$ in f_1 and $10^{-6}\,x^2$ in f_3. The numerical rank of D remains 5 and the factor $L_{p2}\;L_{D2}^{\dagger}$ also does not change. The remainder term however now becomes

$$r \;=\; -1.0 + 22.000004\,x - 10^{-6}\,y - 1.0\,x^2 + 0.0\,y^2 + 0.0\,z^2 - 4.0x^3.$$

The noisy $10^{-6}\,y$ term ends up in the remainder and the coefficient of x is now also perturbed. Again, the forward errors are bounded from above by 10^{-15}. The total running time was 0.013 seconds.

2.4.8 Division by a Gröbner Basis

In this section, we discuss the difference between the results of our division algorithm and the division of a multivariate polynomial by a Gröbner basis. One attractive feature of a Gröbner basis is that the remainder will be independent on the ordering of the divisors. The R basis, found using Theorem 2.3 is not necessarily the R basis R_G, obtained when dividing by the corresponding Gröbner basis. This difference is due to the defining property of the Gröbner basis. Not all leading terms of the polynomial ideal $I = \langle f_1, \ldots, f_s \rangle$ (Appendix B Section B.3) are necessarily divisible by at least one of the polynomials $f_1, \ldots, f_s$ and this implies that $R_G \subseteq R$.

Example 2.10 *We revisit the unperturbed polynomial system of Section 2.4.7 and first compute the Gröbner basis G of $I = \langle f_1, f_2, f_3 \rangle$ using Maple:*

$$G = \begin{cases} g_1 &= -1 + xz, \\ g_2 &= -5 + x^2 + 2y^2, \\ g_3 &= -3 + x^2 + 2z^2, \\ g_4 &= 2\,z - 3\,x + x^3. \end{cases}$$

G contains four polynomials whereas F only three. Applying Algorithm 2.1 for G results in the following R basis $R_G = \{1, x, y, z, x^2, xy, yz, x^2y\}$, which is indeed a subset of R. Since the difference between R and R_G lies in the higher degrees, the remainder r_G from dividing p by G contains fewer terms of higher total degree,

$$r_G = -1.0 + 10.0\,x + 8.0\,z - 1.0\,x^2 + 0.0\,x^3 + 0.0\,xy^2.$$

For this example, the x^3 term of r does not appear in r_G. Computation of this remainder r_G took 0.012 seconds in MATLAB.

2.4.9 Buchberger's Algorithm

Buchberger's Algorithm (Algorithm B.1 in Appendix B Section B.5) can be summarized as constructing S-polynomials and computing remainders. One way of computing these remainders would be to use Algorithm 2.2, but there is an alternative way that does not require any projections. From Theorem 2.3 we know that a basis for $\mathcal{R}$ can be found by checking the columns of D for linear independence from right to left. Instead of using Algorithm 2.1, one can apply Gauss-Jordan elimination, without column pivoting, onto D. Indeed, the resulting pivot columns from the upper triangular result R would then correspond with the linearly independent monomials and the other monomials therefore span $\mathcal{R}$. It is important however that the Gauss-Jordan algorithm also needs to be applied from right to left. Hence, adding p to R as an extra row and applying Gauss-Jordan again from right to left leads to the reduction of p. We will denote this reduced polynomial by p'. It is now easy to see that p' is in fact the remainder of p by division of F. Either $p \in \text{row}(D)$ and $p' = 0$ or $p \notin \text{row}(D)$ and p' is then a linear combination of monomials, none of which is divisible by any of the leading monomials $\text{LM}(f_1), \ldots, \text{LM}(f_s)$, due to the resulting reduced row echelon form. Observe that applying Gauss-Jordan elimination to the concatenation of D with p is sufficient to compute the remainder. One then needs to keep track of p during the reduction.

Example 2.11 *We revisit the unperturbed example from Section 2.4.7. The concatenation of the divisor matrix D with p is*

$$\begin{array}{c} \begin{array}{cccccccccccccc} 1 & x_1 & x_2 & x_3 & x_1^2 & x_1x_2 & x_1x_3 & x_2^2 & x_2x_3 & x_3^2 & x_1^3 & x_1^2x_2 & x_1^2x_3 & x_1x_2^2 \end{array} \\ \left(\begin{array}{cccccccccccccc} -4 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ -5 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 \\ -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ -5 & 2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 8 \end{array}\right). \end{array}$$

Applying Gauss-Jordan elimination from right to left results in the matrix

$$\begin{array}{c} \begin{array}{cccccccccccccc} 1 & x_1 & x_2 & x_3 & x_1^2 & x_1x_2 & x_1x_3 & x_2^2 & x_2x_3 & x_3^2 & x_1^3 & x_1^2x_2 & x_1^2x_3 & x_1x_2^2 \end{array} \\ \left(\begin{array}{cccccccccccccc} -1 & 2 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & -22 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ -5 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right), \end{array}$$

where the 3rd row corresponds with p'. Indeed, this row corresponds up to a scalar with the polynomial

$$r \;=\; -4\,x_1^3 - x_1^2 + 22\,x_1 - 1,$$

which is the remainder as found by Algorithm 2.2.

This means that the main operation of the Buchberger algorithm, computing remainders, is the reduction of a matrix to its reduced row echelon form. This insight, that a Gröbner basis can be computed from the reduction of a matrix is due to Lazard [56], and led to Faugère's F4 and F5 algorithms [37,38]. These algorithms are multiprecision integer implementations of Buchberger's Algorithm, where the remainders are computed using row echelon forms. In addition, remainders can be computed independently of one another and hence the whole algorithm is parallelized. F5 was the first algorithm to solve the previously intractable "cyclic 10" problem [38]. The link between the reduction of a matrix and Gröbner bases will be further explored in Chapter 5, where we introduce the canonical decomposition of $\mathcal{C}_d^n$, and Chapter 6, where this decomposition is related to the numerical computation of a Gröbner basis.

Chapter 3

Macaulay matrix

In this chapter, we introduce and discuss the most important matrix of this thesis: the Macaulay matrix. This matrix plays a central role in all subsequent chapters. Expressions for its size are given and an expression for its density is derived. Important interpretations of its row space, left null space and null space are also derived. These interpretations will be crucial for the development of the algorithms in Chapter 6. Finally, we also derive an upper bound for the largest singular value of the Macaulay matrix. We will use this bound to quantify the change of the singular values of the Macaulay matrix due to perturbations.

3.1 Definition

Simply stated, the Macaulay matrix is a generalization of the multiplication matrix. Whereas the multiplication matrix $M_{f_1}(d)$ is defined by only one polynomial f_1, the Macaulay matrix is defined by a system of multivariate polynomials $f_1, \ldots, f_s$. The reason this matrix is called the Macaulay matrix is because it was Macaulay who introduced this matrix, drawing from earlier work by Sylvester [85], in his work on elimination theory, resultants and solving multivariate polynomial systems [60,61]. We start this chapter by giving its formal definition.

Definition 3.1 *Given a set of polynomials $f_1, \ldots, f_s \in \mathcal{C}^n$, each of degree d_i $(i = 1, \ldots, s)$, then the Macaulay matrix of degree $d \geq max(d_1, \ldots, d_s)$ is the matrix containing the coefficients of*

$$M(d) = \begin{pmatrix} f_1 \\ x_1 f_1 \\ \vdots \\ x_n^{d-d_1} f_1 \\ f_2 \\ x_1 f_2 \\ \vdots \\ x_n^{d-d_s} f_s \end{pmatrix} \tag{3.1}$$

where each polynomial f_i is multiplied with all n-variate monomials from degree 0 up to $d - d_i$ for all $i = 1, \ldots, s$.

Notice again the similarity with the multiplication matrix. Indeed, the Macaulay matrix is nothing but the multiplication matrices $M_{f_i}(d)$ for each polynomial f_i of the polynomial system, all stacked onto one another. As with the multiplication matrix, the dependence of this matrix on the degree d for which it is defined is expressed explicitly in the notation $M(d)$. When constructing the Macaulay matrix, it is more practical to start with the coefficient vectors of the original polynomial system $f_1, \ldots, f_s$, after which all the rows corresponding to multiplied polynomials $x^a f_i$ up to a degree $\max(d_1, \ldots, d_s)$ are added. Then one can add the coefficient vectors of all polynomials $x^a f_i$ of one degree higher and so forth until the desired degree d is obtained. This is illustrated in the following example.

Example 3.1 *For the following polynomial system in $\mathcal{C}_2^2$*

$$\begin{cases} f_1: & x_1x_2 - 2x_2 &= 0, \\ f_2: & x_2 - 3 &= 0, \end{cases}$$

we have that $\max(d_1, d_2) = 2$ and we want to construct $M(3)$. The first 2 rows then correspond with the coefficient vectors of f_1, f_2. Since $\max(d_1, d_2) = 2$ and $d_2 = 1$, the next 2 rows correspond to the coefficient vectors of $x_1 f_2$ and $x_2 f_2$ of degree 2. Notice that these first 4 rows make up $M(2)$ when the columns are limited to all monomials of degree 0 up to 2. The next rows that are added are the coefficient vectors of $x_1 f_1, x_2 f_1$ and $x_1^2 f_2, x_1 x_2 f_2, x_2^2 f_2$ which are all polynomials

of degree 3. This way of constructing the Macaulay matrix $M(3)$ then results in

$$M(3) \;=\; \begin{array}{c} \\ f_1 \\ f_2 \\ x_1\,f_2 \\ x_2\,f_2 \\ x_1 f_1 \\ x_2 f_1 \\ x_1^2\,f_2 \\ x_1x_2\,f_2 \\ x_2^2\,f_2 \end{array} \begin{array}{c} \begin{array}{cccccccccc} 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 & x_1^3 & x_1^2x_2 & x_1x_2^2 & x_2^3 \end{array} \\ \begin{pmatrix} 0 & 0 & -2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -3 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}.$$

3.2 Size, number of nonzero coefficients and density

For a given degree d, the number of rows $p(d)$ of $M(d)$ is given by the polynomial

$$p(d) \;=\; \sum_{i=1}^{s} \binom{d-d_i+n}{n} \;=\; \frac{s}{n!}\,d^n + O(d^{n-1}), \tag{3.2}$$

and the number of columns $q(d)$ by

$$q(d) \;=\; \binom{d+n}{n} \;=\; \frac{1}{n!}\,d^n + O(d^{n-1}). \tag{3.3}$$

From these two expressions it is clear that the number of rows will grow faster than the number of columns as soon as $s > 1$.

Example 3.2 *Suppose we have a multivariate polynomial system $F = \{f_1, \ldots, f_5\}$ with $n = 10$ and respective degrees $d_1 = 1, d_2 = 2, d_3 = 3, d_4 = 4, d_5 = 5$. The number of rows of the Macaulay matrix of F at degree d is then given by the polynomial*

$$p(d) \;=\; \frac{1}{24}\,d^5 + \frac{5}{8}\,d^3 + \frac{1}{3}\,d,$$

and the number of columns by the polynomial

$$q(d) \;=\; \frac{1}{120}\,d^5 + \frac{1}{8}\,d^4 + \frac{17}{24}\,d^3 + \frac{15}{8}\,d^2 + \frac{137}{60}\,d + 1.$$

We denote the rank of $M(d)$ by $r(d)$ and the dimension of its left and right null space by $l(d)$ and $c(d)$ respectively. The rank-nullity theorems of $M(d)^T$ and $M(d)$

can then be expressed as

$$\begin{aligned} q(d) &= r(d) + c(d), \\ p(d) &= r(d) + l(d), \end{aligned}$$

which shows that $r(d), l(d), c(d)$ also are polynomials over all positive integers $d > \max(d_1, \ldots, d_s)$. This polynomial increase of the size of $M(d)$, due to the combinatorial explosion of the number of monomials, is the main bottleneck when solving practical problems but is unfortunately inherent to multivariate polynomials. Fortunately, as we will now show, the Macaulay matrix is very sparse. We will exploit this sparsity in the recursive orthogonalization algorithm of Chapter 4.

The density of a matrix is defined as the ratio of the total number of nonzero elements to the total number of elements. Let $n_1, \ldots, n_s$ be the total number of nonzero coefficients of the polynomials $f_1, \ldots, f_s$ respectively. It is then easily derived that the total number of nonzero elements of $M(d)$ is given by the polynomial

$$\sum_{i=1}^{s} n_i \binom{d - d_i + n}{n} = \frac{(n_1 + \ldots + n_s)}{n!} d^n + O(d^{n-1}).$$

An interesting question is how much storage is needed to store $M(d)$ in memory. Assuming that each number requires 8 Bytes to store, since we are working in double precision, then storing all elements takes

$$p(d) \times q(d) \times 8 \text{ Bytes.}$$

Using the compressed column data structure (Appendix A Section A.13), it is necessary to store the row indices Ai, the nonzero values Ax and the column indices Ap. Assuming there are no zero columns, this is approximately

$$\left(2 \sum_{i=1}^{s} n_i \binom{d - d_i + n}{n} + \binom{d+n}{n} \right) \times 8 \text{ Bytes.}$$

The gain in memory when storing $M(d)$ using the compressed column data structure is therefore approximately

$$\frac{p(d)\, q(d)\, 8}{\left(2 \sum_{i=1}^{s} n_i \binom{d-d_i+n}{n} + \binom{d+n}{n} \right) 8} \approx \frac{s\, d^n}{(2n_1 + \ldots + 2n_s + 1)\, n!}.$$

Remark 3.1 *No use was made in the calculation of the required memory that the coefficients of the polynomials $f_1, \ldots, f_s$ are repeated in the Macaulay matrix. An interesting line of future research would be to further compress the data structure such that each nonzero coefficient needs to be stored only once. This would result in an additional gain of required memory.*

In Chapter 4 some gains in required storage will be reported for particular examples. These will be in the range of 13 up to 500.

Example 3.3 *Suppose* $n = 10, s = 10$ *and that* $d_1 = \ldots = d_{10} = 4$*, then Table 3.1 lists the gain in required memory in Bytes for* d *from* 4 *up to* 10*. The rapid* $d^n/n!$ *growth is already visible from this table. For* $d = 16$*, another increment of the degree by 6, the gain is approximately* 2.652×10^3*.*

Table 3.1: gain in required memory [Bytes]

d	4	5	6	7	8	9	10
gain	0.4762	1.4798	3.9759	9.6814	21.8095	46.0721	92.1795

Using the above polynomial expressions, the density $\rho(d)$ of $M(d)$ is written as

$$\rho(d) \;=\; \frac{\sum_{i=1}^{s} n_i \binom{d-d_i+n}{n}}{\sum_{i=1}^{s} \binom{d-d_i+n}{n}\binom{d+n}{n}}. \tag{3.4}$$

An approximation of the density can be found from the dominant term of (3.4) for large d. By substituting the polynomials of the numerator and denominator of (3.4) by their highest order terms we get

$$\rho(d) \approx \frac{\frac{(n_1+\ldots+n_s)\,d^n}{n!}}{\frac{s\,d^n}{n!}\,\frac{d^n}{n!}} = \frac{(n_1+\ldots+n_s)\,n!}{s\,d^n}.$$

Hence, the density of $M(d)$ is inverse proportional to d^n for large d. The sparsity of $M(d)$ is defined naturally as $1-\rho(d)$. Already for moderate values of d and n, d^n can grow quite fast so that the density will decrease accordingly and hence $M(d)$ will quickly become sparse.

Example 3.4 *Suppose* $n = 10, s = 10, d_1 = \ldots = d_{10} = 10$*. Assume further that the polynomial system is very dense: every possible monomial has a nonzero coefficient. Obviously, for this case* $\rho(10) = 1$*. Figure 3.1 displays the density* $\rho(d)$ *in* % *as a function of the degree* d*. Notice how the density has already dropped to* 1% *for* $d \approx 19$*. Hence, for this degree we have that* $M(19)$ *consists approximately for* 99% *out of zero entries.*

3.3 Upper bound on largest singular value

The Macaulay matrix is, up to a row permutation, a concatenation of multiplication matrices. Just like for the multiplication matrix it is possible to express an

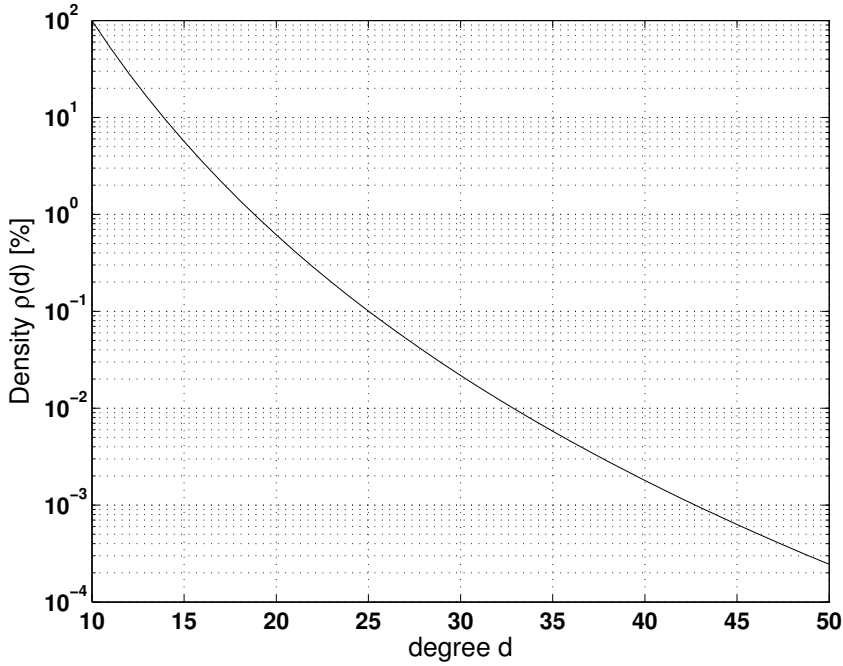

Figure 3.1: **Density of the polynomial system of Example 3.4 as a function of the degree d.**

upper bound for the largest singular value and thus bound $||\sum_i h_i f_i||_2$. Another application of this bound is in that it can be used to bound the change of the singular values of the Macaulay matrix when the coefficients of $f_1, \ldots, f_s$ are perturbed. The first step to derive the bound on the largest singular value is a trivial generalization of Lemma 2.1, which bounds the absolute column sum of the Macaulay matrix.

Lemma 3.1 *Let $f_1, \ldots, f_s \in \mathcal{C}^n$, each of degree d_i $(i = 1, \ldots, s)$, with*

- $d_0 = max(d_1, \ldots, d_s)$,
- $d_{m_i} = \min\limits_{a_\alpha \neq 0} deg(a_\alpha x^\alpha)$ *(for each f_i, $1 \leq i \leq s$),*
- d_l *the total degree of the LCM of all monomials of $f_1, \ldots, f_s$ with nonzero coefficients,*
- $d^\star = \max\limits_{1 \leq i \leq s} d_0 + (d_l - d_{m_i}) - (d_0 - d_i)$

and $M(d)$ the corresponding Macaulay matrix. Then

$$\max_{1\leq j\leq q} c_j \;\leq\; \sum_{i=1}^{s} ||f_i||_1,$$

where equality is guaranteed from $d \geq d^\star$.

PROOF. The proof is a generalization of the proof in Lemma 2.1. Since $M(d)$ is a concatenation of multiplication matrices, the maximal absolute column sum c_j is obviously the sum of the 1-norms of all the polynomials $f_1, \ldots, f_s$. This maximal absolute column sum will happen when all nonzero monomials of the polynomial system are shifted to their least common multiple. At $d = d_0$, we have that for each polynomial f_i the monomial of minimal total degree d_{m_i} is already multiplied with a degree $d_0 - d_i$. Since each monomial needs to be multiplied until it has degree d_l, the degree for which this happens is $d^\star = \max_{1\leq i\leq s} d_0 + (d_l - d_{m_i}) - (d_0 - d_i)$. □

Lemma 3.1 immediately results in the following theorem that bounds the largest singular value of the Macaulay matrix.

Theorem 3.1 *Let $f_1, \ldots, f_s \in \mathcal{C}^n$ and $M(d)$ the corresponding Macaulay matrix. Then its largest singular value σ_1 is bounded from above by*

$$\sigma_1 \leq \sqrt{\sum_{j=1}^{s} ||f_j||_1 \max_{1\leq i\leq s} ||f_i||_1}. \tag{3.5}$$

PROOF. Like for the multiplication matrix, the starting point is the upper bound by Schur [74]

$$\sigma_1^2 \leq \max_{1\leq i\leq p, 1\leq j\leq q} r_i\, c_j.$$

The maximal absolute row sum r_i will be obviously $\max_{1\leq i\leq s} ||f_i||_1$. Lemma 3.1 ensures that the maximal c_j is $\sum_{i=1}^{s} ||f_i||_1$. From this, the theorem follows. □

Example 3.5 *For the polynomial system*

$$\left\{\begin{array}{lrcl} f_1: & 2\,x_4^2 + 2\,x_3^2 + 2\,x_2^2 + 2\,x^2 - x_1 &=& 0,\\ f_2: & 2\,x_3\,x_4 + 2\,x_2\,x_3 + 2\,x_1\,x_2 - x_2 &=& 0,\\ f_3: & 2\,x_2\,x_4 + 2\,x_1\,x_3 + x_2^2 - x_3 &=& 0,\\ f_4: & 2\,x_4 + 2\,x_3 + 2\,x_2 + x_1 - 1 &=& 0, \end{array}\right.$$

both the largest singular value σ_1 and the upper bound from Theorem 3.1 are displayed in Figure 3.2 for degree 2 up to 19. Since $\max_{1 \leq i \leq s} ||f_i||_1 = ||f_1||_1 = 9$ *and* $\sum_i ||f_i||_1 = 30$*, the upper bound is for all degrees* $\sqrt{30 \times 9} = 16.432$*. Notice that the upper bound gets better for increasing degrees. For this polynomial system we also have that* $d^\star = 9$*.*

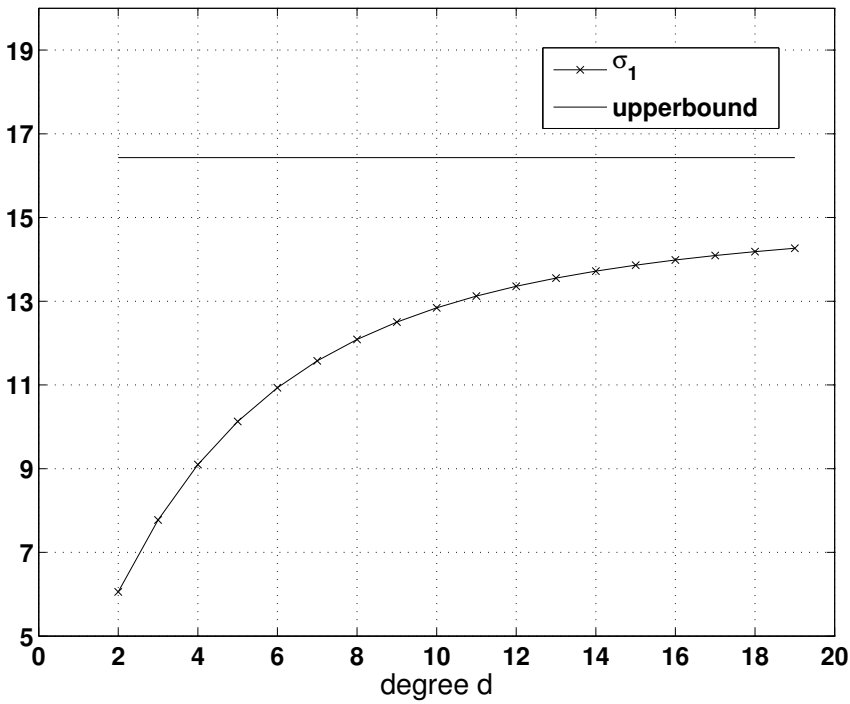

Figure 3.2: **Largest singular value σ_1 of $M(d)$ in Example 3.5 as a function of d.**

Just like for the multiplication matrix, this upper bound on the largest singular value bounds $||\sum_i h_i f_i||_2$.

Corollary 3.1 *Let h be the concatenation of all coefficient vectors of $h_1, \ldots, h_s$ into one row vector, and f the concatenation of all coefficient vectors of $f_1, \ldots, f_s$ into one row vector, then*

$$||\sum_{i=1}^{s} h_i f_i||_2 \leq min \left(||h||_2 \sqrt{\sum_{j=1}^{s} ||f_j||_1 \max_{1 \leq i \leq s} ||f_i||_1}, ||f||_2 \sqrt{\sum_{j=1}^{s} ||h_j||_1 \max_{1 \leq i \leq s} ||h_i||_1} \right).$$

Remark 3.2 *Lemma 3.1, Theorem 3.1 and Corollary 3.1 are indeed generalizations of respectively Lemma 2.1, Theorem 2.1 and Corollary 2.1 for the Macaulay matrix. When* $s = 1$ *and* $h_1 = f_2$*, then*

- *Lemma 3.1 becomes Lemma 2.1,*
- *Theorem 3.1 becomes Theorem 2.1 and*
- *Corollary 3.1 becomes Corollary 2.1.*

Another application of Theorem 3.1 is in bounding the perturbation of the singular values of the Macaulay matrix when the coefficients of $f_1, \ldots, f_s$ are subject to noise. Suppose the coefficients of the polynomial system $f_1, \ldots, f_s$ are corrupted by noise in the following way

$$\begin{aligned} \tilde{f}_1 &= f_1 + e_1, \\ \tilde{f}_2 &= f_2 + e_2, \\ &\vdots \\ \tilde{f}_s &= f_s + e_s, \end{aligned}$$

where

$$||e_i||_1 \leq \epsilon \ (i = 1, \ldots, s).$$

The Macaulay matrix $\tilde{M}(d)$ of $\tilde{f}_1, \ldots, \tilde{f}_s$ is then related to $M(d)$ as

$$\tilde{M}(d) \ = \ M(d) + E(d). \tag{3.6}$$

The matrix $E(d)$ is in fact a Macaulay matrix of the polynomial system that consists only of the noise polynomials $e_1, \ldots, e_s$. We can then prove the following theorem that bounds the perturbation of the singular values of $M(d)$.

Theorem 3.2 *Let* $\tilde{f}_1, \ldots, \tilde{f}_s$ *be the polynomial system after perturbing* $f_1, \ldots, f_s$*, with each of the noise polynomials bounded by* $||e_i||_1 \leq \epsilon \ (i = 1, \ldots, s)$*. Let* $\tilde{M}(d)$ *be the perturbed Macaulay matrix, with singular values* $\tilde{\sigma}$*, and* $M(d)$ *the original Macaulay matrix, with singular values* σ*. Then*

$$|\tilde{\sigma}_j - \sigma_j| \leq \epsilon \sqrt{s}$$

with $j = 1, \ldots, min(p(d), q(d))$.

PROOF. Weyl [80, 88] proved the following bound on the absolute difference between the singular values

$$|\tilde{\sigma}_j - \sigma_j| \leq ||E||_2.$$

We apply Theorem 3.1 to bound $||E||_2$

$$||E||_2 \leq \sqrt{\sum_{i=1}^{s} ||e_i||_1 \max_{1 \leq i \leq s} ||e_1||_1}.$$

Obviously $\sum_{i=1}^{s} ||e_i||_1$ is bounded by $s\,\epsilon$ and $\max_{1 \leq i \leq s} ||e_1||_1$ is bounded by ϵ. We can therefore write

$$||E||_2 \leq \sqrt{s\ \epsilon\ \epsilon} = \epsilon\sqrt{s},$$

which concludes the proof. □

A nice feature of this bound is that it is independent of the size of the Macaulay matrix.

Example 3.6 *Suppose we have the following polynomial system*

$$F = \begin{cases} 2\,x_4^2 + 2\,x_3^2 + 2\,x_2^2 + 2\,x_1^2 - x_1 &= 0, \\ 2\,x_3\,x_4 + 2\,x_2\,x_3 + 2\,x_1\,x_2 - x_2 &= 0, \\ 2\,x_2\,x_4 + 2\,x_1\,x_3 + x_2^2 - x_3 &= 0, \\ 2\,x_4 + 2\,x_3 + 2\,x_2 + x_1 - 1 &= 0. \end{cases}$$

We now perturb each of the coefficients of F by noise, uniformly drawn from the interval $[-10^{-3}, 10^{-3}]$. Each of the noise polynomials $e_1, \ldots, e_4$ is bounded by 5×10^{-3}. Applying Theorem 3.2 tells us then that

$$|\tilde{\sigma}_j - \sigma_j| \leq 5 \times 10^{-3}\,\sqrt{4} = 10^{-2}.$$

For example, when $d = 10$, the maximal perturbation of the singular values is 1.4244×10^{-3}.

This bound on the perturbation of the singular values is used in [32] for the computation of approximate solutions to noisy overdetermined polynomial systems.

3.4 Row space

In this section, we will present two interpretations of the row space of the Macaulay matrix. This will naturally lead to the concept of the canonical decomposition, which we will introduce in Chapter 5, and the ideal membership problem, which we will solve in Chapter 6.

3.4.1 Affine interpretation

A first interesting observation is the affine interpretation of the row space of $M(d)$. The row space of $M(d)$, denoted by $\mathcal{M}_d$, contains all n-variate polynomials

$$\mathcal{M}_d = \left\{ \sum_{i=1}^{s} h_i f_i \ : \ h_i \in C^n_{d-d_i} \ (i = 1, \ldots, s) \right\}. \tag{3.7}$$

A polynomial ideal $\langle f_1, \ldots, f_s \rangle$ is defined as the set

$$\langle f_1, \ldots, f_s \rangle \ = \ \left\{ \sum_{i=1}^{s} h_i f_i : h_1, \ldots, h_s \in \mathcal{C}^n \right\}.$$

It is now tempting to have the following interpretation

$$\mathcal{M}_d \ = \ \langle f_1, \ldots, f_s \rangle \cap \mathcal{C}^n_d \triangleq \langle f_1, \ldots, f_s \rangle_d,$$

or in words: the row space of $M(d)$ contains all polynomials of the ideal $\langle f_1, \ldots, f_s \rangle$ from degree 0 up to d. This is not necessarily valid. $\mathcal{M}_d$ does not in general contain all polynomials of degree d that can be written as a polynomial combination (3.7).

Example 3.7 *Consider the following polynomial system [35] in* $\mathcal{C}^3_4$

$$\left\{ \begin{array}{rcrcrcrcrcl} -9 & - & x_2^2 & - & x_3^2 & - & 3\,x_2^2 x_3^2 & + & 8\,x_2 x_3 & = & 0, \\ -9 & - & x_3^2 & - & x_1^2 & - & 3\,x_1^2 x_3^2 & + & 8\,x_1 x_3 & = & 0, \\ -9 & - & x_1^2 & - & x_2^2 & - & 3\,x_1^2 x_2^2 & + & 8\,x_1 x_2 & = & 0. \end{array} \right.$$

The polynomial

$$\begin{array}{rcl} p & = & 867\,x_1^5 - 1560\,x_3\,x_2\,x_1 - 2312\,x_2^2\,x_1 + 1560\,x_3\,x_1^2 + 2104\,x_2\,x_1^2 - 1526\,x_1^3 \\ & & +4896\,x_2 - 2295\,x_1, \end{array}$$

of degree 5 is not an element of $\mathcal{M}_5$*. This can easily be verified by a rank test: append the coefficient vector of* p *to* $M(5)$ *and the rank increases by one, which means that* p *does not lie in* $\mathcal{M}_5$*. However,* $p \in \mathcal{M}_{11}$*, which implies that a polynomial combination of degree eleven is necessary in order to construct* p*. In doing so, all terms of degrees six up to eleven cancel one another.*

Hence, the reason that not all polynomials of degree d lie in $\mathcal{M}_d$ is that it is possible that a polynomial combination of a degree higher than d is required. This is due to the polynomial system having roots at infinity. The problem of determining whether a given multivariate polynomial p lies in the ideal $\langle f_1, \ldots, f_s \rangle$ generated by given polynomials $f_1, \ldots, f_s$ is called the ideal membership problem in algebraic geometry.

Problem 3.1 ***(Ideal Membership problem)*** *Let $p, f_1, \ldots, f_s \in \mathcal{C}^n$. Decide whether $p \in \langle f_1, \ldots, f_s \rangle$.*

Example 3.7 indicates that Problem 3.1 could be solved using numerical linear algebra: one could append the coefficient vector of p as an extra row to the Macaulay matrix $M(d)$ and do a rank test. As Example 3.7 also showed, it is not sufficient to do the rank test only for the degree of the given polynomial p. The algorithm requires a stop condition on the degree d for which $M(d)$ should be constructed. We can therefore restate Problem 3.1 in the following way.

Problem 3.2 *Find the degree d_I such that the ideal membership problem can be decided by checking whether*

$$rank\left(\begin{pmatrix} M(d_I) \\ p \end{pmatrix}\right) = rank\,(\,M(d_I)\,).$$

Problem 3.2 is related to finding the ideal membership degree bound. The ideal membership degree bound I_b is the least value such that for all polynomials $f_1, \ldots, f_s$ whenever $p \in \langle f_1, \ldots, f_s \rangle$ then

$$p = \sum_{i=1}^{s} h_i\, f_i \quad h_i \in \mathcal{C}^n, \deg(h_i\, f_i) \leq I_b + \deg(p).$$

Upper bounds are available on the ideal membership degree bound I_b. They are for the general case tight and doubly exponential [57, 62, 89], which renders them useless for most practical purposes. Central in this problem is the concept of a Gröbner basis. We will show in Chapter 6 Section 6.3 how Problem 3.1 can be solved numerically and how the degree d_I is related to the degree for which a Gröbner basis occurs in $\mathcal{M}_d$.

3.4.2 Projective interpretation

There is a different interpretation of the row space of $M(d)$ such that all polynomials of degree d are contained in it. This requires the notion of homogeneous polynomials and will be crucial to understand the null space of the Macaulay matrix. It will turn out that the dimension of the null space of $M(d)$ is related to the total number of projective roots of the polynomial system. This includes roots at infinity and in this way homogeneous polynomials (Appendix B Section B.2) are relevant. The vector space of homogeneous polynomials in $n+1$ variables of degree d is denoted $\mathcal{P}_d^n$. The dimension of this vector space is

$$\dim \mathcal{P}_d^n = \binom{d+n}{n} = q(d).$$

This is no coincidence, given a set of non-homogeneous polynomials $f_1, \ldots, f_s$ we can also interpret $\mathcal{M}_d$ as the vector space

$$\mathcal{M}_d \;=\; \left\{ \sum_{i=1}^{s} h_i\, f_i^h \;:\; h_i \in \mathcal{P}^n_{d-d_i}\ (i = 1, \ldots, s) \right\}, \tag{3.8}$$

where the f_i^h's are $f_1, \ldots, f_s$ homogenized and the h_i's are also homogeneous. The corresponding homogeneous ideal is denoted by $\langle f_1^h, \ldots, f_s^h \rangle$. The homogeneity ensures that the effect of higher order terms cancelling one another as in Example 3.7 does not occur. This guarantees that all homogeneous polynomials of degree d are contained in $\mathcal{M}_d$. Or in other words,

$$\mathcal{M}_d \;=\; \langle f_1^h, \ldots, f_s^h \rangle_d,$$

where $\langle f_1^h, \ldots, f_s^h \rangle_d$ are all homogeneous polynomials of degree d contained in the homogeneous ideal $\langle f_1^h, \ldots, f_s^h \rangle$. In Appendix B, Section B.4, it is shown that the homogenization of $f_1, \ldots, f_s$ typically introduces extra roots that satisfy $x_0 = 0$ and at least one $x_i \neq 0\,(i = 1, \ldots, s)$. These points are roots at infinity. We revisit Example 3.1 to illustrate this point.

Example 3.8 *The homogenization of the polynomial system in Example 3.1 is*

$$\left\{ \begin{array}{lrcl} f_1^h: & x_1x_2 - 2x_2x_0 & = & 0 \\ f_2^h: & x_2 - 3x_0 & = & 0. \end{array} \right.$$

All homogeneous polynomials $\sum_{i=1}^{2} h_i\, f_i^h$ *of degree three belong to the row space of*

$$\begin{array}{c} \\ x_0 f_1 \\ x_0^2 f_2 \\ x_0 x_1 f_2 \\ x_0 x_2 f_2 \\ x_1 f_1 \\ x_2 f_1 \\ x_1^2\, f_2 \\ x_1 x_2\, f_2 \\ x_2^2\, f_2 \end{array} \begin{array}{c} \begin{array}{cccccccccc} x_0^3 & x_1x_0^2 & x_2x_0^2 & x_1^2x_0 & x_1x_2x_0 & x_2^2x_0 & x_1^3 & x_1^2x_2 & x_1x_2^2 & x_2^3 \end{array} \\ \left(\begin{array}{cccccccccc} 0 & 0 & -2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -3 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & 1 \end{array} \right) \end{array}$$

which equals $M(3)$ *from Example 3.1. Note that the non-homogeneous polynomial system had only 1 root* $= \{(2, 3)\}$*. After homogenization, the resulting polynomial system* f_1^h, f_2^h *has 2 nontrivial roots* $= \{(1, 2, 3), (0, 1, 0)\}$.

The homogeneous interpretation is in effect nothing but a relabelling of the columns and rows of $M(d)$. The fact that all homogeneous polynomials of degree d are contained in $\mathcal{M}_d$ simplifies the ideal membership problem for a homogeneous polynomial to a single rank test.

Theorem 3.3 *Let $f_1, \ldots, f_s \in \mathcal{C}^n$ and $p \in \mathcal{P}_d^n$. Then $p \in \langle f_1^h, \ldots, f_s^h \rangle$ if and only if*

$$rank\left(\begin{pmatrix} M(d) \\ p \end{pmatrix} \right) = rank(\, M(d)\,). \tag{3.9}$$

Example 3.9 *Suppose that*

$$\begin{cases} f_1 &= x_2 - x_1^2, \\ f_2 &= x_3 - x_1^3, \end{cases}$$

and that

$$p = x_0^2 x_3 - x_0 x_1 x_2.$$

We now want to assess whether $p \in \langle x_0 x_2 - x_1^2, x_0^2 x_3 - x_1^3 \rangle$. Since deg$(p) = 3$, we need to construct the 5×20 matrix $M(3)$. The rank of $M(3)$ is five, since its five singular values are

$$1.732, 1.414, 1.414, 1.414, 1.000.$$

Recomputing the singular values after appending the coefficient vector of p to $M(3)$ results in

$$1.732, , 1.732, 1.414, 1.414, 1.414, 3.4 \times 10^{-16}.$$

Since the last singular value is numerically zero, we can conclude that the rank is also five and hence that $p \in \langle x_0 x_2 - x_1^2, x_0^2 x_3 - x_1^3 \rangle$.

3.5 Left null space

3.5.1 Syzygy analysis

The left null space of $M(d)$, null$(M(d)^T)$, is the vector space

$$\text{null}(M(d)^T) = \{h \in \mathbb{R}^{p(d)} \,|\, h\, M(d) = 0\}.$$

The vectors h are not to be interpreted as polynomials but rather as s-tuples of multivariate polynomials. Indeed, the expression $hM(d) = 0$ is equivalent with

$$\sum_{i=1}^{s} h_i\, f_i = 0. \tag{3.10}$$

The vector h therefore contains the coefficients of all polynomials h_i. A polynomial combination such as (3.10) is called a syzygy [84], from the Greek word $\sigma\upsilon\zeta\upsilon\gamma\iota\alpha$, which refers to an alignment of three celestial bodies. In our case, the polynomials h_i are thought to be in syzygy with the polynomials f_i, hence their polynomial

combination is zero. From this we then deduce that the dimension of the left null space, $l(d)$, counts the number of syzygies that occur in $\mathcal{M}_d$. It is therefore possible to identify with each syzygy a linearly dependent row of $M(d)$. The linear dependence of this particular row is then with respect to the remaining rows. Algorithm 3.1 finds a maximal set of such linearly dependent rows l for a given Macaulay matrix $M(d)$, starting from the top row r_1 where r_i stands for the ith row of the Macaulay matrix.

Algorithm 3.1 *Find a maximal set of linearly dependent rows*
Input: *Macaulay matrix* $M(d)$
Output: *a maximal set of linearly dependent rows* l

```
l ← ∅
if r_1 = 0 then
  l ← [l , r_1]
end if
for i = 2 : 1 : p(d) do
  if r_i linearly dependent with respect to {r_1, ..., r_{i-1}} then
    l ← [l , r_i]
  end if
end for
```

Note that it is possible to deduce the growth of $l(d)$. Indeed, suppose the first element l_1 of l is found in Algorithm 3.1. This row is then linearly dependent with respect to all rows above it. In fact, this linear dependence expresses a certain syzygy $\sum_{i=1}^{s} h_i f_i = 0$. The row l_1 then also corresponds with a certain monomial multiple $x_i^\alpha f_k$ since it is a row of the Macaulay matrix. Observe now that $x_j^\beta \sum_{i=1}^{s} h_i f_i = 0$, which means that all rows corresponding with $x_j^\beta x_i^\alpha f_k$ will also be linearly dependent. We will call l_1 in this case a basis syzygy and its monomial multiples $x_j^\beta l_1$, derived syzygies. The above observation can now be summarized in the following lemma.

Lemma 3.2 *If a basis syzygy* l *occurs at a degree* d_l*, then it introduces a term*

$$\binom{d - d_l + n}{n} \tag{3.11}$$

to $l(d)$*.*

Proof. This follows from $x_j^\beta \sum_{i=1}^{s} h_i f_i = 0$ and the fact that the total number of monomials x_j^β at a degree $d \geq d_l$ is given by (3.11). □

It can be shown that the number of basis syzygies is finite. This is in fact linked with the finiteness of the Gröbner basis for a polynomial ideal [21, p. 223] and leads in our PNLA framework to the following problem.

Problem 3.3 *Find the degree d_S such that all basis syzygies are found with Algorithm 3.1 for $M(d_S)$.*

Just like for the ideal membership problem, this problem also has a doubly exponential bound and is related to finding the syzygy basis bound [9, 57, 62]. The syzygy basis bound S_b is the least value such that for a polynomial system $f_1, \ldots, f_s$ each basis syzygy has a degree at most S_b. The degree of a syzygy is the maximum degree of the h_i's. Again, Gröbner bases are of crucial importance in understanding Problem 3.3. In Section 6.1 it will be shown how d_S is related to the degree for which a Gröbner basis occurs in $\mathcal{M}_d$.

We will now assume that all basis syzygies were found, using for example Algorithm 3.1 for a sufficiently large degree, and explain how all basis syzygies can be used to derive an expression for the polynomial $l(d)$. As mentioned above, each linearly dependent row can be labeled as a monomial shift of one of the polynomials $f_1, \ldots, f_s$. The first step of the syzygy analysis is to divide the basis syzygies into groups according to the polynomial that is multiplied. The following simple example illustrates this grouping of basis syzygies.

Example 3.10 *Consider the following polynomial system in C^3*

$$\left\{ \begin{array}{lrcl} f_1: & x^2\,y^2 + z & = & 0, \\ f_2: & x\,y - 1 & = & 0, \\ f_3: & x^2 + z & = & 0, \end{array} \right.$$

where $x_1 = x$, $x_2 = y$, $x_3 = z$. The first basis syzygy is found, using Algorithm 3.1, in $M(4)$ and corresponds with the row

$$xy\,f_3.$$

The remaining basis syzygies are all found in $M(6)$ and correspond with the rows

$$x^3y\,f_2,\ x^2y^2\,f_2,\ xy^2z\,f_2.$$

We can now divide these basis syzygies into the following two groups

$$\{xy\,f_3\} \quad \textit{and} \quad \{x^3y\,f_2, x^2y^2\,f_2, xy^2z\,f_2\},$$

which is one group for f_3 and one group for f_2.

The key observation here is that each of these groups can be analyzed separately since no interference between rows of different groups is possible (indeed, they involve different polynomials). We will now continue Example 3.10 and show how all contributions of basis syzygies to $l(d)$ are described by binomial coefficients.

Example 3.11 *The first group* $\{xy\, f_3\}$ *has only one element and describes a syzygy at degree 4. Lemma 3.2 tells us then that this will introduce a term*

$$\binom{d-4+3}{3}$$

to $l(d)$*. We can therefore write*

$$l(d) \;=\; \binom{d-4+3}{3} \;=\; \frac{1}{6}d^3 - d^2 + \frac{11}{6}\,d - 1.\ (d \geq 4) \tag{3.12}$$

The second group has three basis syzygies, $\{x^3y\, f_2, x^2y^2\, f_2, xy^2z\, f_2\}$*, at degree six and therefore introduces three terms*

$$\binom{d-6+3}{3}.$$

We can therefore update $l(d)$ *to*

$$l(d) \;=\; \binom{d-4+3}{3} + 3\binom{d-6+3}{3} = \frac{2}{3}d^3 - 7d^2 + \frac{76}{3}d - 31.\ (d \geq 4) \tag{3.13}$$

Expression (3.13) *for* $l(d)$ *is still valid for degrees* $d \geq 4$*, since the extra terms correspond with polynomials that have roots at* $d \in \{3, 4, 5\}$*. The difference between* (3.12) *and* (3.13) *is only visible therefore for* $d \geq 6$*. We have not yet found the final expression for* $l(d)$ *however. The three binomial coefficient terms at degree 6 will count too many contributions. Take, for example, the monomials* x^3yf_2 *and* $x^2y^2f_2$*. Their least common multiple is* $x^3y^2f_2$*, which means that the linearly dependent row* $x^3y^2f_2$ *is counted once too much.*

Example 3.11 shows that the the analysis of all syzygies is reduced to a combinatorial problem: within a group of basis syzygies, one needs to count the total number of linearly dependent rows these basis syzygies 'generate'. This combinatorial problem is solved by the Inclusion-Exclusion principle.

Theorem 3.4 *(Inclusion-Exclusion Principle [22, p. 454]) Let* $A_1, \ldots, A_n$ *be a collection of finite sets with* $|A_i|$ *the cardinality of* A_i*. Then*

$$|\cup_{i=1}^{n} A_i| \;=\; \sum_{k=1}^{n} (-1)^{k+1} \left(\sum_{1 \leq i_1 < \ldots < i_k \leq n} |A_{i_1} \cap \ldots \cap A_{i_k}| \right). \tag{3.14}$$

Proof. Pick an element of the union of all sets $A_1, \ldots, A_n$. This element is counted once by the left-hand side of (3.14), so it is necessary to show that it is

also counted only once by the right-hand side. From the binomial theorem we have that

$$(1-1)^n \;=\; \binom{n}{0} - \binom{n}{1} + \binom{n}{2} - \ldots + (-1)^n \binom{n}{n}.$$

Since $\binom{n}{0} = 1$, this can be rewritten as

$$\begin{aligned} 1 &= \tbinom{n}{1} - \tbinom{n}{2} + \ldots - (-1)^n \tbinom{n}{n} \\ &= |\{A_i | 1 \leq i \leq n\}| - |\{A_i \cap A_j | 1 \leq i < j \leq n\}| + \ldots \\ &\quad -(-1)^n |\{A_1 \cap A_2 \cap \ldots \cap A_n\}|, \end{aligned}$$

which shows that the right-hand side of (3.14) counts the chosen element indeed only once. □

Example 3.12 *Armed with the Inclusion-Exclusion Principle, we now continue and finalize the analysis of all syzygies of Example 3.11. If we denote the set of all monomial multiples of $x^3y\,f_2$ by A_1 and likewise A_2, A_3 for $x^2y^2\,f_2, xy^2z\,f_2$ respectively, then applying Theorem 3.4 on these sets results in the final expression for $l(d)$. Note that all terms of* (3.14) *for $k = 1$ are the binomial coefficients of Lemma 3.2, which already have been added to $l(d)$ in* (3.13)*. The remaining analysis is hence on all terms of* (3.14) *for $k \geq 2$. The cardinality of the intersections $A_1 \cap A_2, A_1 \cap A_3, A_2 \cap A_3$ are described by the binomial terms*

$$\binom{d-7+3}{3}, \binom{d-7+3}{3}, \binom{d-8+3}{3},$$

which will each contribute to $l(d)$ with a minus sign since $k = 2$. The degrees for which these terms are introduced are the degrees of the least common multiples between $x^3y\,f_2$ and $x^2y^2\,f_2$, between $x^3y\,f_2$ and $xy^2z\,f_2$ and between $x^2y^2\,f_2$ and $xy^2z\,f_2$. These degrees are 7, 8 and 7 respectively. The next intersection, $A_1 \cap A_2 \cap A_3$, corresponds with a binomial term introduced at the degree of the least common multiple of all 3 basis syzygies in f_2. This least common multiple is $x^3y^2z\,f_2$ with a degree of 8. This concludes the analysis of all linearly dependent rows of $M(d)$ and we can therefore write

$$\begin{aligned} l(d) &= \tbinom{d-4+3}{3} + 3\tbinom{d-6+3}{3} - 2\tbinom{d-7+3}{3} - \tbinom{d-8+3}{3} + \tbinom{d-8+3}{3} \\ &= \tbinom{d-4+3}{3} + 3\tbinom{d-6+3}{3} - 2\tbinom{d-7+3}{3} \\ &= \tfrac{1}{3}d^3 - 2d^2 + \tfrac{2}{3}d + 9. \; (d \geq 4) \end{aligned}$$

Note that the terms at degree eight have cancelled one another. Since the term $\binom{d-7+3}{3}$ has roots at $d \in \{4, 5, 6\}$, this expression for $l(d)$ is valid for all $d \geq 4$.

An important observation is that once $l(d)$ is known, then the rank of $M(d)$ and the dimension of its null space are also fully determined for all $d \geq 4$ by:

$$r(d) = p(d) - l(d) = \frac{1}{6}d^3 + d^2 + \frac{5}{6}d - 10,$$

$$c(d) = q(d) - r(d) = d + 11.$$

3.5.2 Degree of regularity

Example 3.12 illustrated how the expression for $l(d)$, and consequently for $r(d), c(d)$, changed during the syzygy analysis as new basis syzygies were found. The fact that there are only a finite number of basis syzygies therefore has an important implication: there exists a certain degree for which the polynomials that describe $l(d), r(d), c(d)$ do not change anymore for all $d \geq d^\star$.

Definition 3.2 *The degree $d^\star \in \mathbb{N}$ such that the polynomials found from the basis syzygy analysis describe $l(d), r(d), c(d)$ for all $d \geq d^\star$ is called the degree of regularity.*

This degree of regularity will be of vital importance when we discuss the null space of $M(d)$ in the next Section.

Remark 3.3 *Notice that in order to find the degree of regularity $d^\star$, it is required to construct the Macaulay matrix for a degree larger than $d^\star$. For example, the degree of regularity $d^\star = 4$ of the polynomial system in Example 3.10 was found from $M(6)$.*

The following example illustrates that the degree for which all basis syzygies are found can be quite high and consequently that the total number of binomial terms of $l(d)$ can be very large.

Example 3.13 *Consider the following polynomial system in C^4*

$$\begin{cases} f_1: & x_2^2\,x_3 + 2\,x_1\,x_2\,x_4 - 2\,x_1 - x_3 = 0, \\ f_2: & -x_1^3\,x_3 + 4\,x_1\,x_2^2\,x_3 + 4\,x_1^2\,x_2\,x_4 + 2\,x_2^3\,x_4 + 4\,x_1^2 - 10\,x_2^2 \\ & +4\,x_1\,x_3 - 10\,x_2\,x_4 + 2 = 0, \\ f_3: & 2\,x_2\,x_3\,x_4 + x_1\,x_4^2 - x_1 - 2\,x_3 = 0, \\ f_4: & -x_1\,x_3^3 + 4\,x_2\,x_3^2\,x_4 + 4\,x_1\,x_3\,x_4^2 + 2\,x_2\,x_4^3 + 4\,x_1\,x_3 \\ & +4\,x_3^2 - 10\,x_2\,x_4 - 10\,x_4^2 + 2 = 0, \end{cases}$$

with degrees $d_1 = d_3 = 3$, $d_2 = d_4 = 4$. *The first basis syzygy group is found in* $M(5)$ *and corresponds with the row* $x_2^2\, x_3\, f_3$. *The next basis syzygies group are the rows*

$$\{x_2^2\, x_3\, f_2,\ x_2^3\, x_4\, f_2,\ x_1\, x_2^2\, x_4^2\, f_2,\ x_1^3\, x_2\, x_3^2\, f_2,\ x_1^2\, x_2\, x_4^3\, f_2,\ x_1^4\, x_3\, x_4^4\, f_2,\ x_1^2\, x_2\, x_3^4\, x_4^2\, f_2,$$
$$x_2^2\, x_4^7\, f_2,\ x_1^2\, x_2\, x_3^6\, x_4\, f_2,\ x_1\, x_2\, x_4^8\, f_2,\ x_1^2\, x_2\, x_3^8\, f_2,\ x_1^3\, x_3\, x_4^8\, f_2\}$$

and are found for the degrees

$$\{6,\ 7,\ 8,\ 9,\ 12,\ 12,\ 12,\ 13,\ 13,\ 14,\ 15\}.$$

The last basis syzygy group are the rows

$$\{x_2^2\, x_3\, f_4,\ x_2\, x_3\, x_4\, f_4,\ x_1\, x_2\, x_4^2\, f_4,\ x_2^3\, x_4\, f_4,\ x_1^3\, x_4^2\, f_4,\ x_1^2\, x_4^3\, f_4,\ x_1^4\, x_3\, x_4\, f_4,$$
$$x_1^3\, x_2\, x_3^2\, f_4,\ x_1^3\, x_3^2\, x_4\, f_4,\ x_1^2\, x_3^3\, x_4\, f_4,\ x_1\, x_2^5\, f_4,\ x_1^5\, x_3^2\, f_4,\ x_1^4\, x_3^3\, f_4,\ x_1^3\, x_3^4\, f_4\}$$

and are found for the degrees

$$\{7,\ 7,\ 8,\ 8,\ 9,\ 9,\ 10,\ 10,\ 10,\ 10,\ 10,\ 11,\ 11,\ 11\}.$$

Application of the Inclusion-Exclusion Principle for each of these groups results in the final expression

$$\begin{aligned} l(d) &= \tbinom{d-6+4}{4} + 4\tbinom{d-7+4}{4} + \tbinom{d-8+4}{4} + 4\tbinom{d-13+4}{4} - 6\tbinom{d-11+4}{4} - \tbinom{d-14+4}{4} \\ &= \tfrac{1}{8}\, d^4 - \tfrac{13}{12}\, d^3 - \tfrac{5}{8}\, d^2 + \tfrac{19}{12}\, d + 105. \ (d \geq 10) \end{aligned}$$

Observe that $l(d)$ *consists in fact of 10241 binomial terms, of which 10225 cancel until only 16 terms are left. The degree of regularity is* $d^\star = 10$.

As the previous example showed, using the Inclusion-Exclusion principle to find the expression for $l(d)$ results in a large number of binomial terms. A large number of computations are actually wasted since most of these binomial terms cancel out and therefore do not contribute to $l(d)$. We will address this issue in Chapter 6, in which we present an recursive algorithm that computes $l(d)$ and avoids doing the wasted computations.

3.6 Null space

Due to the homogeneous interpretation of $\mathcal{M}_d$, the null space of the Macaulay matrix will be shown to be linked with the number of projective roots of the corresponding polynomial system. The notion of the dual of the row space will play an important role in describing these roots. We will also discuss the possible exchange of multiplicities of a root for extra polynomials and illustrate this with some examples. An algorithm for both affine root-finding and removing multiplicities will be presented in Chapter 6.

3.6.1 Link with projective roots

It is a classic result that for a polynomial system $f_1^h, \ldots, f_s^h$ with a finite number of projective roots, the quotient ring $\mathcal{P}^n / \langle f_1^h, \ldots, f_s^h \rangle$ is a finite-dimensional vector space [21, 22]. The dimension of this vector space equals the total number of projective roots of $f_1^h, \ldots, f_s^h$, counting multiplicities. From the rank-nullity theorem of $M(d)$ it then follows that

$$\begin{aligned} c(d) &= q(d) - r(d) \\ &= \dim \mathcal{P}_d^n - \dim \langle f_1^h, \ldots, f_s^h \rangle_d \\ &= \dim \mathcal{P}_d^n / \langle f_1^h, \ldots, f_s^h \rangle_d \end{aligned} \tag{3.15}$$

This leads to the following theorem.

Theorem 3.5 *For a zero-dimensional homogeneous ideal $\langle f_1^h, \ldots, f_s^h \rangle$ with m projective roots (counting multiplicities) and degree of regularity $d^\star$ we have that*

$$c(d) \;=\; m \quad \forall\, d \geq d^\star.$$

Proof. This follows from (3.15) and Definition 3.2. □

Furthermore, when $s = n$, then $c(d) = m = d_1 \cdots d_s$ according to Bézout's Theorem [21, p.97]. This effectively links the degrees of the polynomials $f_1, \ldots, f_s$ to the nullity of the Macaulay matrix. The roots can be retrieved from a generalized eigenvalue problem as discussed in [77, 78]. Another interesting result is that the degree of $c(d)$ is the dimension of projective variety of $f_1^h, \ldots, f_s^h$.

Definition 3.3 *The polynomial*

$$c(d) \;=\; dim\ \mathcal{P}_d^n / \langle f_1^h, \ldots, f_s^h \rangle_d \ (\forall\, d \geq d^\star)$$

is called the Hilbert Polynomial [22, p. 462]. The degree of this polynomial $c(d)$ equals the dimension of the projective variety [22, p.463].

Example 3.14 *For the polynomial system from Example 3.10 we had that $c(d) = d + 11$. Since this is a polynomial of degree one, it follows that the projective solution set of $f_1^h, \ldots, f_s^h$ is one-dimensional. The number of affine solutions of $f_1^h, \ldots, f_s^h$ is finite: $\{(1, 1, 1, -1), (1, -1, -1, -1)\}$, which implies that the nonzero-dimensional part of the solution set lies 'at infinity'.*

3.6.2 Dual vector space

As soon as $d \geq d^\star$ and the number of projective roots is finite, then a basis of the null space can be explicitly written down in terms of the roots. This requires the notion of the dual vector space (Appendix A Section A.4). We denote the dual vector space of $\mathcal{C}_d^n$ by $\mathcal{C}_d^{n'}$, the dual of $\mathcal{M}_d$ by $\mathcal{M}_d'$ and the annihilator of $\mathcal{M}_d$ by $\mathcal{M}_d^o$. By definition, the elements of $\mathcal{M}_d^o$ map each element of $\mathcal{M}_d$ to zero. Therefore, $\dim(\mathcal{M}_d^o) = c(d)$, which implies $\mathcal{M}_d^o \cong \text{null}(M(d))$. A basis for $\mathcal{M}_d^o$ will be described by differential functionals.

Definition 3.4 *Let $j \in \mathbb{N}_0^n$ and $z \in \mathbb{C}^n$, then the differential functional $\partial_j|_z \in \mathcal{C}_d^{n'}$ is defined by*

$$\partial_j|_z \equiv \partial_{x_1^{j_1} x_2^{j_2} \ldots x_n^{j_n}}|_x \equiv \frac{1}{j_1! \ldots j_n!} \frac{\partial^{j_1+\ldots+j_n}}{\partial x_1^{j_1} \ldots \partial x_n^{j_n}}|_z$$

where $|_z$ stands for evaluation in $z = (x_1, \ldots, x_n)$.

Being elements of the dual vector space, these differential functionals $\partial_j|_z$ can be represented as vectors. This is illustrated in the following simple example.

Example 3.15 *In $\mathcal{C}_3^{2'}$ the functionals $\partial_{00}|_z, \partial_{10}|_z, \partial_{01}|_z, \partial_{20}|_z, \partial_{11}|_z$ and $\partial_{02}|_z$ have the following coefficient vectors*

$$\begin{array}{c} \begin{array}{cccccc} \partial_{00}|_z & \partial_{10}|_z & \partial_{01}|_z & \partial_{20}|_z & \partial_{11}|_z & \partial_{02}|_z \end{array} \\ \left(\begin{array}{cccccc} 1 & 0 & 0 & 0 & 0 & 0 \\ x_1 & 1 & 0 & 0 & 0 & 0 \\ x_2 & 0 & 1 & 0 & 0 & 0 \\ x_1^2 & 2x_1 & 0 & 1 & 0 & 0 \\ x_1x_2 & x_2 & x_1 & 0 & 1 & 0 \\ x_2^2 & 0 & 2x_2 & 0 & 0 & 1 \\ x_1^3 & 3x_1^2 & 0 & 3x_1 & 0 & 0 \\ x_1^2x_2 & 2x_1x_2 & x_1^2 & x_2 & 2x_1 & 0 \\ x_1x_2^2 & x_2^2 & 2x_1x_2 & 0 & 0 & x_1 \\ x_2^3 & 0 & 3x_2^2 & 0 & 0 & 3x_2 \end{array} \right) \end{array}, \tag{3.16}$$

with $z = (x_1, x_2) \in \mathbb{C}^2$. The homogeneous interpretation of $M(d)$ implies that the differential functionals also have a homogeneous interpretation. For the example above, the coefficient vectors of the corresponding differential functionals in $\mathcal{P}_3^{2'}$

are

$$\begin{array}{c} \begin{array}{cccccc} \partial_{000}|_z & \partial_{010}|_z & \partial_{001}|_z & \partial_{020}|_z & \partial_{011}|_z & \partial_{002}|_z \end{array} \\ \left(\begin{array}{cccccc} x_0^3 & 0 & 0 & 0 & 0 & 0 \\ x_0^2 x_1 & x_0^2 & 0 & 0 & 0 & 0 \\ x_0^2 x_2 & 0 & x_0^2 & 0 & 0 & 0 \\ x_0 x_1^2 & 2x_0 x_1 & 0 & x_0 & 0 & 0 \\ x_0 x_1 x_2 & x_0 x_2 & x_0 x_1 & 0 & x_0 & 0 \\ x_0 x_2^2 & 0 & 2x_0 x_2 & 0 & 0 & x_0 \\ x_1^3 & 3x_1^2 & 0 & 3x_1 & 0 & 0 \\ x_1^2 x_2 & 2x_1 x_2 & x_1^2 & x_2 & 2x_1 & 0 \\ x_1 x_2^2 & x_2^2 & 2x_1 x_2 & 0 & 0 & x_1 \\ x_2^3 & 0 & 3x_2^2 & 0 & 0 & 3x_2 \end{array} \right) \end{array}, \tag{3.17}$$

with $z = (x_0, x_1, x_2) \in \mathbb{P}^2$. Note that (3.16) *can be retrieved from* (3.17) *by setting $x_0 = 1$. Also observe that for a root at infinity ($x_0 = 0$), the differential functionals will only have nonzero entries in the lower rows of their coefficient vectors.*

Remark 3.4 *We'll make no further distinction between the linear functionals $\partial_j|_z$ and their coefficient vectors. Notice that these coefficient vectors are column vectors, since they are the dual elements of the row space of $M(d)$. Applying the differential functionals $\partial_j|_z$ to the elements of $\mathcal{M}_d$ is then simply taking the inner product $M(d)\,\partial_j|_z$.*

The following lemma expresses the functional $\partial_j|_z$ in terms of $\partial_j|_0$ and will lead to the very interesting observation that multiplicities of roots can be exchanged for additional polynomials.

Lemma 3.3 *Let $\partial_j|_z \in \mathcal{C}_d^{n'}$, then the following relationship holds*

$$\partial_j|_z \;= D_j\; \partial_0|_z \tag{3.18}$$

with D_j a square matrix.

Proof. Since partial differentiation is a linear operator it can be represented as a square matrix D_j and hence the proof is trivial. When $j = 0$, then D_j is simply the unit matrix. □

The following example illustrates how the partial differential operator D_j in matrix form.

Example 3.16 *In $\mathcal{C}_2^{2'}$, we have*

$$\partial_{01}|_z \;=\; D_{01}\; \partial_{00}|_z,$$

or written out in full

$$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ x_1 \\ 2x_2 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1x_2 \\ x_2^2 \end{pmatrix}.$$

Remark 3.5 *Higher order partial derivatives can be decomposed into products of first order derivative matrices. For example, D_{21} in $\mathcal{C}_d^2$ can be decomposed as $D_{21} = D_{10}D_{10}D_{01}$. Since the order of applying these partial derivatives is not relevant, this means that all D_j must form a family of commuting matrices.*

Knowing how any functional $\partial_j|_z$ can be expressed as derivatives of $\partial_j|_0$ leads to the following trivial corollary for the coefficient vectors of polynomials.

Corollary 3.2 *Let $f \in \mathcal{C}_d^n$, then the evaluation of the jth order partial derivative of f in $z \in \mathbb{C}^n$ is*

$$f\ \partial_j|_z\ = f\ D_j\ \partial_0|_z. \tag{3.19}$$

Observe that the right-hand side of (3.19) can be interpreted in two ways. The partial derivative operator D_j has been either applied to the polynomial coefficient vector f or to the functional $\partial_0|_z$. In the former case, this results in the coefficient vector of the jth order partial derivative of f, $f\,D_j$. In the latter case, we end up with the differential functional $D_j\,\partial_0|_z$. We know that when a polynomial system $f_1^h, \ldots, f_s^h$ has a finite number of m projective roots, then $\dim(\mathcal{M}_d^o) = c(d) = m\ \forall\, d \geq d^\star$. Hence, a basis for $\mathcal{M}_d^o$ will consist of differential functionals, evaluated in each projective root and taking multiplicities into account.

Definition 3.5 *Let $f_1, \ldots, f_s \in \mathcal{C}^n$ with a zero-dimensional projective variety and let $m_1, \ldots, m_t$ be the multiplicities of the t projective roots $z_i\,(1 \leq i \leq t)$ such that $\sum_{i=1}^t m_i = m$. Then $\forall\, d \geq d^\star$ there exists a matrix K of m linearly independent columns such that*

$$M(d)\ K\ =\ 0.$$

Furthermore, K can be partitioned into

$$K = \begin{pmatrix} K_1 & K_2 & \ldots & K_t \end{pmatrix},$$

such that each K_i consists of m_i linear combinations of differential functionals $\partial_j|_{z_i} \in \mathcal{P}_d^{n'}\ (1 \leq i \leq t)$. We will call this matrix K the canonical null space of $M(d)$.

Definition 3.5 explicitly depends on the homogeneous interpretation of $M(d)$. Indeed, it is only for the homogeneous case that the projective roots come into the picture. The following example illustrates the structure of the matrix K for a small example.

Example 3.17 *Let us reconsider the small polynomial system from Example 3.1*

$$\left\{ \begin{array}{lrcl} f_1: & x_1x_2 - 2x_2 & = & 0, \\ f_2: & x_2 - 3 & = & 0, \end{array} \right.$$

with 1 affine root $(2,3)$. *From Example 3.8 we know that its corresponding projective variety consists of 2 points: the affine solution* $(1,2,3)$ *and the root at infinity* $(0,1,0)$. *Indeed,* $c(d) = 2 \;\; (\forall\; d \geq 2)$ *and therefore the canonical null space is*

$$\begin{aligned} K &= \begin{pmatrix} \partial_{000}|_{(1,2,3)} & \partial_{000}|_{(0,1,0)} \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \\ 2 & 0 \\ 3 & 0 \\ 4 & 1 \\ 6 & 0 \\ 9 & 0 \end{pmatrix}. \end{aligned}$$

Defining the multiplicity of a zero using the dual space goes back to Macaulay [61]. It is also reminiscent of the univariate case. Remember that for a univariate polynomial $f(x) \in \mathcal{C}_d^1$ a zero z with multiplicity m means that

$$\begin{pmatrix} f \\ f\,D_1 \\ \vdots \\ f\,D_{m-1} \end{pmatrix} \partial_0|_z \;=\; 0. \tag{3.20}$$

Or in other words, $f(z) = f'(z) = f'' = \ldots = f^{(m-1)}(z) = 0$. By using (3.18) and Definition 3.5, we can also write (3.20) as

$$f \;\begin{pmatrix} \partial_0|_z & \partial_1|_z & \partial_2|_z & \ldots & \partial_{m-1}|_z \end{pmatrix} \;=\; 0. \tag{3.21}$$

As already mentioned in Definition 3.5, the multivariate case generalizes this principle by requiring linear combinations of differential functionals.

Example 3.18 *Consider the following polynomial system in* $\mathcal{C}_2^2$ *with the affine root* $z = (2,3)$ *of multiplicity 4 and no roots at infinity*

$$\left\{ \begin{array}{rcl} (x_2 - 3)^2 & = & 0, \\ (x_1 + 1 - x_2)^2 & = & 0. \end{array} \right.$$

The degree of regularity $d^\star$ is 2. We will show that

$$K \;=\; \begin{pmatrix} \partial_{00}|_{(2,3)} & \partial_{10}|_{(2,3)} & \partial_{01}|_{(2,3)} & 2\partial_{20}|_{(2,3)} + \partial_{11}|_{(2,3)} \end{pmatrix} \tag{3.22}$$

is an affine basis for the annihilator $\mathcal{M}_d^o$.

The different linear combinations of functionals needed to construct K_i are called the multiplicity structure of the root z_i. Observe that the multiplicity structure of a root is not unique. Indeed, for any nonsingular $m_i \times m_i$ matrix T we have that the column space of K_i equals the column space of $K_i\,T$. Or written in symbols, $\mathrm{col}(K_i) = \mathrm{col}(K_i\,T)$. Finding the multiplicity structure for a given root of a polynomial system is an active area of research [28, 93]. It is easy to see that this corresponds with finding the null space of a particular matrix.

Example 3.19 *For Example 3.18 we have that the Macaulay matrix $M(2)$ is*

$$M(2) \;=\; \begin{pmatrix} 9 & 0 & -6 & 0 & 0 & 1 \\ 1 & 2 & -2 & 1 & -2 & 1 \end{pmatrix},$$

with corresponding basis C of $\mathcal{C}_2^{2'}$ in the root $(2,3)$

$$C \;=\; \begin{array}{c} \begin{matrix} \partial_{00}|_{(2,3)} & \partial_{10}|_{(2,3)} & \partial_{01}|_{(2,3)} & \partial_{20}|_{(2,3)} & \partial_{11}|_{(2,3)} & \partial_{02}|_{(2,3)} \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 1 & 0 & 0 & 0 \\ 4 & 4 & 0 & 1 & 0 & 0 \\ 6 & 3 & 2 & 0 & 1 & 0 \\ 9 & 0 & 6 & 0 & 0 & 1 \end{pmatrix} \end{array}.$$

Computing the multiplicity structure of the root $(2,3)$ is equivalent with finding a matrix N such that

$$M(2)\,C\,N \;=\; 0. \tag{3.23}$$

N hence specifies the different linear combinations of the columns of C that make up the multiplicity structure of the root. From (3.23) it is clear that N can be found as a basis for the null space of $W = M(2)\,C$. Taking the inner product $M(2)\,C$ results in

$$W \;=\; \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & -2 & 1 \end{pmatrix},$$

with $\mathrm{rank}(W) = 2$. The multiplicity structure of the root $(2,3)$, or linear combination of the differential functionals as in (3.22), can be easily read off from

the null space N of W.

$$N = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Iterative algorithms to compute the multiplicity structure of a root z such as in [93] exploit the closedness property of the differential functionals $\partial_j|_z$ [78, p. 330] to reduce the size of W in every iteration. We will not further discuss these algorithms in this thesis.

3.6.3 Removing multiplicities of affine roots

Observe from (3.20) and (3.21) that, for the univariate case, an exchange between the multiplicity of a root and the number of polynomials is possible. Indeed, the m polynomials $fD_1, \ldots, fD_{m-1}$ of (3.20), which only have 1 root, are exchanged for additional multiplicities $\partial_1|_z, \ldots, \partial_{m-1}|_z$ in (3.21). The same idea applies for the multivariate case. The following example illustrates this for a particular simple case.

Example 3.20 *We revisit the polynomial system from Example 3.18 with multiplicity structure*

$$\begin{pmatrix} \partial_{00}|_{(2,3)} & \partial_{10}|_{(2,3)} & \partial_{01}|_{(2,3)} & 2\partial_{20}|_{(2,3)} + \partial_{11}|_{(2,3)} \end{pmatrix}.$$

Using Corollary 3.2 we can apply the differential operators on the polynomials f_1 and f_2, which results in the polynomial system

$$\begin{cases} f_1 D_{00}: & x_2^2 - 6\,x_2 + 9, \\ f_2 D_{00}: & x_1^2 + x_2^2 + 2\,x_1 - 2\,x_2 - 2\,x_1\,x_2 + 1, \\ f_1 D_{10}: & 0, \\ f_2 D_{10}: & 2\,x_1 + 2 - 2\,x_2, \\ f_1 D_{01}: & 2\,x_2 - 6, \\ f_2 D_{01}: & 2\,x_2 - 2\,x_1 - 2, \\ 2\,f_1 D_{20} + f_1 D_{11}: & 0, \\ 2\,f_2 D_{20} + f_2 D_{11}: & 2. \end{cases}$$

Adding the polynomials that are reduced to scalars would make the polynomial system inconsistent. These are therefore removed to obtain

$$\begin{pmatrix} f_1 \\ f_2 \\ f_2 D_{10} \\ f_1 D_{01} \\ f_2 D_{01} \end{pmatrix} \partial_{00}|_{(2,3)} \;=\; 0,$$

which is a polynomial system that now has the root $z = (2, 3)$ with multiplicity 1.

Exchanging the multiplicities of a root for additional polynomials as shown in Example 3.20 is only possible if every root z_i has the exact same multiplicity structure $K_i\,(1 \leq i \leq t)$. When this is not the case another approach is required, one that surprisingly does not need any information on the roots of the system. The concept of the radical ideal together with square-free parts of polynomials play an important role here. This will be further worked out in Chapter 6.

3.6.4 Conditions for existence of particular roots

In this section we will derive 2 conditions to determine whether a polynomial system has either the zero root $0 = (0, \ldots, 0)$ or roots at infinity. Since the monomial ordering that we use is graded, we can introduce the following convenient block partitioning of the Macaulay matrix

$$M(d) \;=\; \begin{pmatrix} M_0 & M_1 & M_2 & \ldots & M_d \end{pmatrix}, \tag{3.24}$$

where each M_i block corresponds with all monomials of degree $i\,(0 \leq i \leq d)$. The first theorem provides an easy way to check whether 0 is a root of a given polynomial system.

Theorem 3.6 *Let $f_1, \ldots, f_s \in \mathcal{C}_d^n$ and $M(d)$ its Macaulay matrix of degree $d \geq d^\star$. Then 0 is a root of $f_1, \ldots, f_s$ if and only if $M_0 = 0$.*

Proof. If 0 is a root of $f_1, \ldots, f_s$ then we can write

$$M(d)\; \partial_0|_0 \;=\; 0,$$

which can also be written as

$$\begin{pmatrix} M_0 & M_1 & M_2 & \ldots & M_d \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \;=\; 0, \tag{3.25}$$

where $M(d)$ is partitioned as in (3.24) and accordingly for $\partial_0|_0$. Taking the inner product of the left-hand side of (3.25) results in $M_0 = 0$. On the other hand, if $M_0 = 0$, then a basis for null$(M(d))$ will clearly always, up to a nonsingular linear transformation, have an entry $\partial_0|_0$. □

Hence, if the polynomial system has no constant terms, then it will always have $(0, \ldots, 0)$ as a solution. In order to find out whether there are any roots at infinity, we first make the following observation.

Lemma 3.4 *The solutions at infinity of the polynomial system $f_1, \ldots, f_s$ are the only solutions of*

$$\begin{cases} f_1^h &= 0, \\ f_2^h &= 0, \\ \vdots & \\ f_s^h &= 0, \\ x_0 &= 0. \end{cases}$$

When every f_i $(1 \leq i \leq s)$ is made homogeneous, then powers of x_0 are added to each term that is not of degree d_i. Lemma 3.4 therefore means that one needs to solve the polynomial system that is retained when only the highest order terms of each polynomial f_i are kept. This leads to the following condition on the existence of roots at infinity.

Theorem 3.7 *A polynomial system $f_1, \ldots, f_s$ has roots at infinity if and only if the coefficient block M_d is not of full column rank for all Macaulay matrices of degree $d \geq d^\star$.*

Proof. According to Lemma 3.4, the roots at infinity are the only solutions of the polynomial system that contains only the highest order terms of $f_1, \ldots, f_s$. The homogeneous polynomial system

$$M_d \, K \;=\; 0, \tag{3.26}$$

consists of the polynomials of Lemma 3.4 multiplied with monomials x^α such that each polynomial is of degree d. Obviously, the solutions of (3.26) are the roots at infinity and possibly the origin, which is not a valid point in $\mathbb{P}^n$. From this it follows that M_d cannot be of full column rank if $f_1, \ldots, f_s$ has roots at infinity. □

Example 3.21 *As seen in Example 3.8, the homogeneous interpretation of the polynomial system in Example 3.1 was*

$$\begin{cases} f_1^h : & x_1x_2 - 2x_2x_0 &= 0, \\ f_2^h : & x_2 - 3x_0 &= 0, \end{cases}$$

with two roots $(1,2,3)$ *and* $(0,1,0)$*. Its degree of regularity* $d^\star$ *is 2 and the corresponding Macaulay matrix is*

$$M(2) = \begin{array}{c} \begin{array}{cccccc} 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 \end{array} \\ \begin{pmatrix} 0 & 0 & -2 & 0 & 1 & 0 \\ -3 & 0 & 1 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 & 1 & 0 \\ 0 & 0 & -3 & 0 & 0 & 1 \end{pmatrix} \end{array}.$$

The M_2 *coefficient block is not of full column rank and corresponds with the polynomial system*

$$\begin{cases} x_1x_2 &= 0, \\ x_1x_2 &= 0, \\ x_2^2 &= 0. \end{cases}$$

The root of this polynomial system is indeed the root at infinity $(0,1,0)$.

Chapter 4

Fast Recursive Orthogonalization of the Macaulay matrix

Orthogonal bases for either the row space or null space of the Macaulay matrix are essential for all algorithms in this thesis. In this chapter, a fast recursive orthogonalization scheme is derived that allows to determine the rank of the Macaulay matrix and provides the required orthogonal bases. This recursive algorithm exploits both the structure and sparsity of the Macaulay matrix and will be an essential ingredient for all the algorithms of Chapter 6.

4.1 Introduction

Many problems concerning multivariate polynomials can be phrased and solved in the PNLA framework that was setup in the previous chapters. In Chapter 6 of this thesis we will address a wide range of problems:

- computing a Gröbner basis,
- computing the affine roots of a system of multivariate polynomials,
- solving the ideal membership problem,
- doing the syzygy analysis and finding the degree of regularity,
- multivariate polynomial elimination,

- computing an approximate LCM or GCD of two multivariate polynomials,
- removing multiplicities of roots.

The algorithms to solve these different problems all share the same general structure as shown in Algorithm 4.1. The goal is always to compute some desired quantity X (e.g. the affine roots of the polynomial system, a polynomial from which certain variables are eliminated, etc...). The essential object in Algorithm 4.1 is the Macaulay matrix $M(d)$. The degree $d^\dagger$ for which X can be computed from $M(d)$ is in general not known beforehand. Algorithm 4.1 will therefore iterate over increasing values of the degree, starting from $d = \max(d_1, \ldots, d_s)$. In each iteration, the Macaulay matrix is constructed and orthogonal bases for either its row space or null space need to be computed. These orthogonal bases are then used to compute X. Since the Macaulay matrix is by definition rank-deficient, the first principal step is to determine its (numerical) rank. The most robust way to determine the rank and find the orthogonal bases is the SVD. A less computational expensive alternative is the rank-revealing QR decomposition. Both the SVD and the rank revealing QR decomposition will be considered in this chapter. If for the current iteration the desired quantity X cannot be computed, then the degree d is incremented by one.

Algorithm 4.1 *General Algorithm*
Input: *polynomial system* $f_1, \ldots, f_s$ *of degrees* $d_1, \ldots, d_s$
Output: *desired output* X

```
X ← ∅
d_0 ← max(d_1, ..., d_s)
while X = ∅ do
  M(i) ← construct Macaulay matrix for degree i
  U ← orthogonal basis for row space of M(i)
  N ← orthogonal basis for null space of M(i)
  X ← try to compute desired output from U and/or N
  if X = ∅ then
    d ← d + 1
  end if
end while
```

A naive implementation of this general algorithm does not exploit the structure of the Macaulay matrix nor does it use computations of previous iterations when recomputing the orthogonal bases U, N for a higher degree. This is remedied by the recursive orthogonalization scheme presented in this chapter. The naive implementation of Algorithm 4.1 is considerably improved by:

- reducing the computational complexity by working with a submatrix of $M(d)$ instead of the complete matrix,

- introducing an updating strategy for the orthogonal bases U, N which reuses the orthogonal bases from the previous step.

Before presenting the orthogonalization scheme, we first introduce some necessary notation.

4.2 Notation

The orthogonalization scheme derived in Section 4.3 requires the use of a graded monomial ordering. We will use the convention that $M(d)$ is a $p \times q$ matrix whereas $M(d+1)$ is $p' \times q'$. This implies of course that $p' = p + \Delta p$ and $q' = q + \Delta q$. The dimension of the row space $\mathcal{M}_d$ is the rank r and the dimension of its null space $c = q - r$. An orthogonal basis for $\mathcal{M}_d$ is hence $q \times r$ and will be denoted by $U(d)$. Likewise, an orthogonal basis for the null space of $M(d)$ is the $q \times c$ matrix $N(d)$. When the Macaulay matrix $M(d)$ is constructed as explained in Example 3.1, then its transpose can be partitioned as

$$M(d+1)^T \;=\; \begin{matrix} \\ q \\ \Delta q \end{matrix} \begin{matrix} \begin{matrix} p & \Delta p \end{matrix} \\ \begin{pmatrix} M(d)^T & M_a \\ 0 & M_b \end{pmatrix} \end{matrix}$$

where the M_b block contains all coefficients of monomials with total degree $d+1$. Now let

$$\begin{matrix} \\ c \\ \Delta q \end{matrix} \begin{matrix} \begin{matrix} \Delta p \end{matrix} \\ \begin{pmatrix} N(d)^T M_a \\ M_b \end{pmatrix} \end{matrix} \;=\; Q\, S\, V^T \tag{4.1}$$

be the SVD of $\left(M_a^T N(d)\; M_b^T\right)^T$. We denote the rank of this matrix by Δr. Suppose without loss of generality that $c + \Delta q > \Delta p$, then Q, S and V can be partitioned as

$$\begin{matrix} \\ c \\ \Delta q \end{matrix} \begin{matrix} \begin{matrix} \Delta p \end{matrix} \\ \begin{pmatrix} N(d)^T M_a \\ M_b \end{pmatrix} \end{matrix} \;=\; \begin{matrix} \\ c \\ \Delta q \end{matrix} \begin{matrix} \begin{matrix} \Delta r & c' \end{matrix} \\ \begin{pmatrix} L_1 & K_1 \\ L_2 & K_2 \end{pmatrix} \end{matrix} \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix}$$

where Σ is diagonal and contains Δr nonzero singular values. This partitioning of the left singular vectors will be crucial in the proof of the main theorem in the next section.

4.3 The Orthogonalization Scheme

Now, all notation is in place to present the following main theorem. The recursive orthogonalization scheme of the Macaulay matrix is a direct application of this theorem.

Theorem 4.1 *Let $U(d), N(d), M_a, M_b, L_1, L_2, K_1, K_2$ be the matrices as defined in Section 4.2. Then the following relationships hold:*

$$U(d+1) \;=\; \begin{pmatrix} U(d) & N(d)L_1 \\ 0 & L_2 \end{pmatrix} \quad \textit{and} \quad N(d+1) \;=\; \begin{pmatrix} N(d)K_1 \\ K_2 \end{pmatrix}.$$

Proof. For the sake of readability the degree (d) will be dropped from the notation $M(d), U(d), N(d)$. We start with the observation that the matrices

$$M(d+1)^T \;=\; \begin{pmatrix} M^T & M_a \\ 0 & M_b \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} U & M_a \\ 0 & M_b \end{pmatrix}$$

share the same range and left null space. The orthogonal U block in the rightmost matrix can be diagonalized by left-multiplying it with the orthogonal matrix $\begin{pmatrix} U & N \end{pmatrix}^T$. This can be written as

$$\begin{array}{c} \\ r \\ c \\ \Delta q \end{array} \begin{array}{c} \begin{array}{cc} q & \Delta q \end{array} \\ \begin{pmatrix} U^T & 0 \\ N^T & 0 \\ 0 & I_{\Delta q} \end{pmatrix} \end{array} \begin{array}{c} \begin{array}{cc} r & \Delta p \end{array} \\ \begin{pmatrix} U & M_a \\ 0 & M_b \end{pmatrix} \end{array} \;=\; \begin{array}{c} \begin{array}{cc} r & \Delta p \end{array} \\ \begin{pmatrix} I_r & U^T M_a \\ 0 & N^T M_a \\ 0 & M_b \end{pmatrix} \end{array}. \tag{4.2}$$

The orthogonal matrix can be moved to the right-hand side to obtain

$$\begin{array}{c} \\ q \\ \Delta q \end{array} \begin{array}{c} \begin{array}{cc} r & \Delta p \end{array} \\ \begin{pmatrix} U & M_a \\ 0 & M_b \end{pmatrix} \end{array} \;=\; \begin{array}{c} \\ q \\ \Delta q \end{array} \begin{array}{c} \begin{array}{ccc} r & c & \Delta q \end{array} \\ \begin{pmatrix} U & N & 0 \\ 0 & 0 & I_{\Delta q} \end{pmatrix} \end{array} \begin{array}{c} \begin{array}{cc} r & \Delta p \end{array} \\ \begin{pmatrix} I_r & U^T M_a \\ 0 & N^T M_a \\ 0 & M_b \end{pmatrix} \end{array}. \tag{4.3}$$

From (4.2) it is straightforward to see that the rank increase

$$\Delta r \triangleq \operatorname{rank}\left(M(d+1) \right) - \operatorname{rank}\left(M(d) \right)$$

is given by

$$\Delta r = \operatorname{rank}\left(\begin{pmatrix} N^T M_a \\ M_b \end{pmatrix} \right),$$

since

$$\begin{aligned}\operatorname{rank}(\,M(d+1)) &= \operatorname{rank}\left(\begin{pmatrix} U & M_a \\ 0 & M_b \end{pmatrix}\right), \\ &= \operatorname{rank}\left(\begin{pmatrix} I_r & U^T M_a \\ 0 & N^T M_a \\ 0 & M_b \end{pmatrix}\right).\end{aligned}$$

The next step is to replace $\left(M_a^T N\ M_b^T\right)^T$ by its SVD in (4.3) to obtain

$$\left(\begin{array}{c|c} U & M_a \\ \hline 0 & M_b \end{array}\right) = \left(\begin{array}{c|cc} U & N & 0 \\ \hline 0 & 0 & I_{\Delta q} \end{array}\right) \left(\begin{array}{c|c} I_r & U^T M_a \\ \hline 0 & QSV^T \end{array}\right). \tag{4.4}$$

The Q can be factored out from the rightmost matrix in the following manner

$$\begin{pmatrix} I_r & U^T M_a \\ 0 & QSV^T \end{pmatrix} = \begin{array}{c} \\ r \\ c \\ \Delta q \end{array} \begin{array}{c} \begin{array}{ccc} r & \Delta r & c' \end{array} \\ \begin{pmatrix} I_r & 0 & 0 \\ 0 & L_1 & K_1 \\ 0 & L_2 & K_2 \end{pmatrix} \end{array} \begin{pmatrix} I_r & U^T M_a \\ 0 & \Sigma V_1^T \\ 0 & 0 \end{pmatrix}. \tag{4.5}$$

Since Δr is the increase in rank this implies that $c' = c + \Delta q - \Delta r = c + \Delta c$ is the dimension of the null space of $M(d+1)$. Substituting (4.5) into (4.4) results in

$$\begin{pmatrix} U & M_a \\ 0 & M_b \end{pmatrix} = \begin{array}{c} \\ q \\ \Delta q \end{array} \begin{array}{c} \begin{array}{ccc} r & \Delta r & c' \end{array} \\ \begin{pmatrix} U & NL_1 & NK_1 \\ 0 & L_2 & K_2 \end{pmatrix} \end{array} \begin{pmatrix} I_r & U^T M_a \\ 0 & \Sigma V_1^T \\ 0 & 0 \end{pmatrix}. \tag{4.6}$$

The left matrix of the right-hand side is the product of two orthogonal matrices and hence also orthogonal. The theorem follows from (4.6). □

Observe that the orthogonal basis $U(d+1)$ retains a similar block structure as $M(d+1)$ and will therefore also be sparse. Theorem 4.1 can be immediately translated into Algorithm 4.2 to orthogonalize $M(d)$. The algorithm starts for the initial degree $d_0 \triangleq \max(d_1, \ldots, d_s)$. An orthogonal basis for $\mathcal{M}_d$ and the null space of $M(d_0)$ are computed from its SVD. The subsequent steps of the algorithm are then to construct the extra rows $\left(M_a^T\ M_b^T\right)^T$ and update the orthogonal bases $U(d), N(d)$ using Theorem 4.1. Considering the sparsity of $M(d)$ and $U(d)$, it would be interesting to be able to use a sparse matrix data structure, like the column compressed form. This avoids storing the large amount of zero entries in memory. A complete SVD is however not available for matrices using a sparse matrix data structure but a rank revealing multifrontal QR decomposition [24] is. This allows us to replace the rank test of (4.1) by

$$\begin{pmatrix} N(d)^T M_a \\ M_b \end{pmatrix} = Q\ R\ P^T \tag{4.7}$$

where Q is orthogonal, R is upper triangular and P is a column permutation which reduces fill-in of R. Furthermore, if $\left(M_a^T\, N(d)\, M_b^T\right)^T$ is not of full column rank then R can be partitioned as

$$R \;=\; \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix}$$

such that the numerical rank can be estimated from the number of nonzero diagonal elements of R_{11}. This naturally leads to an implementation of Theorem 4.1 with matrices using a sparse matrix data structure. All SVDs are then replaced by sparse rank revealing multifrontal QR decompositions and (4.6) is then given by

$$\begin{pmatrix} U & M_a \\ 0 & M_b \end{pmatrix} \;=\; \begin{matrix} q \\ \Delta q \end{matrix} \begin{matrix} \begin{matrix} r & \Delta\, r & c' \end{matrix} \\ \begin{pmatrix} U & NL_1 & NK_1 \\ 0 & L_2 & K_2 \end{pmatrix} \end{matrix} \begin{pmatrix} I_r & U^T M_a \\ 0 & R_{11}P_1^T + R_{12}P_2^T \\ 0 & 0 \end{pmatrix}.$$

Algorithm 4.2 *Recursive Orthogonalization of $M(d)$*
Input: *polynomial system $f_1, \ldots, f_s$ of degrees $d_1, \ldots, d_s$, degree d*
Output: *orthogonal bases $U(d), N(d)$*

$d_0 \leftarrow \max(d_1, \ldots, d_s)$
$U, N \leftarrow$ *orthogonal bases for row space and null space of* $M(d_0)$
for $i = d_0 + 1 \ldots d$ ***do***
  construct M_a, M_b *for degree* i
  $Q \leftarrow SVD(\left(M_a^T N\, M_b^T\right)^T)$ *or* $QR(\left(M_a^T N\, M_b^T\right)^T)$
  $\begin{pmatrix} L_1 & K_1 \\ L_2 & K_2 \end{pmatrix} \leftarrow Q$
  $U \leftarrow \begin{pmatrix} U & NL_1 \\ 0 & L_2 \end{pmatrix}$
  $N \leftarrow \begin{pmatrix} NK_1 \\ K_2 \end{pmatrix}$
end for

It is now possible to adjust Algorithm 4.1 such that it uses the recursive orthogonalization scheme of Theorem 4.1. The pseudo-code for this update is shown in Algorithm 4.3.

Algorithm 4.3 *Updated Algorithm Macaulay matrix*
Input*: polynomial system $f_1, \ldots, f_s$ of degrees $d_1, \ldots, d_s$*
Output*: desired output X*

$d \leftarrow \max(d_1, \ldots, d_s)$
$U, N \leftarrow$ *orthogonal bases for row space and null space of of* $M(d)$
$X \leftarrow$ *try to compute desired output from* U *and/or* N
$d \leftarrow d + 1$
while $X = \emptyset$ ***do***
  construct M_a, M_b *for degree* d
  $Q \leftarrow SVD(\left(M_a^T N\ M_b^T\right)^T)$ *or* $QR(\left(M_a^T N\ M_b^T\right)^T)$
  $U, N \leftarrow$ *update* U, N *using* Q *and Theorem 4.1*
  $X \leftarrow$ *try to compute desired output from* U *and* N
  if $X = \emptyset$ ***then***
    $d \leftarrow d + 1$
  end if
end while

4.4 Computational Complexity

The most expensive computational step in both Algorithm 4.1 and Algorithm 4.3 is the computation of the orthogonal bases. In this section an estimate of the gain in computational complexity is derived for both the SVD and sparse QR-based implementation.

4.4.1 SVD

We first provide an estimate on the total number of operations for computing the SVD of a complete $M(d)$. We will assume from here on that the number of columns of $M(d)^T$ is always bigger than the number of rows. In each iteration of Algorithm 4.1 the SVD is computed from the complete Macaulay matrix. Not the full SVD is required however. Only the left singular vectors Q and the diagonal matrix S are needed. This takes about $4p(d)q(d)^2 + 8q(d)^3$ operations [40, p. 254]. We approximate both $p(d), q(d)$ by their highest order term in order to have an estimate on their order of magnitude. Substituting this into the expression for the number of operations results in the following estimate for the computational cost of the SVD in Algorithm 4.1

$$4\left(\frac{sd^n}{n!}\right)\left(\frac{d^n}{n!}\right)^2 + 8\left(\frac{d^n}{n!}\right)^3 = \frac{4(s+2)}{(n!)^3}\, d^{3n}. \tag{4.8}$$

The SVD-step in Algorithm 4.3 is applied on a $(c + \Delta q) \times \Delta p$ matrix. $c(d)$ is a polynomial of maximal degree $n - 1$. This is due to Definition 3.3: the degree of the Hilbert Polynomial is the dimension of the solution set of $f_1^h, \ldots, f_s^h$, which is maximally $n - 1$. Obviously, the highest order terms of $\Delta p(d)$ and $\Delta q(d)$ are $sd^{n-1}/(n-1!)$ and $d^{n-1}/(n-1!)$ respectively. Retaining the highest order terms in the expression for the total amount of operations gives us the following estimate for the cost of the SVD in Algorithm 4.3

$$4\left(\frac{sd^{n-1}}{(n-1)!}\right)\left(\frac{d^{n-1}}{(n-1)!}\right)^2 + 8\left(\frac{d^{n-1}}{(n-1)!}\right)^3 = \frac{4(s+2)}{(n-1)!^3}\, d^{3n-3}. \tag{4.9}$$

Dividing (4.8) by (4.9) leads to an estimated gain of d^3/n^3 operations when using the recursive orthogonalization scheme. This gain can be quite substantial for large degrees d and small number of variables n (Figure 4.1). Memory is still the bottleneck for the orthogonalization of $M(d)$ for large degrees however. Although the SVD of a smaller submatrix needs to be computed, it still grows with $\approx O(d^{n-1})$. This polynomial growth is unfortunately inherent to problems which involve multivariate polynomials.

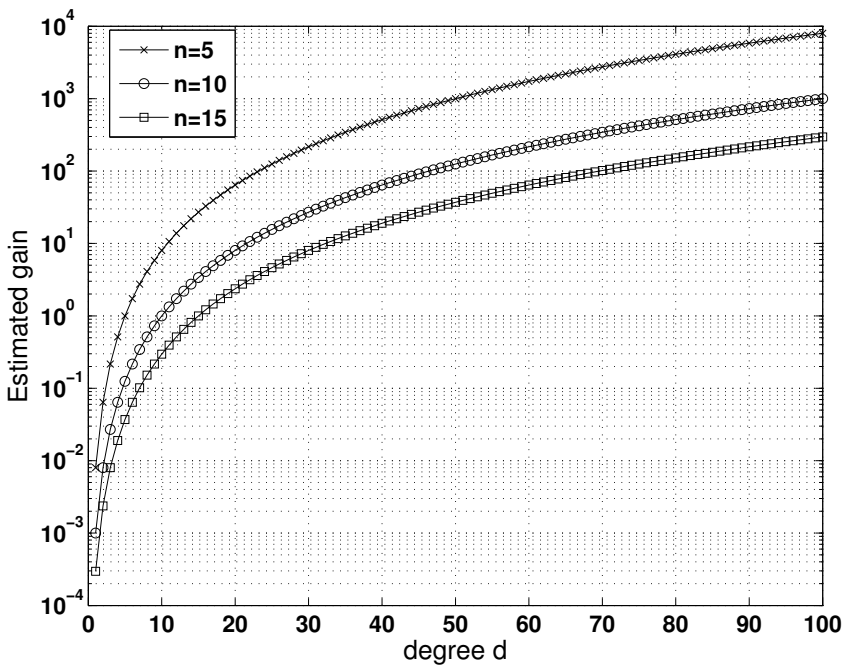

Figure 4.1: **Estimated gain (ratio of** (4.8) **and** (4.9)**) in number of operations as a function of the degree** d**.**

4.4.2 Rank revealing QR decomposition

In order to describe the computational complexity of the rank revealing QR decomposition, the assumption is made that Businger-Golub column pivoting [18] is used. This serves as an upper bound on the complexity since it does not take the sparsity pattern of the Macaulay matrix into account. In practice, a sparse multifrontal QR decomposition algorithm can be used which will exploit the structure. For a $q(d) \times p(d)$ Macaulay matrix $M(d)^T$ of rank $r(d)$, the computational complexity of the Businger-Golub QR factorization is given by $4p(d)q(d)r(d) - 2r(d)^2(p(d) + q(d)) + 4r(d)^3/3$ [40, p. 250]. In this expression $r(d)$ can be replaced by $q(d) - c(d)$ where $c(d)$ is again a polynomial of maximal degree $n-1$. Like before, only higher order terms are retained for $p(d), q(d)$ and substituted into $4p(d)q(d)r(d) - 2r(d)^2(p(d) + q(d)) + 4r(d)^3/3$ to obtain

$$4\,\frac{sd^{3n}}{(n!)^3} - 2(s+1)\frac{d^{3n}}{(n!)^3} + \frac{4}{3}\,\frac{d^{3n}}{(n!)^3} = \frac{2(3s-1)}{3(n!)^3}\,d^{3n}. \tag{4.10}$$

The same reasoning can be applied for the $(c + \Delta q) \times \Delta p$ submatrix which leads to

$$4\,\frac{sd^{3n-3}}{((n-1)!)^3} - 2(s+1)\frac{d^{3n-3}}{((n-1)!)^3} + \frac{4}{3}\,\frac{d^{3n-3}}{((n-1)!)^3} = \frac{2(3s-1)}{3((n-1)!)^3}\,d^{3n-3}$$

with the same gain of d^3/n^3 operations when using the recursive orthogonalization scheme. Comparing (4.8) with (4.10) reveals that the QR decomposition will take about

$$\frac{6(s+2)}{(3s-1)}$$

times less operations than the SVD. This factor reaches 3 for $s = 5$ and then asymptotically approaches 2 in the limit for large s (Figure 4.2).

4.5 Choosing the numerical tolerance τ

All computations are performed in double precision. This sets the machine precision to $\epsilon \approx 2.22 \times 10^{-16}$. A crucial step in the recursive algorithm is the determination of the numerical rank. The determination of an incorrect numerical rank during one of the iterations affects all consequent iterations. A good choice for the numerical tolerance is therefore of the utmost importance to guarantee a correct result. For the SVD-based approach, numerical experiments indicate that a 'standard' choice of $\tau(d) = \max(q(d), p(d))\,||M(d)||_2\,\epsilon$ is a good choice. Let $\sigma_1 \geq \ldots \geq \sigma_{c+\Delta p}$ be the singular values of $\left(M_a^T N\, M_b^T\right),^T$ then the numerical rank Δr is chosen such that

$$\sigma_1 \geq \ldots \geq \sigma_{\Delta r} \geq \tau(d) \geq \sigma_{\Delta r+1} \geq \ldots \geq \sigma_{c+\Delta p}.$$

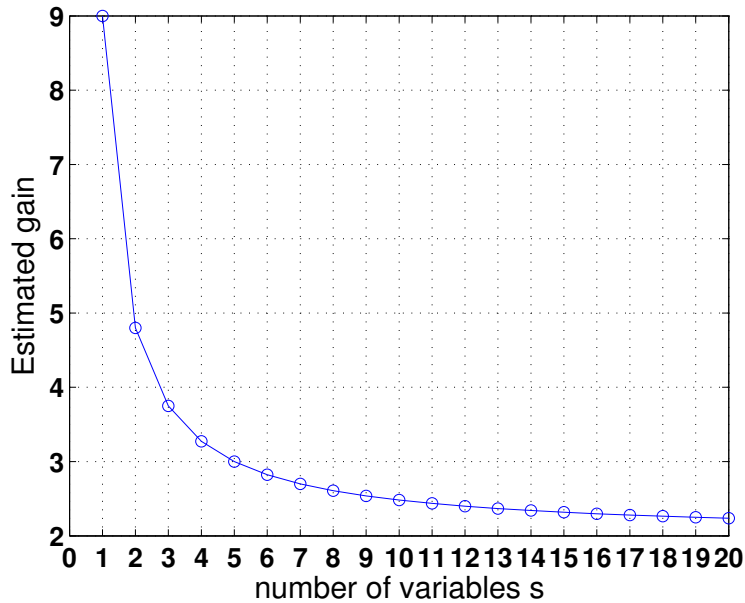

Figure 4.2: **Estimated gain (ratio of** (4.8) **and** (4.10)**) in number of operations as a function of the degree** d**.**

The approx-rank gap $\sigma_{\Delta r}/\sigma_{\Delta r+1}$ [58, p.920] then serves as a measure of how well the numerical rank is defined. Indeed, if there is a large gap between $\sigma_{\Delta r}$ and $\sigma_{\Delta r+1}$ and $\tau(d)$ lies between these two values then small changes in $\tau(d)$ will not affect the determination of the numerical rank. When the default value for the numerical tolerance fails, one could try to determine the numerical rank such that the approxi-rank gap is maximal. This will be explored in numerical experiments in Section 4.6. The sparse multifrontal QR decomposition method from [24], SuiteSparseQR, uses the numerical tolerance $\tau = 20\,(m + \Delta p + \Delta q)\,\epsilon\,D$ where D is the largest 2-norm of any row of $\left(M_a^T N\, M_b^T\right)^T$. The numerical rank is then estimated as the number of nonzero diagonal entries. The rank-revealing QR decomposition is known to be less reliable for the rank determination. It is reported in [24] that SuiteSparseQR is able to correctly determine the correct numerical rank for about two-thirds of the rank deficient matrices in the University of Florida Sparse Matrix Collection [27]. The numerical rank for many of those matrices is ill-defined. If the numerical rank was very well-defined (approxi-rank gap $> 10^3$), then 95 % of the time the numerical rank was correctly determined. We have observed that dense polynomial systems for which every possible monomial has a nonzero coefficient tend to produce Macaulay matrices for which it is difficult to determine the rank.

The difficulty lies not in a small approxi-rank gap but rather in a failing of the default choices for the tolerance. For these cases it is recommended for the user to manually check either the singular values or the diagonal elements of R. This is illustrated in the next section.

4.6 Numerical Experiments

The effectiveness of the orthogonalization scheme is illustrated by comparing the total run times and required memory for three different ways for the orthogonalization of $M(d)$. The reference method for computing $U(d)$ and $N(d)$ is by simply computing a SVD of $M(d)$. This approach will be called 'naive SVD' in the numerical experiments. Both the SVD and sparse QR-based approach of Algorithm 4.2 will be compared to the reference method. They will be called 'recursive SVD' and 'recursive QR' respectively. In addition to total run time, the required memory to store the matrices during the orthogonalization is also compared for the three aforementioned methods. For the naive SVD, this is the memory required to store $M(d)$, while Algorithm 4.2 needs to store $U(d-1), N(d-1), \begin{pmatrix} M_a^T & M_b^T \end{pmatrix}^T$ using either a dense or sparse matrix data structure.

4.6.1 Example 1

In the first numerical experiment the capability of the algorithms to deal with high total degrees is tested. The polynomial system consists of 3 polynomials in 3 unknowns of total degree 12:

$$\begin{cases} f_1: & x_1^{12} + x_2^{12} + x_3^{12} - 4 &= 0, \\ f_2: & x_1^{12} + 2\,x_2^{12} - 5 &= 0, \\ f_3: & x_1^6\,x_3^6 - 1 &= 0. \end{cases}$$

The Macaulay matrix was orthogonalized for degrees $d = 12, \ldots, 35$. The total run time for these degrees is shown in Figure 4.3. The first observation is that for the naive SVD the total run time seems to increase approximately linearly on the log plot, which implies an exponential increase. In contrast, the total run time for the recursive algorithms does not grow so fast. Therefore, a continuously increasing 'gap' can be seen between the naive orthogonalization and the recursive algorithms. The ratio between the run times of the recursive SVD and QR orthogonalizations increases very slowly. For $d = 35$, the recursive QR is 119 times faster than the naive approach and 6 times faster than the recursive SVD. Also at this degree, the recursive SVD is 18 times faster than the naive approach. All numerical ranks were correctly determined by each of the 3 different methods.

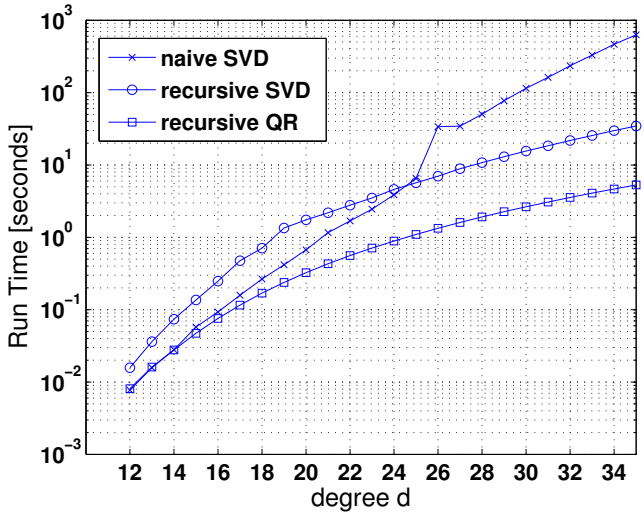

Figure 4.3: **Total run time Example 1 as a function of the degree d.**

The graph of the total amount of required memory for orthogonalization is shown in Figure 4.4. For this particular case the recursive SVD method needs more memory than the naive method. This is due to the need of the recursive method to store both $U(d-1)$ and $N(d-1)$, in addition to $\begin{pmatrix} M_a^T & M_b^T \end{pmatrix}^T$. It is only for the rather large degree of 35 that the recursive SVD method starts to need less memory than the naive approach. Since the polynomial system was already very sparse, the sparse QR implementation of Algorithm 4.2 is clearly superior for this example. Whereas for $d = 35$ both SVD-based methods require about 500 MB, the sparse QR method needs only 1MB.

4.6.2 Example 2

For the next numerical experiment, the number of variables is increased to 6 and the degrees are limited to maximally 4:

$$\begin{cases} x_1^2 + x_3^2 - 1 &= 0, \\ x_2^2 + x_4^2 - 1 &= 0, \\ x_5\, x_3^3 + x_6\, x_4^3 - 1.2 &= 0, \\ x_5\, x_1^3 + x_6\, x_2^3 - 1.2 &= 0, \\ x_5\, x_3^2\, x_1 + x_6\, x_4^2\, x_2 - 0.7 &= 0, \\ x_5\, x_3\, x_1^2 + x_6\, x_4\, x_2^2 - 0.7 &= 0. \end{cases}$$

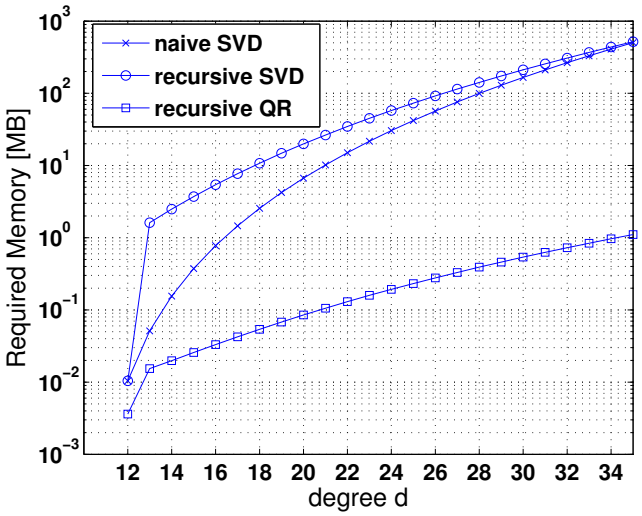

Figure 4.4: **Required memory Example 1 as a function of the degree d.**

As shown in Figure 4.5, all total run times are quite similar for degrees 4 up to 7. Differences become more apparent as soon as the total run time is around 10 seconds. Again, the naive approach has the largest exponential increase of run time. For $d = 10$, the total run times for the naive, recursive SVD and recursive QR method are 831, 92 and 12 seconds respectively.

The total required memory, shown in Figure 4.6, for both SVD-based orthogonalizations is quite similar. The sparse QR implementation needs one order of magnitude less memory. For $d = 10$, the SVD-based methods need about 592 MB while the sparse QR needs only 44 MB. All numerical ranks were correctly determined by each of the 3 different methods.

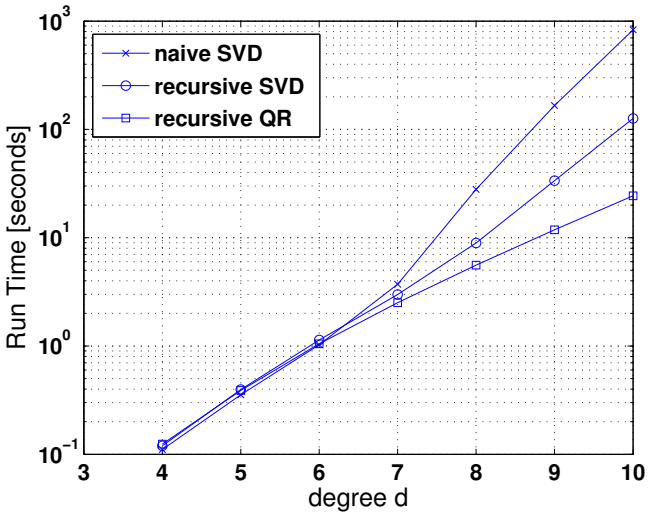

Figure 4.5: **Total run time Example 2 as a function of the degree d.**

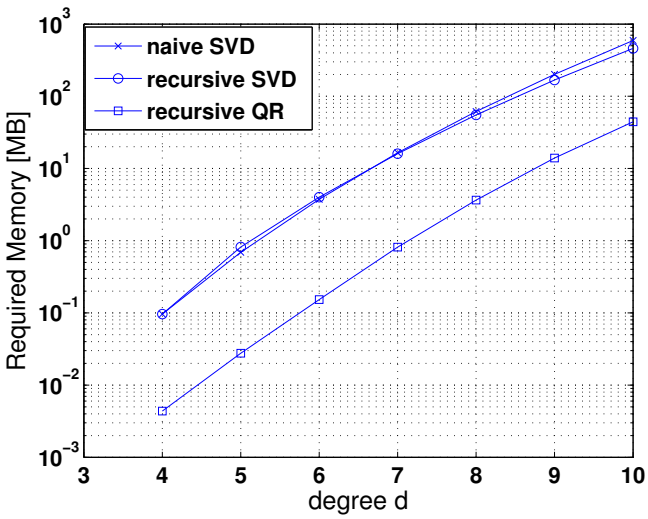

Figure 4.6: **Required memory Example 2 as a function of the degree d.**

4.6.3 Example 3

The number of variables and polynomials is further increased to 10 polynomials in 10 variables of total degree 2

$$\begin{cases} 5\,x_1\,x_2 + 5\,x_1 + 3\,x_2 + 55 = 0, \\ 7\,x_2\,x_3 + 9\,x_2 + 9\,x_3 + 19 = 0, \\ 3\,x_3\,x_4 + 6\,x_3 + 5\,x_4 - 4 = 0, \\ 6\,x_4\,x_5 + 6\,x_4 + 7\,x_5 + 118 = 0, \\ x_5\,x_6 + 3\,x_5 + 9\,x_6 + 27 = 0, \\ 6\,x_6\,x_7 + 7\,x_6 + x_7 + 72 = 0, \\ 9\,x_7\,x_8 + 7\,x_7 + x_8 + 35 = 0, \\ 4\,x_8\,x_9 + 4\,x_8 + 6\,x_9 + 16 = 0, \\ 8\,x_9\,x_{10} + 4\,x_9 + 3\,x_{10} - 51 = 0, \\ 3\,x_1\,x_{10} - 6\,x_1 + x_{10} + 5 = 0. \end{cases}$$

Observe that the different run times for this polynomial system in Figure 4.7 have a similar growth as in Figure 4.5. For this particular example, the three run times become different starting from $d = 4$. The difference between the naive and recursive SVD, however, appears from $d = 6$. For this degree, $M(6)$ is a 10010×8008 matrix and the naive, recursive SVD and recursive QR algorithms need 916, 373 and 61 seconds respectively.

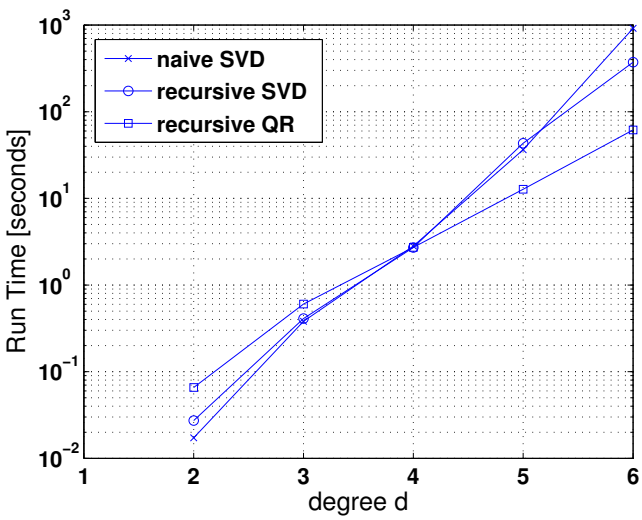

Figure 4.7: **Total run time Example 3 as a function of the degree d.**

Also for the memory requirement, Figure 4.8 is almost identical to Figure 4.6. Both SVD-based methods need practically the same amount of memory while the

sparse QR-based method needs about one order of magnitude less. At $d = 6$, the SVD-based methods need about 611 MB while the sparse QR-method needs only 50 MB. All numerical ranks were correctly determined by each of the 3 different methods.

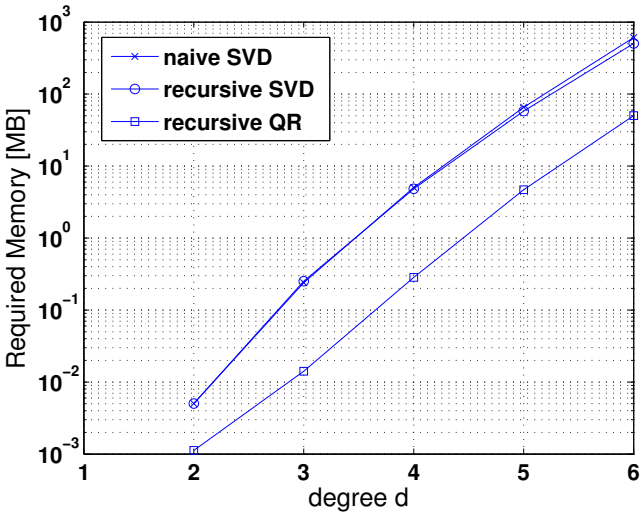

Figure 4.8: **Required memory Example 3 as a function of the degree d.**

4.6.4 Example 4

In this example we illustrate the failure of the default numerical tolerances when working with dense polynomial systems. The polynomial system consists of 3 polynomials, each of degree 4. Each polynomial consists of 35 terms, corresponding to all possible monomials in 3 variables from degrees 0 up to 4. The integer coefficients are uniformly drawn from the interval $[-100, 100]$. Normalizing the coefficient vector of each polynomial such that it is a unit vector keeps the singular values small. For this particular instance the rank test fails at degree $d = 23$. The matrix $\left(M_a^T N\ M_b^T\right)^T$ is then 570×340. We first discuss the SVD-based algorithm. The numerical tolerance is $\tau = 1.26 \times 10^{-13}$. The singular values σ_{276} up to σ_{280} are

$$\{0.1266, 1.72 \times 10^{-13}, 1.30 \times 10^{-13}, 1.05 \times 10^{-13}, 5.01 \times 10^{-14}\}.$$

Using the default tolerance would determine the numerical rank to be 278 with a corresponding approxi-rank gap of 1.24. The numerical rank however is well-determined to be 276 with an approxi-rank gap of $7.34\,10^{11}$. It is clear from this example that maximizing the approxi-rank gap over all pairs of consecutive

singular values would correctly retrieve the numerical rank. Care needs to be taken on how this maximization is carried out, since it is possible that there is more than one "large" gap. The QR-based algorithm suffers from the same problem. In contrast to the singular values, the diagonal entries of R, denoted by r_{ii}, are not all positive nor sorted in descending order. We therefore denote the values of $|r_{ii}|$, sorted in descending order, by s_j where j runs from 1 to $\min(m+\Delta q, \Delta p)$. One can then do a similar analysis on s_j as with the singular values. At $d = 10$ the default numerical tolerance is $\tau = 1.11 \times 10^{-12}$ and the last two nonzero elements of s_j for $d = 10$ are $s_{78} = 0.044, s_{79} = 2.79 \times 10^{-12}$. The numerical rank determined from using the default tolerance is 79 while the singular values indicate it should be 78 with an approxi-rank gap of 2.34×10^{12}. Maximizing the ratio of two consecutive nonzero s_j values would also in this case retrieve the correct numerical rank.

4.6.5 Example 5

The following polynomial system demonstrates the failure of the rank revealing QR decomposition to correctly determine the rank. As will be shown for the degree $d = 8$, this failure is not related to the default values of the numerical tolerances. Consider the following polynomial system in $\mathcal{C}_2^4$:

$$\begin{cases} 2\,x_4^2 + 2\,x_3^2 + 2\,y^2 + 2\,x_1^2 - x_1 &= 0, \\ 2\,x_3\,x_4 + 2\,x_2\,x_3 + 2\,x_1\,x_2 - x_2 &= 0, \\ 2\,x_2\,x_4 + 2\,x_1\,x_3 + x_2^2 - x_3 &= 0, \\ 2\,x_4 + 2\,x_3 + 2\,x_2 + x_1 - 1 &= 0. \end{cases}$$

At $d = 7$, the default tolerance for the QR-based orthogonalization fails. Inspecting $s_{120} = 0.068$ and $s_{121} = 2.01 \times 10^{-12}$ shows that the numerical rank should be 120 instead of 121 although the default tolerance is $\tau = 1.69 \times 10^{-12}$. The SVD-based method confirms the numerical rank of 120 with an approxi-rank gap of $\sigma_{120}/\sigma_{121} = 2.85 \times 10^{14}$. At $d = 8$, the numerical rank is estimated by the rank revealing QR to be 166. Indeed, the numerical tolerance is $\tau = 2.27 \times 10^{-12}$ with $s_{166} = 0.062$ and $s_{167} = 0$. The SVD-based algorithm however shows that the numerical rank is 165 and well determined with an approxi-rank gap $\sigma_{165}/\sigma_{166} = 2.10 \times 10^{14}$.

Chapter 5

The Canonical Decomposition of $\mathcal{C}_d^n$

In this chapter, we introduce the notions of both the canonical and reduced canonical decomposition of $\mathcal{C}_d^n$. These concepts will play a crucial role in almost all recursive algorithms described in Chapter 6 by allowing us to determine stop criteria for all of these recursive algorithms. In this way we will show that, contrary to methods in computer algebra, no explicit computation of a Gröbner basis is necessary.

5.1 The canonical decomposition of $\mathcal{C}_d^n$

Starting from the rank of the Macaulay matrix the canonical decomposition is defined. Then, the SVD-based algorithm to compute the canonical decomposition numerically is presented. This is followed by a detailed discussion on numerical aspects, which are illustrated by worked-out examples.

5.1.1 Definition

Knowing the interpretation of the row space $\mathcal{M}_d$, one can immediately give a similar interpretation for the rank $r(d)$ of $M(d)$. Evidently, the rank $r(d)$ counts the number of linearly independent polynomials lying in $\mathcal{M}_d$. More interestingly, the rank also counts the number of linearly independent leading monomials of $\mathcal{M}_d$. This is easily seen from bringing the Macaulay matrix $M(d)$ into a reduced row echelon form $R(d)$. In order for the linearly independent monomials to be leading

monomials a column permutation Q is required which flips all columns from left to right. Then the Gauss-Jordan elimination algorithm can be run, working from left to right. The reduced row echelon form then ensures that each pivot element corresponds with a linearly independent leading monomial. We illustrate this procedure in the following example.

Example 5.1 *Consider the polynomial system*

$$\left\{ \begin{array}{lrcl} f_1: & x_1\,x_2 - 2x_2 & = & 0, \\ f_2: & x_2 - 3 & = & 0, \end{array} \right.$$

and fix the degree to 3. First, the left-to-right column permutation Q is applied to $M(3)$,

$$M(3)\,Q \;=\; \begin{array}{c} \\ f_1 \\ x_1 f_1 \\ x_2 f_1 \\ f_2 \\ x_1\,f_2 \\ x_2\,f_2 \\ x_1^2\,f_2 \\ x_1x_2\,f_2 \\ x_2^2\,f_2 \end{array} \begin{array}{c} \begin{array}{cccccccccc} x_2^3 & x_1x_2^2 & x_1^2x_2 & x_1^3 & x_2^2 & x_1x_2 & x_1^2 & x_2 & x_1 & 1 \end{array} \\ \left(\begin{array}{cccccccccc} 0 & 0 & 0 & 0 & 0 & 1 & 0 & -2 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -3 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -3 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -3 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & 0 & 0 \end{array} \right) \end{array}.$$

Now bringing $M(3)\,Q$ into reduced row echelon form results in

$$R(3) \;=\; \begin{array}{c} \begin{array}{cccccccccc} x_2^3 & x_1x_2^2 & x_1^2x_2 & x_1^3 & x_2^2 & x_1x_2 & x_1^2 & x_2 & x_1 & 1 \end{array} \\ \left(\begin{array}{cccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -27 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -18 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -12 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -9 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right) \end{array}.$$

From the reduced row echelon form one can see that the rank of $M(3)$ is 8. Notice how the left-to-right permutation ensured that the 8 pivot elements, corresponding with the monomials

$$\{x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^2x_2, x_1x_2^2, x_2^3\},$$

are leading monomials with respect to the monomial ordering. The Gauss-Jordan algorithm returns a set of 8 polynomials, that all together span $\mathcal{M}_3$. In addition,

for each of these polynomials, its leading monomial corresponds with a particular pivot element of $R(3)$.

This Gaussian elimination is in fact similar to Algorithm 2.1 that finds a monomial basis for the remainder space $\mathcal{R}$ when doing multivariate polynomial division. The $r(d)$ polynomials that can be read off from $R(d)$ span $\mathcal{M}_d$ and we will show how for a particular degree a subset of these polynomials corresponds with a reduced Gröbner basis. Interpreting the rank $r(d)$ in terms of linearly independent leading monomials naturally leads to a canonical decomposition of $\mathcal{C}_d^n$. The vector space spanned by the $r(d)$ leading monomials of $R(d)$ will be denoted $\mathcal{A}_d$. Its complement spanned by the remaining monomials will be denoted $\mathcal{B}_d$. We will call these monomials that span $\mathcal{B}_d$ the normal set or standard monomials.

Definition 5.1 *Let* $f_1, \ldots, f_s$ *be a multivariate polynomial system with a given monomial ordering. Then we define the canonical decomposition as the decomposition of the monomial basis of* $\mathcal{C}_d^n$ *into a set of linearly independent leading monomials* $A(d)$ *and standard monomials* $B(d)$.

Naturally,

$$\mathcal{C}_d^n \;=\; \mathcal{A}_d \oplus \mathcal{B}_d$$

and $\dim \mathcal{A}_d = r(d), \dim \mathcal{B}_d = c(d)$. Observe that the monomial bases for $\mathcal{A}_d$ and $\mathcal{B}_d$ also have a homogeneous interpretation.

Example 5.2 *For the polynomial system of the previous example,*

$$\left\{ \begin{array}{lrcl} f_1: & x_1\,x_2 - 2x_2 & = & 0, \\ f_2: & x_2 - 3 & = & 0, \end{array} \right.$$

and degree 3, the canonical decomposition is hence

$$A(3) \;=\; \{x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^2x_2, x_1x_2^2, x_2^3\},$$

and

$$B(3) \;=\; \{1, x_1^3\}.$$

In matrix form, these monomial bases are

$$A(3) \;=\; \begin{array}{c} \begin{array}{cccccccccc} 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 & x_1^3 & x_1^2x_2 & x_1x_2^2 & x_2^3 \end{array} \\ \left(\begin{array}{cccccccccc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right) \end{array},$$

and

$$B(3) = \begin{array}{c} \begin{array}{cccccccccc} 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 & x_1^3 & x_1^2x_2 & x_1x_2^2 & x_2^3 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}.$$

For the sake of readability the notation for $A(d)$ and $B(d)$ is used for both the set of monomials and the matrices, as in Example 5.2. The dependence of the canonical decomposition on the monomial ordering is easily understood from Example 5.2. A different admissible monomial ordering would correspond with a different column permutation Q and this would result in different monomial bases $A(3)$ and $B(3)$.

5.1.2 Importance of the canonical decomposition

The importance of this canonical decomposition is twofold. First, the linearly independent monomials $A(d)$ play an important role in the computation of a Gröbner basis of $f_1, \ldots, f_s$ and in the determination of the various stop criteria of the recursive algorithms in Chapter 6. Second, the normal set $B(d)$ is intimately linked with the problem of finding the roots of the polynomial system $f_1, \ldots, f_s$. Remember from Section 3.6 that

$$\begin{aligned} c(d) &= q(d) - r(d) \\ &= \dim \mathcal{P}_d^n - \dim \langle f_1^h, \ldots, f_s^h \rangle_d \\ &= \dim \mathcal{P}_d^n / \langle f_1^h, \ldots, f_s^h \rangle_d \\ &= \dim \mathcal{B}_d. \end{aligned}$$

Or in other words, for all degrees d larger than the degree of regularity $d^\star$, the number of monomials in $B(d)$ are equal to the total amount of projective roots of the polynomial system $f_1^h, \ldots, f_s^h$. In Chapter 6, we will show how the affine roots can be separated from the roots at infinity and computed from a generalized eigenvalue problem.

It is commonly known that bringing a matrix into a reduced row echelon form is numerically not the most reliable way of determining the rank of a matrix. In the next section, a more robust SVD-based method for computing the canonical decomposition of $\mathcal{C}_d^n$ and finding the polynomial basis $R(d)$ is presented.

5.1.3 Numerical Computation of the Canonical Decomposition

As mentioned in the previous section, the determination of $r(d)$ is the first essential step in computing the canonical decomposition of $\mathcal{C}_d^n$. Bringing the matrix into reduced row echelon form by means of Gauss-Jordan elimination is not a robust method for determining the rank. Furthermore, since the monomial ordering is fixed, no column pivoting is allowed, which potentially results in numerical instabilities. Our recursive orthogonalization algorithm is therefore the method of choice to determine the numerical rank. The next step is to find $A(d), B(d)$ and the $r(d)$ polynomials of $R(d)$. The key idea here is that each of these $r(d)$ polynomials is spanned by the standard monomials and one leading monomial of $A(d)$. Suppose a subset $A \subseteq A(d)$ and $B \subseteq B(d)$, both ordered in ascending order, are available. It is then possible to test whether the next monomial larger than the largest monomial of $A(d)$ is a linearly independent leading monomial. We illustrate the principle by the following example.

Example 5.3 *Suppose that the following subsets*

$$A \;=\; \{x_1, x_2\}, \qquad B \;=\; \{1\},$$

of

$$A(3) = \{x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^2x_2, x_1x_2^2, x_2^3\},\; B(3) = \{1, x_1^3\},$$

from Example 5.2 are available. The next monomial according to the monomial ordering is x_1^2. The next possible polynomial from $R(3)$ is then spanned by $\{1, x_1^2\}$. If such a polynomial lies in $\mathcal{M}_3$ then x_1^2 is a linearly independent leading monomial and can be added to A. If not, x_1^2 should be added to B. This procedure can be repeated until all monomials up to degree 3 have been tested. For the case of x_1^2 there is indeed such a polynomial present in $R(3)$ as can be seen from Example 5.2: $x_1^2 - 4$. This polynomial therefore lies in both the vector spaces $\mathcal{M}_3$ and span$(1, x_1^2)$. Computing a basis for the intersection between $\mathcal{M}_3$ and span$(1, x_1^2)$ will therefore reveal whether $x_1^2 \in A(3)$.

Given the subsets A and B, testing whether a monomial $x^a \in A(d)$ corresponds with computing the intersection between $\mathcal{M}_d$ and span(B, x^a). This corresponds with detecting whether the smallest principal angle between $\mathcal{M}_d$ and span(B, x^a) is zero (Appendix A, Sections A.9 and A.10). $\{B, x^a\}$ is a monomial basis, its matrix representation Q_2 can therefore be written as

$$Q_2 \;=\; P\,E,$$

where P is a permutation matrix and $E = (I_m \; 0)^T$, with $m = |\{B, x^a\}|$. From Section A.10, it then also follows that the sines of the principal angles are given by the singular values of $N^T\,Q_2$. If the indices of the nonzero rows of Q_2 are given by j, then the SVD of $N(j,:)^T$ needs to be computed. The same tolerance

τ used for the determination of the numerical rank of $M(d)$ in Algorithm 4.2 can be used to decide whether a principal angle is numerically zero. This then also implies that the reduced polynomial r can be retrieved as the right singular vector v_m corresponding with the zero singular value μ_m. The whole algorithm is summarized in pseudo-code in Algorithm 5.1. The first step is to compute a basis N for null$(M(d))$ using Algorithm 4.2. Next, the algorithm iterates over all n-variate monomials from degree 0 up do d, in ascending order. The set containing all these monomials is denoted by $\mathcal{T}_d^n$. The computational complexity of the orthogonalization step is $O(d^{3n-3})$. The subsequent $q(d)$ SVDs have a maximal computational complexity of $O(d^{3n-3})$, which amounts to a total computational complexity of $O(d^{4n-3})$. This is only valid if $\deg(c(d)) = n-1$. For the case of a finite number of projective roots, the computational cost is completely dominated by the orthogonalization step.

Algorithm 5.1 *Computation of the Canonical Decomposition of* $\mathcal{C}_d^n$
***Input**: polynomial system* $f_1, \ldots, f_s$*, degree* d*, tolerance* τ
***Output**:* $A(d), B(d)$ *and polynomials* $R(d)$

```
N ← orthogonal basis for null space of M(d) using Algorithm 4.2
A(d), B(d), R(d) ← ∅
for all x^a ∈ T_d^n do
   construct vector of indices j from B(d) and x^a
   [W S Z] ← SVD( N(j,:)^T )
   if arcsin(μ_m) < τ then
      append x^a to A(d)
      append v_m^T to R(d)
   else
      append x^a to B(d)
   end if
end for
```

Remark 5.1 *The orthogonal basis U for $\mathcal{M}_d$ was never used in Algorithm 5.1. Hence, during the recursive orthogonalization of $M(d)$, it is never required to update or store this matrix. This reduces the total required memory significantly.*

5.1.4 Numerical Experiment - no perturbations on the coefficients

We first consider the case of polynomials with exact coefficients and illustrate Algorithm 5.1 using the SVD-based iterative orthogonalization.

Example 5.4 *Consider the following polynomial system in* $\mathcal{C}_4^3$

$$\left\{\begin{array}{rcl} x_1^2 + x_1\,x_3 - 2\,x_2 + 5 &=& 0, \\ 2\,x_1^3\,x_2 + 7\,x_2\,x_3^2 - 4\,x_1\,x_2\,x_3 + 3\,x_1 - 2 &=& 0, \\ x_2^4 + 2\,x_2\,x_3 + 5\,x_1^2 - 5 &=& 0, \end{array}\right.$$

with degrees $d_1 = 2, d_2 = 4, d_3 = 4$. *The canonical decomposition is computed for* $d = 10$ *with Algorithm 5.1. Each polynomial is normalized and the* 333×286 *Macaulay matrix* $M(10)$ *is constructed. From its SVD the tolerance is set to* $\tau = 1.47 \times 10^{-13}$ *and the numerical rank is determined as 254 with an approxi-rank gap of* $\approx 4 \times 10^{13}$. *This implies that* $A(10)$ *and* $B(10)$ *will have 254 and 32 monomials respectively. Algorithm 5.1 indeed returns this number of monomials and corresponding polynomials* $R(10)$. *The principal angles corresponding with the leading monomials* $A(10)$ *are all around* 10^{-15} *and hence the tolerance* τ *for the rank test also works for the principal angles. The smallest principal angle for a monomial of the normal set* $B(10)$ *is* 2.17×10^{-9}. *The rank estimated from the reduced row echelon form of* $M(10)$ *is 259 which is a strong indication that the reduced row echelon form is not well-suited to compute* $A(10), B(10)$ *and* $R(10)$.

5.1.5 Numerical Experiment - perturbed coefficients

Perturbing the coefficients of the polynomial system $f_1, \ldots, f_s$ will change the corresponding canonical decomposition $A(d), B(d)$. This is easily observed from Example 5.2. Suppose we perturb all coefficients of f_2 with noise, uniformly distributed over the interval $[0, 10^{-1}]$, to obtain

$$\tilde{f_2} = -2.9056 + 0.0456\,x_1 + 1.0789\,x_2.$$

Then the reduced row echelon form of $\tilde{M}(3)Q$ is

$$\tilde{R}(3) \;=\; \begin{array}{c} \begin{array}{cccccccccc} x_2^3 & x_1x_2^2 & x_1^2x_2 & x_1^3 & x_2^2 & x_1x_2 & x_1^2 & x_2 & x_1 & 1 \end{array} \\ \left(\begin{array}{cccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2876 & -18.3252 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2205 & -14.0501 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0.1691 & -10.7723 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -4191.63 & 8375.27 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0.1102 & -7.0250 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.0845 & -5.3862 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -65.7197 & 127.4393 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.0423 & -2.6931 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right) \end{array}.$$

The corresponding canonical decomposition hence becomes

$$\tilde{A}(3) = \{x_2^3, x_1x_2^2, x_1^2x_2, x_1^3, x_2^2, x_1x_2, x_1^2, x_2\} \;\text{ and }\; \tilde{B}(3) = \{1, x_1\}.$$

A small continuous change of coefficients has therefore led to a 'jump' of the canonical decomposition. This implies that the computation of the canonical decomposition for a given polynomial system under perturbations of its coefficients is an ill-posed problem. This ill-posedness is called a representation singularity [78, p. 325] and is due to the insistence that the monomials of $A(d)$ need to be leading monomials with respect to the monomial ordering. This condition is sufficient to make the representation singularity unavoidable and has implications for the numerical determination of Gröbner bases. An alternative approach to avoid this representation singularity is the use of border bases [52,53,65]. In practice however, one does not know the normal set and will still need to compute a reduced row echelon form of $M(d)$ using Gaussian elimination or Algorithm 5.1. We discuss border bases and an algorithm to compute them in Section 5.3.

5.2 The Reduced Canonical Decomposition of $\mathcal{C}_d^n$

In this section we introduce the notion of divisibility into the canonical decomposition. This naturally leads to the concept of a reduced canonical decomposition, which will play an important role in the computation of a numerical Gröbner basis together with the determination of stop criteria for the recursive algorithms in Chapter 6. First, some new notation and concepts are introduced after which Algorithm 5.1 is adjusted such that it produces the reduced decomposition. A numerical example is then worked out and discussed.

5.2.1 The Reduced Monomials $A^\star(d), B^\star(d)$ and Polynomials $G(d)$

The polynomial basis $R(d)$ will grow unbounded with the rank $r(d)$ when d increases. It is possible however to reduce this basis to a finite subset that generates the whole ideal $\langle f_1, \ldots, f_s \rangle$. It will be shown in Chapter 6 Section 6.1 that for a sufficiently large degree, this reduced polynomial basis is a reduced Gröbner basis. First the reduced leading monomials $A^\star(d)$ are defined.

Definition 5.2 *Given a set of linearly independent leading monomials $A(d)$, then the set of reduced leading monomials $A^\star(d)$ is defined as the smallest subset of $A(d)$ for which each element of $A(d)$ is divisible by an element of $A^\star(d)$.*

Since there is a one-to-one mapping between leading monomials in $A(d)$ and polynomials of $R(d)$, each element of $A^\star(d)$ will also correspond with a polynomial.

Definition 5.3 *For a given canonical decomposition $A(d), B(d), R(d)$ the reduced polynomials $G(d)$ are defined as the polynomials of $R(d)$ corresponding to the reduced monomial system $A^\star(d)$:*

$$G(d) \;=\; \{r \in R(d) : \forall a \in A^\star(d),\; LM(r) = a\}.$$

Example 5.5 *The polynomial system from Example 5.2 had the canonical decomposition*

$$A(3) \;=\; \{x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^2x_2, x_1x_2^2, x_2^3\},$$

and

$$B(3) \;=\; \{1, x_1^3\}.$$

The reduced leading monomials for this decomposition are

$$A^\star(3) \;=\; \{x_1, x_2\},$$

since this subset is the smallest proper subset of monomials that divide all other monomials of $A(3)$. The corresponding reduced polynomials are

$$G(3) \;=\; \left\{ \begin{array}{rcl} x_1 - 2 & = & 0, \\ x_2 - 3 & = & 0. \end{array} \right.$$

The reduced leading monomials $A^\star(d)$ can be interpreted as a monomial system for which the Macaulay matrix can also be constructed. We will denote this matrix by $M_{A^\star}(d)$ and it is essential for defining the reduced normal set $B^\star(d)$.

Definition 5.4 *Let $A(d), B(d)$ be a canonical decomposition implied by $f_1, \ldots, f_s$ and a given monomial ordering. Then the reduced normal set $B^\star(d)$ is the normal set obtained from the canonical decomposition implied by $A^\star(d)$ and the same monomial ordering.*

Typically $B^\star(d) \subseteq B(d)$. The following example illustrates why this is the case.

Example 5.6 *The reduced monomial system $A^\star(3)$ of the canonical decomposition in Example 5.2 is*

$$A^\star(3) \;=\; \{x_1, x_2\}.$$

Its Macaulay matrix of degree 3 is

$$M_{A^\star}(3) \;=\; \begin{array}{c} \begin{array}{cccccccccc} 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 & x_1^3 & x_1^2x_2 & x_1x_2^2 & x_2^3 \end{array} \\ \left(\begin{array}{cccccccccc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right) \end{array},$$

which is almost the same as the matrix $A(3)$ except for the monomial x_1^3. This means that the reduced normal set is

$$B^\star(3) \;=\; \{1\},$$

compared to $B(3) = \{1, x_1^3\}$.

The property that the reduced normal set $B^\star(d) \subseteq B(d)$ holds in general. When constructing the Macaulay matrix of $A^\star(d)$ it is possible that columns corresponding to standard monomials $B(d)$ are filled. Hence these monomials will not be in $B^\star(d)$ anymore. Using the following lemma it is easy to determine the standard monomials from $M_{A^\star}(d)$.

Lemma 5.1 *Each standard monomial of $B^\star(d)$ derived from the Macaulay matrix $M_{A^\star}(d)$ always corresponds with a zero column of $M_{A^\star}(d)$.*

Proof. This follows trivially from the structure of the Macaulay matrix. □

Now a useful property on the zero-dimensionality of monomial ideals will be derived. First, the concept of a pure power is introduced.

Definition 5.5 *We call a monomial x_k^d $(1 \leq k \leq n)$ a pure power and denote the set of these n monomials by X_n^d.*

For example, $X_3^5 \;=\; \{x_1^5, x_2^5, x_3^5\}$. It is clear from the definition of the reduced leading monomials that if pure powers are present in $A(d)$ that they will also be present in $A^\star(d)$. The following lemma determines the growth of $B^\star(d)$ as a function of the degree d.

Lemma 5.2 *All monomials in n variables of degree*

$$d \geq d_{max} = n\,(d_0 - 1) + 1$$

can be written as a product of an element of $X_n^{d_0}$ with another monomial.

PROOF. The proof can be completely done in $\mathbb{N}_0^n$ since there is a bijection between the exponents of monomials and $\mathbb{N}_0^n$. We first show that for any degree $d < d_{\max}$, monomials can be found which cannot be written as a product of a pure power and another monomial. For degree $d_{\max} - 1 = n\,(d_0 - 1)$ we can write the following monomial

$$(d_0 - 1, d_0 - 1, \ldots, d_0 - 1), \tag{5.1}$$

which clearly cannot be written as a product of a pure power and another monomial. It's possible to come up with similar examples for all degrees between d_0 and $d_{\max} - 1$ by just subtracting the necessary amount of a component of (5.1). For degree $d_{\max} = n\,(d_0 - 1) + 1$ we can write the following monomial

$$(d_0, d_0 - 1, \ldots, d_0 - 1), \tag{5.2}$$

which is clearly the product of $x_1^{d_0}$ and $x_2^{d_0-1} \ldots\ x_n^{d_0-1}$. Any other monomial of degree $d_{\max}$ can now be formed by rearranging (5.2) (subtracting from one component and adding to another). If, however, one component is subtracted with a certain amount then the other components should be increased such that the sum of all components remains constant. From this it is easy to see that there will always be at least 1 component $\geq d_0$. □

Furthermore, the presence of a pure power for each variable in $A^\star(d)$ is a necessary condition for the finiteness of $B^\star(d)$. This is easily seen by an example. If there is no pure power for the variable x_1 in $A^\star(d)$, then all subsequent powers of x_1 will be zero columns in $M_{A^\star}(d)$ and $B^\star(d)$ will grow linearly.

A monomial system $A^\star(d)$ has a projective solution set because it is already homogeneous. It is shown in [22, p.452] that the dimension of this projective solution set is always one less than its affine solution set. Hence, if the monomial ideal has a finite number of affine roots it will have no projective roots whatsoever. We can now state the following theorem relating the zero-dimensionality of a monomial system to the presence of all pure powers.

Theorem 5.1 *A monomial system $A^\star(d)$ has m affine roots, counting multiplicities, if and only if it contains for each variable x_i $(1 \leq i \leq n)$ a pure power. It then also holds that for all degrees d larger than the degree of regularity:* $\dim \mathcal{B}_d^\star = m$.

PROOF. This follows from Lemma 5.1 and 5.2. □

From Theorem 3.5 we know that for a polynomial system with a finite number of projective roots, the nullity of $M(d)$, $c(d)$, will equal the total number of projective roots. Theorem 5.1 will allows us to separate the affine roots from the ones at infinity. In order to do this, the notion of a Gröbner basis will first need to be introduced in Chapter 6. In the same vein as $M_{A^\star}(d)$, the Macaulay matrix of the reduced polynomials $G(d)$ will be denoted $M_G(d)$.

5.2.2 Numerical Computation of $A^\star(d), B^\star(d)$ and $G(d)$

The definition of $A^\star(d)$ uses the complete set of linearly independent leading monomials $A(d)$. A straightforward way to find $A^\star(d)$ would hence be to compute $A(d)$ using Algorithm 5.1, find $A^\star(d)$ from $A(d)$ and select the corresponding polynomials of $R(d)$ to obtain $G(d)$. This is however not efficient since the whole canonical decomposition is computed while only subsets are required. By using the defining property of $A^\star(d)$, it is possible to adjust Algorithm 5.1 such that it directly computes $A^\star(d), B^\star(d)$ and $G(d)$. The whole procedure is summarized in pseudo-code in Algorithm 5.2.

Algorithm 5.2 *Computation of $A^\star(d), B^\star(d)$ and $G(d)$*
***Input**: polynomial system $f_1, \ldots, f_s$, degree d, tolerance τ*
***Output**: $A^\star(d), B^\star(d)$ and polynomials $G(d)$*

```
N ← orthogonal basis for null space of M(d) using Algorithm 4.2
A*(d), B*(d), G(d) ← ∅
𝒳 ← T_d^n
while 𝒳 ≠ ∅ do
  x^a ← smallest monomial in 𝒳 according to monomial ordering
  j ← vector of indices for B(d) and x^a
  [W S Z] ← SVD( N(j,:)^T )
  if arcsin(μ_m) < τ then
    append x^a to A*(d)
    remove x^a and all its monomial multiples from 𝒳
    append v_m^T to G(d)
  else
    append x^a to B*(d)
    remove x^a from 𝒳
  end if
end while
```

The algorithm iterates over a set of monomials $\mathcal{X}$ which is initially all monomials of degree 0 up to d. The key idea is that each monomial of $A(d)$ is a monomial multiple of a monomial of $A^\star(d)$. So as soon as a linearly independent leading monomial x^a is found, all its monomial multiples do not need to be checked anymore and can be removed from $\mathcal{X}$. When the monomial x^a is not linearly

independent, it is also removed from $\mathcal{X}$ and added to $B^\star(d)$. When $\mathcal{X}$ is empty, the algorithm terminates. Removing monomial multiples of x^a from $\mathcal{X}$ reduces the number of iterations significantly and also guarantees that the computed $B^\star$ is correct. The same arguments on the computational complexity apply as for Algorithm 5.1.

5.2.3 Numerical Experiments

Since Algorithm 5.2 is an adjustment of Algorithm 5.1 the same comments on numerical issues apply. We revisit the polynomial system of Example 5.4.

Example 5.7 *The linearly independent leading monomials $A(10)$ of the polynomial system*

$$\left\{\begin{array}{rcl} x_1^2 + x_1\,x_3 - 2\,x_2 + 5 & = & 0, \\ 2\,x_1^3\,x_2 + 7\,x_2\,x_3^2 - 4\,x_1\,x_2\,x_3 + 3\,x_1 - 2 & = & 0, \\ x_2^4 + 2\,x_2\,x_3 + 5\,x_1^2 - 5 & = & 0, \end{array}\right.$$

consists of 254 monomials. Running Algorithm 5.2 on the polynomial system results in the following reduced canonical decomposition:

$$\begin{aligned} A^\star(10) &= \{x_1\,x_3, x_1^3\,x_2, x_2^4, x_3\,x_2^3, x_3^3\,x_2, x_1^5, x_3^5\}, \\ B^\star(10) &= \{1, x_1, x_2, x_3, x_1^2, x_2\,x_1, x_2^2, x_2\,x_3, x_3^2, x_1^3, x_2\,x_1^2, x_2^2\,x_1, x_2^3, x_3\,x_2^2, x_2\,x_3^2 \\ &\quad\; x_3^3, x_1^4, x_2^2\,x_1^2, x_2^3\,x_1, x_3^2\,x_2^2, x_3^4, x_2^3\,x_1^2\}. \end{aligned}$$

$A^\star(10)$ consists of 7 monomials and the normal set $B(10)$ is reduced from 32 to 22 monomials. $G(10)$ consists of the following 7 polynomials

$$\left\{\begin{array}{l} 0.89803 - 0.35921\,x_2 + 0.17961\,x_1^2 + 0.17961\,x_1x_3 = 0, \\ \\ -0.085592 + 0.12839\,x_1 + 0.85592\,x_2 - 0.34237\,x_2^2 + 0.17118\,x_1^2x_2 \\ +0.29957\,x_2x_3^2 + \ 0.085592\,x_1^3x_2 = 0, \\ \\ -0.6742 + 0.6742\,x_1^2 + 0.26968\,x_2x_3 + 0.13484\,x_2^4 = 0, \\ \\ -0.025205 - 0.77127\,x_1 + 0.0040328\,x_2 + 0.49401\,x_3 + 0.023188\,x_1^2 \\ +0.31254\,x_1x_2 - 0.19156\,x_2x_3 + 0.0020164\,x_3^2 - 0.15627\,x_1^3 - 0.010082\,x_1^2x_2 \\ +0.0075614\,x_2^3 - 0.017643\,x_3^3 - 0.025205\,x_1^4 + 0.0010082\,x_1x_2^3 \\ +0.0010082\,x_2^3x_3 = 0, \end{array}\right.$$

$$
\begin{cases}
-0.089289 - 0.13951\,x_1 - 0.71432\,x_2 - 0.022322\,x_3 + 0.39064\,x_1x_2 \\
+0.31251\,x_2^2 - 0.16742\,x_2x_3 - 0.26787\,x_1^2x_2 - 0.15626\,x_1x_2^2 + 0.066967\,x_2^2x_3 \\
-0.27345\,x_2x_3^2 + 0.044645\,x_1^2x_2^2 + 0.078128\,x_2x_3^3 = 0, \\
\\
0.69381 - 0.57918\,x_2 + 0.0034475\,x_3 + 0.37578\,x_1^2 + 0.12066\,x_2^2 - 0.030166\,x_3^2 \\
-0.0086188\,x_1^3 - 0.15514\,x_1^2x_2 + 0.0017238\,x_2^3 + 0.047404\,x_1^4 - 0.0025856\,x_1x_2^3 \\
+0.0086188\,x_1^5 = 0, \\
\\
0.19201 + 0.74673\,x_1 - 0.062728\,x_2 - 0.4885\,x_3 + 0.025128\,x_1^2 - 0.30287\,x_1x_2 \\
-0.0059821\,x_2^2 + 0.19451\,x_2x_3 + 0.062794\,x_3^2 + 0.16707\,x_1^3 + 0.0079195\,x_1^2x_2 \\
+0.0018612\,x_1x_2^2 - 0.0070246\,x_2^3 - 0.0022368\,x_2^2x_3 + 0.0033854\,x_2x_3^2 \\
+0.0081701\,x_3^3 + 0.025733\,x_1^4 - 0.00070942\,x_1^2x_2^2 - 3.8079 \times 10^{-5}\,x_1x_2^3 \\
-0.0019462\,x_3^4 - 2.0865 \times 10^{-5}\,x_1^2x_2^3 + 0.0002556\,x_3^5 = 0.
\end{cases}
$$

We will show in Chapter 6 that $G(10)$ is a Gröbner basis.

5.3 Border Bases

As mentioned earlier, insisting that the monomials of $A(d)$ are leading monomials with respect to a monomial ordering unavoidably leads to the representation singularity. The concept of border bases resolves the representation singularity for polynomial systems with a finite number of affine roots. This is because border bases vary continuously under perturbations of the coefficients of the polynomial system. Before demonstrating this continuous change, we first define the border of a given reduced normal set $B^\star(d)$.

Definition 5.6 *For a given reduced normal set $B^\star(d)$, its border is*

$$\partial B^\star(d) \;=\; \{x_i\,b \mid 1 \le i \le n,\, b \in B^\star(d)\} \setminus B^\star(d).$$

Once the border of a reduced normal set $B^\star(d)$ is given, one can define the $B^\star(d)$-border prebasis.

Definition 5.7 *Let $B^\star(d)$ be a reduced normal set, then a $B^\star(d)$-border prebasis are the set of polynomials*

$$BB(d) \;=\; \{bb_j \mid bb_j = t_j - \sum_{i=1}^{m} \alpha_i\,b_i\ ,\ 1 \le j \le \mu\} \tag{5.3}$$

with $\partial B^\star(d) = \{t_1, \ldots, t_\mu\}$ and $B^\star(d) = \{b_1, \ldots, b_m\}$.

The polynomials of (5.3) are then a border basis for $B^\star(d)$ when they generate the polynomial ideal $\langle f_1, \ldots, f_s \rangle$ and the residue classes of the monomials in $B^\star(d)$ are a basis for the finite dimensional vector space $\mathcal{C}_d^n / I$. This can be summarized by a Buchberger's criterion for border bases [52, 53], or alternatively as commutation relations between multiplication matrices [65]. Algorithm 5.1 can be adapted in a straightforward manner to numerically compute the $B^\star(d)$-border prebasis $BB(d)$ for a given reduced normal set $B^\star(d)$. Since each polynomial of $BB(d)$ lies in $\mathrm{span}(B^\star(d), t)$, with $t \in \partial B^\star(d)$, one simply needs to replace the monomial x^a by a monomial $t \in \partial B^\star(d)$. The whole algorithm is summarized in pseudo-code in Algorithm 5.3.

Algorithm 5.3 *Computation of a $B^\star(d)$-border prebasis $BB(d)$*
***Input**: normal set $B^\star(d)$, tolerance τ*
***Output**: $B^\star(d)$-border prebasis $BB(d)$*

```
BB(d) ← ∅
∂B*(d) ← border monomials of B*(d)
for all t ∈ ∂B*(d) do
  j ← vector of indices for B*(d) and t
  [W S Z] ← SVD(N(j,:)^T)
  if arcsin(μ_m) < τ then
    append v_m^T to BB(d)
  end if
end for
```

Using Algorithm 5.3, we can now demonstrate that border basis avoid the representation singularity under perturbations of the coefficients of $f_1, \ldots, f_s$.

Example 5.8 *Consider the polynomial system ([55, p. 430])*

$$F \;=\; \left\{ \begin{array}{lcl} f_1 & = & \frac{1}{4}\, x_1^2 + x_2^2 - 1, \\ f_2 & = & x_1^2 + \frac{1}{4}\, x_2^2 - 1, \end{array} \right.$$

and its slightly perturbed version

$$\tilde{F} \;=\; \left\{ \begin{array}{lcl} \tilde{f}_1 & = & \frac{1}{4}\, x_1^2 + 10^{-5} x_1 x_2 + x_2^2 - 1, \\ \tilde{f}_2 & = & x_1^2 + 10^{-5} x_1 x_2 + \frac{1}{4}\, x_2^2 - 1. \end{array} \right.$$

Computing the reduced normal set $B^\star(3)$ for F using Algorithm 5.2 results in

$$B^\star(3) \;=\; \{1, x_1, x_2, x_1 x_2\}.$$

Applying Algorithm 5.3 and scaling the bb polynomials such that each leading term is monic results in the prebasis

$$BB(3) = \begin{cases} bb_1 &= -0.8000 + x_1^2, \\ bb_2 &= -0.8000 + x_2^2, \\ bb_3 &= -0.8000x_2 + x_1^2x_2, \\ bb_4 &= -0.8000x_1 + x_1\,x_2^2, \end{cases}$$

which is also a border basis for $\langle f_1, f_2\rangle$. *Now, applying Algorithm 5.3 for the perturbed polynomial system* $\tilde{F}$, *using the same reduced normal set* $B^\star(3)$, *returns the following prebasis*

$$\tilde{BB}(3) = \begin{cases} \tilde{bb}_1 &= -0.8 + 8\times 10^{-06}x_1x_2 + x_1^2, \\ \tilde{bb}_2 &= -0.8 + 8\times 10^{-06}x_1x_2 + x_2^2, \\ \tilde{bb}_3 &= 6.4\times 10^{-6}x_1 - 0.800\,x_2 + x_1^2x_2, \\ \tilde{bb}_4 &= -0.800\,x_1 + 6.4\times 10^{-6}\,x_2 + x_1\,x_2^2. \end{cases}$$

The -0.800 *terms in* $\tilde{bb}_3$ *and* $\tilde{bb}_4$ *have more nonzero digits, which is indicated by the trailing zeros. One can now see that the introduction of the noisy* x_1x_2 *term did not lead to any discontinuous jump from* $BB(3)$ *to* $\tilde{BB}(3)$. *The continuous change of the prebasis can be demonstrated by using symbolical computations. Replacing the coefficient of* x_1x_2 *in* $\tilde{F}$ *by* ϵ *and computing the prebases symbolically with respect to the degree negative lex ordering results in a prebasis*

$$BB(3) = \begin{cases} bb_1 &= -\frac{4}{5} + x_1^2, \\ bb_2 &= -\frac{4}{5} + x_2^2, \\ bb_3 &= -\frac{4}{5}x_2 + x_1^2x_2, \\ bb_4 &= -\frac{4}{5}x_1 + x_1x_2^2, \end{cases}$$

for F *and*

$$\tilde{BB}(3) = \begin{cases} bb_1 &= -\frac{4}{5} + \frac{4}{5}\epsilon x_1x_2 + x_1^2, \\ bb_2 &= -\frac{4}{5} + \frac{4}{5}\epsilon x_1x_2 + x_2^2, \\ bb_3 &= \frac{16\epsilon}{16\epsilon^2-25}x_1 - \frac{20}{16\epsilon^2-25}x_2 + x_1^2x_2, \\ bb_4 &= -\frac{20}{16\epsilon^2-25}x_1 + \frac{16\epsilon}{16\epsilon^2-25}x_2 + x_1x_2^2, \end{cases}$$

for $\tilde{F}$. *From these symbolic expressions it is seen that* $\tilde{BB}(3)$ *changes continuously into* $BB(3)$ *when* ϵ *goes to zero. Setting* $\epsilon = 10^{-5}$ *in these symbolic expressions results in the numerical prebases computed by Algorithm 5.3.*

We have seen that the computation of a canonical decomposition is ill-posed under perturbations of the coefficients of $f_1, \ldots, f_s$. Nonetheless, the reduced canonical

decomposition still allows to compute all affine roots of a polynomial system. Although it is guaranteed that the monomial sets $A(d), B(d)$ will change under perturbations of the coefficients, their cardinality however will not. This is due to the continuity of polynomial zeros [78, p. 304]. In other words, the total number of monomials in $B^\star(d)$ still represents the total number of affine roots for $d \geq d_G$. The importance of the canonical decomposition in root-finding and other applications will be further demonstrated in Chapter 6.

Chapter 6

Applications

The focus of this chapter is solving problems. In particular, we develop and implement algorithms for solving the following problems: finding a Gröbner basis, affine root-finding, solving the ideal membership problem, multivariate polynomial elimination, doing the syzygy analysis, finding least common multiples and greatest common divisors of two multivariate polynomials and the removal of multiplicities of roots. Where applicable, we will show how in the PNLA framework solving these problems can be done without the computation of a Gröbner basis.

6.1 Gröbner basis

In this section, the link is made between the reduced polynomials $G(d)$ from the reduced canonical decomposition and a Gröbner basis of the ideal $\langle f_1, \ldots, f_s \rangle$. This will lead to some insights on the separation of the roots of a polynomial system into an affine part and roots at infinity for the zero-dimensional case. A condition will be derived for this case to determine the affine part of the normal set. One can think of a Gröbner basis as another set of generators for the ideal $\langle f_1, \ldots, f_s \rangle$. It is a classical result that for each ideal $\langle f_1, \ldots, f_s \rangle$, there exists such a finite set of polynomials G [21, 22]. The finiteness of G relies on Hilbert's Basis Theorem [45]. This implies that there exists a particular degree d for which $G \in \mathcal{M}_d$, which leads to the following problem.

Problem 6.1 *Find for a multivariate polynomial system $f_1, \ldots, f_s$ the degree d_G such that for all $d \geq d_G : \ G \in \mathcal{M}_d$.*

The degree d_G is for the general case related to the Gröbner basis bound G_b. This bound is the least value such that for all $f_1, \ldots, f_s$ there is a Gröbner basis G whose members are polynomials of degree at most G_b. Bounds for G_b are related to those for I_b and S_b and are therefore also doubly exponential. Lazard proved the following useful theorem for the practical case of zero-dimensional projective varieties.

Theorem 6.1 *([56, p.154-155]) Let $f_1, \ldots, f_s$ be multivariate polynomials of degrees $d_1, \ldots, d_s$ such that $d_1 \geq d_2 \ldots \geq d_s$. Suppose that a multiplicative monomial ordering is used and that the homogenized polynomial system $f_1^h, \ldots, f_s^h$ has a finite number of nontrivial projective roots. Then the polynomials of the reduced Gröbner basis have degrees at most $d_1 + \ldots + d_{n+1} - n + 1$ with $d_{n+1} = 1$ if $s = n$.*

This theorem provides a nice linear bound on the maximal degrees of the Gröbner basis. Unfortunately, this does not imply that $d_G \leq d_1 + \ldots + d_{n+1} - n + 1$ since $\mathcal{M}_d$ does not necessarily contain all polynomials of $\langle f_1, \ldots, f_s \rangle$ of degree d.

The reduced polynomials $G(d)$ computed from Algorithm 5.2 ensure by definition that

$$\forall\, p \in \mathcal{M}_d\ \exists\, g \in G(d) \text{ such that } \mathrm{LM}(g) \mid \mathrm{LM}(p).$$

This suggests that $G(d)$ is a Gröbner basis when $d \geq d_G$. Furthermore, it will be a reduced Gröbner basis. A criterion is needed to be able to decide whether $G(d)$ is a Gröbner basis. This is given by Buchberger's criterion, which we formulate in terms of the Macaulay matrix $M(d)$ and the reduced monomial system $A^\star(d)$.

Theorem 6.2 (Buchberger's Criterion) *Let $f_1, \ldots, f_s$ be a multivariate polynomial system with reduced monomial system $A^\star(d)$ and reduced polynomials $G(d)$ for a given degree d. Then $G(d)$ is a Gröbner basis for $\langle f_1, \ldots, f_s \rangle$ if $M(d^\star)$ has the same reduced leading monomials $A^\star(d)$ for a degree $d^\star$ such that all S-polynomials of $G(d)$ lie in $\mathcal{M}_{d^\star}$.*

Proof. Saying that $M(d^\star)$ has the same reduced leading monomials $A^\star(d)$ is equivalent with saying that all S-polynomials have a zero remainder on division by $G(d)$. This is exactly the stop-criterion for Buchberger's Algorithm. □

Buchberger's Criterion implies that for all degrees $d \geq d_G$, the Macaulay matrix $M_G(d)$ has the same reduced canonical decomposition as $M_{A^\star}(d)$. This implies that for all degrees $d \geq d_G$, the reduced canonical decomposition will not change anymore. We therefore have the following useful corollary.

Corollary 6.1 *Let $f_1, \ldots, f_s$ be a multivariate polynomial system with a finite number of affine roots. Then $\forall d \geq d_G$ its reduced monomial set $A^\star(d)$ will contain*

for each variable $x_i\,(1 \leq i \leq n)$ *a pure power. Furthermore,* $B^\star(d)$ *is then the affine normal set.*

PROOF. This follows from Theorem 5.1 and Buchberger's Criterion that $\forall\, d \geq d_G$ both $M_G(d)$ and $M_{A^\star}(d)$ have the same reduced monomial decomposition. □

If it is known that the affine solution set of a polynomial ideal is zero-dimensional, then detecting pure powers in $A^\star(d)$ allows to determine the degree d_G and Gröbner basis $G(d)$. This is summarized in Algorithm 6.1. In general, in going from $A^\star(d)$ to $A^\star(d+1)$, monomials are both removed and added. When a monomial x^α from $A^\star(d)$ is not present anymore in $A^\star(d+1)$, then this implies that $x^\alpha \in B^\star(d+1)$. It has, in other words, become a linearly dependent leading monomial. One therefore always needs to recompute the reduced canonical decomposition for each degree. Once d_G is known, it then becomes possible to numerically compute all affine roots by solving an eigenvalue problem without the explicit computation of a Gröbner basis. This will be further described in Section 6.2.

Algorithm 6.1 *Computation of Gröbner basis* G *of* $\langle f_1, \ldots, f_s \rangle$
Input*:* $f_1, \ldots, f_s$ *with finite number of nontrivial projective roots*
Output*: Gröbner basis* G

```
d ← max(deg(f_1), deg(f_2), ..., deg(f_s))
A*(d), B*(d), G(d) ← ∅
pp ← 0
while pp ≠ 0 do
   A*(d), B*(d), G(d) ← reduced canonical decomposition (Algorithm 5.2)
   if all pure powers in A*(d) then
      pp ← 1
      G ← G(d)
   else
      d ← d + 1
   end if
end while
```

Example 6.1 *Again, we revisit the polynomial system in* $\mathcal{C}_4^3$

$$\left\{\begin{array}{rcl} x_1^2 + x_1\,x_3 - 2\,x_2 + 5 &=& 0, \\ 2\,x_1^3\,x_2 + 7\,x_2\,x_3^2 - 4\,x_1\,x_2\,x_3 + 3\,x_1 - 2 &=& 0, \\ x_2^4 + 2\,x_2\,x_3 + 5\,x_1^2 - 5 &=& 0, \end{array}\right.$$

from Example 5.4. For this polynomial system $s = n = 3$ *and therefore* $d_1 + d_2 + d_3 - n + 1 = 8$. *This polynomial system has a zero-dimensional solution set and, following Algorithm 6.1, we start to compute the reduced canonical decomposition for* $d = 4$. *Algorithm 5.2 returns*

$$A^\star(4) = \{x_1\,x_3, x_1^3\,x_2, x_2^4\},$$

which already contains 1 pure power: x_2^4. The next pure power, x_1^5, is retrieved for $d = 7$ in

$$A^\star(7) = \{x_1\,x_3, x_1^3\,x_2, x_2^4, x_2\,x_3^3, x_1^5\}.$$

The last pure power, x_3^5, is found for $d = d_G = 10$:

$$A^\star(10) = \{x_1\,x_3, x_1^3\,x_2, x_2^4, x_2\,x_3^3, x_1^5, x_3^5\}.$$

The Gröbner basis is therefore $G(10)$ as given in Example 5.7. Indeed, computing an exact Gröbner basis in Maple and normalizing each polynomial results in $G(10)$. Although the maximal degree of the polynomials of G is $5 < 8$, which is in agreement with Theorem 6.1, the Gröbner basis is found only for $d = 10$.

The ill-posedness of the canonical decomposition under the influence of noise directly affects the computation of a Gröbner basis. As shown by Nagasaka in [67], it is impossible to define an approximate Gröbner basis in the same sense as an approximate GCD or approximate factorization of multivariate polynomials. In addition, Gröbner basis polynomials typically have large integer coefficients. It is even possible that these coefficients fall out of the range of the double precision standard. In this case, it would be necessary to perform the computations in higher precision.

The importance of the Gröbner basis and the degree d_G for which it can be found from the Macaulay matrix is demonstrated by the following theorem. This theorem solves Problem 3.3 of finding the degree d_S for which all basis syzygies are found by inspecting linearly dependent rows of $M(d)$ as described in Algorithm 3.1.

Theorem 6.3 *Consider the problem of finding all basis syzygies as described in Problem 3.3. Let $G = \{g_1, \ldots, g_t\}$ be a Gröbner basis of $\langle f_1, \ldots, f_s\rangle$ and*

$$d_0 \;=\; \max_{i \neq j} deg(S(g_i, g_j)). \tag{6.1}$$

Then

$$d_S \;\leq\; d_G + d_0.$$

Proof. It is a well-known result that all basis syzygies of $\langle f_1, \ldots, f_s\rangle$ can be determined from a Gröbner basis [21, p. 223]. Indeed, the reduction to zero of every S-polynomial of a pair of polynomials in a Gröbner basis provides a basis syzygy. This implies that it is required to construct $M(d)$ for a degree which contains all these S-polynomials, which leads to (6.1). □

Observe that the proof relies on the notion of a Gröbner basis. The following example illustrates the application of Theorem 6.3.

Example 6.2 *We reconsider the polynomial system in C^4 from Example 3.13*

$$\begin{cases} f_1: & x_2^2\,x_3 + 2\,x_1\,x_2\,x_4 - 2\,x_1 - x_3 = 0, \\ f_2: & -x_1^3\,x_3 + 4\,x_1\,x_2^2\,x_3 + 4\,x_1^2\,x_2\,x_4 + 2\,x_2^3\,x_4 + 4\,x_1^2 - 10\,x_2^2 \\ & +4\,x_1\,x_3 - 10\,x_2\,x_4 + 2 = 0, \\ f_3: & 2\,x_2\,x_3\,x_4 + x_1\,x_4^2 - x_1 - 2\,x_3 = 0, \\ f_4: & -x_1\,x_3^3 + 4\,x_2\,x_3^2\,x_4 + 4\,x_1\,x_3\,x_4^2 + 2\,x_2\,x_4^3 + 4\,x_1\,x_3 \\ & +4\,x_3^2 - 10\,x_2\,x_4 - 10\,x_4^2 + 2 = 0, \end{cases}$$

with degrees $d_1 = d_3 = 3$, $d_2 = d_4 = 4$. For a degree $d = 11$, the reduced leading monomials $A^\star(11)$ contain 31 monomials. More importantly, among those 31 monomials we have the following pure powers

$$x_1^7, x_2^5, x_3^4, x_4^4.$$

The presence of a pure power for each variable implies that $d_G = 11$ and hence the Gröbner basis $G(11)$ of the polynomial system consists of 31 polynomials. The maximum degree of all S-polynomials $S(g_i, g_j)$ is 12. By applying Theorem 6.3 we know that the degree d_S for which all basis syzygies are guaranteed to be found is $d_G + 12 = 23$. From Example 3.13 we know that all basis syzygies are in fact found for $d = 15$. Theorem 6.3 therefore provides a bound on d_S that can be quite an overestimation. Furthermore, the theorem explicitly requires the computation of a Gröbner basis because all its S-polynomials need to be computed as well. This is quite inpractical and an alternative syzygy analysis that does not require the computation of a Gröbner basis will be provided in Section 6.4 of this chapter.

6.2 Affine root-finding

In this section it will be shown how the affine roots of a multivariate polynomial system $f_1, \ldots, f_s$ can be numerically computed from either a standard or generalized eigenvalue problem. This procedure is described in much more detail in [32]. The key ingredients for affine root-finding are Definition 3.5, which states that the canonical kernel K of $M(d)$ consists of linear combinations of the partial differential functionals $\partial_j|_z$, and Corollary 6.1, which states that the reduced normal set $B^\star(d)$ counts the total number of affine roots for $d \geq d^\star$. For this purpose, we will assume that the polynomial system $f_1, \ldots, f_s$ has a zero-dimensional affine variety: m roots $z_1, \ldots, z_m$ with no multiplicities. No assumptions on the roots are infinity are required. Indeed, it is even allowed that the part of the variety at infinity is nonzero-dimensional. First notice that the no-multiplicities assumption and Definition 3.5 implies that the canonical basis

for the null space K consists only of $\partial_0|_z$ functionals. We further introduce the column partitioning

$$K \;=\; \begin{pmatrix} K_a & K_\infty \end{pmatrix},$$

where K_a contains all functionals corresponding with affine roots and K_∞ consists of the functionals corresponding with roots at infinity. Throughout this whole section, the non-homogeneous interpretation ($x_0 = 1$) of both $\mathcal{M}_d$ and $\mathcal{C}_d^{n'}$ is used. Observe now that for a functional $\partial_0|_z$, with $z = (x_1, \ldots, x_n)$ an affine root, the following relationship holds:

$$\begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n^{d-1} \end{pmatrix} x_1 \;=\; \begin{pmatrix} x_1 \\ x_1^2 \\ x_1 x_2 \\ \vdots \\ x_1 x_n^{d-1} \end{pmatrix}. \tag{6.2}$$

Or in other words, the operation of multiplying $\partial_0|_z$ with x_1 corresponds with a particular row selection of the same functional. In this case, the first row becomes the second, the second is mapped to row $n+2$, and so forth. If we want to express the multiplication of functionals in $\mathcal{C}_d^{n'}$ with monomials of degree 1, then only the rows corresponding with monomials up to degree $d-1$ are allowed to be multiplied. Indeed, monomials of degree d would be 'shifted' out of the coefficient vector. Hence (6.2) can be rewritten as

$$S_1\, \partial_0|_z\, x_1 \;=\; S_{x_1}\, \partial_0|_z, \tag{6.3}$$

where S_1 selects at most all $\binom{d-1+n}{n}$ rows corresponding with monomials from degree 0 up to $d-1$ and S_{x_1} selects the corresponding rows after multiplication with x_1. Later on, the selection matrices S_1, S_{x_1} will need to be further constrained to arrive at an eigenvalue problem.

Example 6.3 *Writing down* (6.3) *for functionals in* $\mathcal{C}_2^{2'}$ *results in*

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \end{pmatrix} x_1 \;=\; \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \end{pmatrix},$$

where the selection matrix S_1 *selects, in this case, all rows corresponding with all monomials of degree 0 up to 1.*

Remark 6.1 *If* S_1 *in Example 6.3 would have selected any of the rows corresponding with monomials of degree 2, then no corresponding* S_{x_1} *could have been constructed since the functionals do not contain any monomials of degree 3.*

Observe that relations similar to (6.3) can be written down for multiplication with any variable x_i. Indeed, for every variable x_i a corresponding row selection matrix S_{x_i} can be derived. Under the assumption that none of the m affine roots has multiplicities, (6.2) can be extended to all functionals of affine roots $K_a = (\partial_0|_{z_1} \quad \dots \quad \partial_0|_{z_m})$ and any multiplication variable x_i so that we can write

$$S_1 \, K_a \, D_{x_i} = \, S_{x_i} \, K_a, \tag{6.4}$$

where D_{x_i} is a square diagonal matrix containing x_i's. Now, it will be shown how (6.4) can be written as a standard or generalized eigenvalue problem. The matrix $q \times m$ matrix K_a cannot be directly computed from $M(d)$. It is possible however, to compute a numerical basis N for $\text{null}(M(d))$, using our recursive orthogonalization algorithm. Since both N and K are bases for $\text{null}(M(d))$, they are related by a nonsingular matrix T, or in other words, $K = N\,T$. So, if it were possible to write $K_a = N\,T_a$, one could substitute K_a in (6.4) and obtain

$$S_1 \, N \, T_a \, D_{x_1} \; = \; S_{x_i} \; N \, T_a.$$

Setting $B = S_1 \, N$ and $A = S_{x_i} \, N$, we would then have $B \, T_a \, D_{x_i} = A \, T_a$, which looks like a generalized eigenvalue problem. It is not however, since A, B and, more importantly, T_a are not square. We will now show how through a particular basis change, also called a column compression, the matrices can be made square. The first step is to partition the rows of N into

$$\begin{array}{c} \\ k \\ q-k \end{array} \begin{array}{c} c \\ \begin{pmatrix} N_1 \\ N_2 \end{pmatrix} \end{array}$$

such that $\text{rank}(N_1) = m$. Suppose also that we have orthogonal matrices Q_1, Q_2 such that $\text{col}(Q_1) = \text{col}(N_1)$ and $\text{col}(Q_2) = \text{null}(N_1)$. These can again be computed from either a rank-revealing QR or SVD using Algorithm 4.2. The row partitioning of N implies then the same partitioning of the rows of K:

$$K \; = \; \begin{array}{c} \\ k \\ q-k \end{array} \begin{array}{c} \begin{array}{cc} m & c-m \end{array} \\ \begin{pmatrix} K_{a1} & K_{\infty 1} \\ K_{a2} & K_{\infty 2} \end{pmatrix} \end{array}$$

Now all ingredients are in place to transform (6.4) into a standard eigenvalue problem. First, the following linear change of basis vectors of N is performed

$$Z = N \, Q \; = \; \begin{array}{c} \\ k \\ q-k \end{array} \begin{array}{c} c \\ \begin{pmatrix} N_1 \\ N_2 \end{pmatrix} \end{array} \quad \begin{array}{c} \begin{array}{cc} m & c-m \end{array} \\ \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \end{array}.$$

Since Q_2 is orthogonal to N_1, Z is

$$Z = \begin{array}{c} \\ k \\ q-k \end{array} \begin{array}{c} \begin{array}{cc} m & c-m \end{array} \\ \begin{pmatrix} N_1 Q_1 & 0 \\ N_2 Q_1 & N_2 Q_2 \end{pmatrix} \end{array}.$$

The reason this change of basis is also called a column compression is because $N_1 Q_2 = 0$ and hence $c - m$ zero columns are introduced in the upper k rows of Z. Also, since $K = N\,T$ and $Z = N\,Q$, then there is a nonsingular $V = Q^{-1}\,T$ such that $K = Z\,V$, which can be written using the 2-by-2 block partitioning

$$\begin{array}{c} \\ k \\ q-k \end{array} \begin{array}{c} \begin{array}{cc} m & c-m \end{array} \\ \begin{pmatrix} N_1 Q_1 & 0 \\ N_2 Q_1 & N_2 Q_2 \end{pmatrix} \end{array} \begin{array}{c} \begin{array}{cc} m & c-m \end{array} \\ \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \end{array} = \begin{pmatrix} K_{a1} & K_{\infty 1} \\ K_{a2} & K_{\infty 2} \end{pmatrix}.$$

From this we see that $K_{a1} = N_1 Q_1 V_1 1$. Hence, if the total number of rows k and the row selection matrices S_1, S_{x_i} are chosen such that we can write

$$S_1\ K_{a1}\ D_{x_i} = \ S_{x_i}\ K_{a1},$$

then by substituting K_{a1} we get

$$S_1\ N_1\ Q_1\ V_{11}\ D_{x_i} = \ S_{x_i}\ N_1\ Q_1\ V_{11}. \tag{6.5}$$

If we now set $B = S_1\,N_1\,Q_1$ and $A = S_{x_i}\,N_1\,Q_1$, then (6.5) can be written as a standard eigenvalue problem

$$V_{11}\ D_{x_i}\ V_{11}^{-1}\ = \ B^{\dagger} A, \tag{6.6}$$

since V_{11} is now square. Once the eigenvectors V_{11} are computed, then K_{a1} can be computed from $K_{a1} = N_1 Q_1 V_1 1$. Scaling each column of K_{a1} such that its first element is 1 allows then to read off each affine root.

Remark 6.2 *If the polynomial system $f_1, \ldots, f_s$ has no roots at infinity, then no column compression is required. Indeed, for this case $c = m$ and $K = K_a$ so that substituting $K = N\,T$ into* (6.4) *directly results in an eigenvalue problem.*

Remark 6.3 *The condition that rank$(N_1) = m$ is crucial since this ensures that $c - m$ columns of N_1 are zeroed out. It is this that allows us to write* (6.4) *as an eigenvalue problem.*

Remark 6.4 *If the row selection matrix S_1 is chosen such that S_{x_i} selects rows from K_{a2}, then* (6.4) *would be*

$$S_1\ K_{a1}\ D_{x_i}\ = \ \begin{pmatrix} S_{x_i 1} & S_{x_i 2} \end{pmatrix} \begin{pmatrix} K_{a1} \\ K_{a2} \end{pmatrix},$$

which cannot be written as an eigenvalue problem.

A proper choice of S_1, S_{x_i} such that A, B are square matrices enables us to write (6.5) as a generalized eigenvalue problem

$$B\ V_{11}\ D_{x_1}\ = \ A\ V_{11}, \tag{6.7}$$

and no pseudoinverse is needed anymore. From the reduced canonical decomposition we know that each element of $B^\star(d)$ corresponds with a linearly dependent column of $M(d)$. The duality between linearly dependent columns of $M(d)$ and linearly independent rows of N implies that each row of N corresponding with an element of $B^\star(d)$ is linearly independent. From Corollary 6.1 we also know that $|B^\star(d)| = m$. Hence, by setting S_1 such that it selects the rows corresponding with the monomials of $B^\star(d)$ we are sure that B is a $m \times m$ matrix of rank m. In addition, the monomials of $B^\star(d)$ are of minimal total degree, which means that there is no danger of rows 'falling out of K' after multiplying with the monomial x_i. As soon as all rows selected by S_{x_i} are also in K_{a1}, one can solve (6.7) and compute K_{a1} to find all affine roots. In order to make sure that all rows selected by S_{x_i} lie in K_{a1}, some extra iterations of the canonical decomposition are required. This follows from the following lemma.

Lemma 6.1 *Let b be a monomial of $B(d)$ that does not lie in $B^\star(d)$ and of minimal multidegree and $b^\star$ the monomial of $B^\star(d)$ with highest multidegree. If $deg(b) > deg(b^\star) + 1$, then for $k = \binom{deg(b^\star)+n}{n}$ S_{x_i} will only select rows of K_{a1}.*

Proof. Let j be the vector of indices of all monomials from 1 up to b and i the same as j but with the last index removed. Then due to the duality argument, $\text{rank}(N(j,:)) = m + 1$ and $\text{rank}(N(i,:)) = m$. So we can set $K_{a1} = N(i,:)\, Q_1\, V_{11}$. If $\deg(b) > \deg(b^\star) + 1$, then A, B are guaranteed to lie in K_{a1}. □

The whole affine root-finding procedure is summarized in Algorithm 6.2. One of the most important steps is the computation of the reduced canonical decomposition. Indeed, $A^\star(d)$ allows to detect whether $d = d_G$, since from this degree the computed basis for the kernel will contain all functionals of affine roots. Some extra iterations of the canonical decomposition algorithm are necessary to determine the monomial b. If then $\deg(b) > \deg(b^\star) + 1$ is true, the column compression is performed and the affine roots are computed from either a standard or generalized eigenvalue problem.

Remark 6.5 *Instead of computing K from $Z\,V$, it would also be possible to solve n eigenvalue problems for each component x_i $(1 \leq i \leq n)$. Then one would also need to 'match' each of the components x_i for each affine root by doing an exhaustive search.*

Remark 6.6 *In computer algebra, the affine roots of a polynomial system $f_1, \ldots, f_s$ are found by first computing a Gröbner basis G of $\langle f_1, \ldots, f_s \rangle$. From this Gröbner basis G a normal set B and multiplication matrices $M_{x_1}, \ldots, M_{x_n}$ are determined. Each of the components of the affine roots are then retrieved as the eigenvalues of these multiplication matrices. In the PNLA framework, no explicit computation of a Gröbner basis is necessary. Instead, all information to retrieve*

all affine roots is directly determined from the reduced leading monomials $A^\star(d)$ and reduced normal set $B^\star(d)$.

Algorithm 6.2 *Affine Root-Finding*
Input: *polynomials $f_1, \ldots, f_s \in \mathcal{C}_d^n$ with m distinct affine roots*
Output: *affine roots $z_1, \ldots, z_m$*

```
d ← max(deg(f_1), deg(f_2), ..., deg(f_s))
pp ← 0
N ← orthogonal basis for null(M(d)) (Algorithm 4.2)
while pp ≠ 0 do
  A*(d), B*(d) ← reduced canonical decomposition (Algorithm 5.2)
  if all pure powers in A*(d) then
    b* ← monomial of B*(d) with highest multidegree
    b ← extra iterations of Algorithm 5.1 to find next element of B(d)
    if deg(b) > deg(b*) + 1 then
      pp ← 1
      Z ← column compression of N with k = C(deg(b*)+n, n)
      B, A ← row selections of Z
      [V, D] ← eig(A, B) or eig(B^† A)
      K ← Z V
      z_1, ..., z_m ← affine roots from normalized K
    end if
  else
    d ← d + 1
    update N using Algorithm 4.2
  end if
end while
```

Example 6.4 *Algorithm 6.2 is applied to the polynomial system in $\mathcal{C}_4^3$*

$$\left\{\begin{array}{rcl} x_1^2 + x_1\,x_3 - 2\,x_2 + 5 &=& 0, \\ 2\,x_1^3\,x_2 + 7\,x_2\,x_3^2 - 4\,x_1\,x_2\,x_3 + 3\,x_1 - 2 &=& 0, \\ x_2^4 + 2\,x_2\,x_3 + 5\,x_1^2 - 5 &=& 0. \end{array}\right.$$

We know from Example 6.1 that $d_G = 10$. For this degree, we have that $b^\star = x_1^2 x_2^3$ and $b = x_2^3 x_3^2$ and hence $deg(b) = deg(b^\star)$. The total number of affine roots can already be deduced from $B^\star(10)$ to be 22. For $d = 11$ we now have that $b^\star = x_1^2 x_2^3$ and $b = x_1^4 x_2^3$ and $deg(b) = 7 > deg(b^\star) = 5 + 1$. We can therefore set $k = \binom{6+3}{3} = 84$ and construct the 22×22 matrices A, B. Using the eigenvectors V_{11}, the kernel

K_{a1} is computed, from which the 22 affine roots

$$\begin{array}{rrrr}
(& -0.09 \mp 1.03\,i, & 1.76 \pm 0.024\,i, & 0.16 \mp 0.40\,i \quad) \\
(& 0.11 \mp 1.037\,i, & 1.85 \pm 0.072\,i, & -0.37 \mp 0.18\,i \quad) \\
(& -0.77 \pm 1.55\,i, & 0.42 \mp 1.86\,i, & -0.08 \pm 1.55\,i \quad) \\
(& 1.47 \pm 1.69\,i, & 0.55 \pm 2.17\,i, & -1.15 \pm 0.89\,i \quad) \\
(& 0.51 \mp 1.79\,i, & 0.33 \mp 2.29\,i, & 1.22 \mp 1.12\,i \quad) \\
(& -0.98, & 0.018, & 6.05 \quad) \\
(& -0.36 \mp 2.17\,i, & -2.42 \pm 0.35\,i, & 0.79 \mp 2.28\,i \quad) \\
(& 1.36 \mp 2.76\,i, & -2.44 \mp 0.61\,i, & -2.42 \mp 0.30\,i \quad) \\
(& -0.0032 \pm 2.91\,i, & -0.13 \mp 2.70\,i, & -1.85 \mp 1.093\,i \quad) \\
(& 0.99, & -0.0035, & -6.02 \quad) \\
(& -7.47 \mp 0.63\,i, & -2.78 \mp 2.73\,i, & 8.94 \pm 1.23\,i \quad) \\
(& -5.74 \mp 1.25\,i, & 2.85 \mp 2.48\,i, & 5.81 \pm 2.10\,i \quad)
\end{array}$$

are read off. Notice that all 20 complex roots come in conjugate pairs and that there are only 2 real roots.

Example 6.5 *This example illustrates the affine root-finding algorithm on the small LTI system identification problem of Chapter 1 Section 1.1. The model parameters of the following Output-Error model*

$$y(t) \;=\; \frac{0.2q^{-1}}{(1 - 1.6q^{-1} + .89q^{-2})}u(t) + e(t), \tag{6.8}$$

are estimated. This corresponds with solving a polynomials system of 7 polynomials in 7. In this example

$$\begin{array}{rcl}
x_1 & = & b_1, \\
x_2 & = & f_1, \\
x_3 & = & f_2, \\
x_4 & = & \lambda_1, \\
x_5 & = & \lambda_2, \\
x_6 & = & \lambda_3, \\
x_7 & = & \lambda_4.
\end{array}$$

The reduced monomial system $A^\star(9)$ contains all pure powers. This is well below the upper bound from Theorem 6.1, which evaluates to 13*. Indeed, the upper bound $d_1 + \ldots + d_{n+1} - n + 1$ is in practice very pessimistic. The Macaulay matrix $M(9)$ is a 16731 by 11440 matrix with a density of* 0.077%*. The pure powers are $\{b_1^4, f_1^3, f_2^4, \lambda_1^3, \lambda_2^3, \lambda_3^3, \lambda_4\}$. The affine solution set consists of 43 solutions of a total of 801. Only 7 of the 43 are real. The solution that minimizes the cost function*

$$V = \frac{1}{12}\sum_{t=1}^{6} e(t)^2,$$

is given by

$$\begin{aligned} b_1 &= 0.2174, \\ f_1 &= -1.5738, \\ f_2 &= 0.8506, \\ \lambda_1 &= 0.0004, \\ \lambda_2 &= -0.0009, \\ \lambda_3 &= -0.0099, \\ \lambda_4 &= 0.0180. \end{aligned}$$

with $V = 0.00822$. *The model corresponding with the global minimum is hence given by*

$$y(t) = \frac{0.2174\, q^{-1}}{(1 - 1.5738\, q^{-1} + 0.8506\, q^{-2})} u(t) + e(t). \tag{6.9}$$

The 6 remaining affine solutions correspond with non-stable solutions and therefore the MATLAB System Identification toolbox returns exactly the same result. This confirms that the proposed method solves the optimization problem (1.4) *as described in [59].*

Example 6.6 *In this example we solve the polynomial system to find the maximum likelihood estimates of the mixing probabilities* x_1, x_2, x_3 *from Chapter 1 Section 1.1. After the elimination of* x_3*, the following polynomial system in 2 unknowns and of degree 3 remains:*

$$\begin{cases} -0.0289\, x_1 + 0.0047\, x_2 + 0.00396\, x_1^3 - 0.0000136\, x_2^3 + 0.0120 - 0.00131\, x_1^2 \\ + 0.000378 x_1\, x_2 - 0.0000357\, x_2^2 - 0.00183\, x_1^2\, x_2 + 0.000276\, x_1\, x_2^2 = 0, \\ \\ 0.0047\, x_1 - 0.0008\, x_2 - 0.00062\, x_1^3 + 0.000002\, x_2^3 - 0.00187 + 0.00017\, x_1^2 \\ -0.000056\, x_1\, x_2 + 0.000006\, x_2^2 + 0.00028\, x_1^2\, x_2 - 0.000041\, x_1\, x_2^2 = 0. \end{cases}$$

All pure powers are found in $A^\star(5)$*, they are* x_1^5 *and* x_2^3*. One can deduce that the polynomial system has no roots at infinity since* $c(5) = 9$ *and 9 affine roots are found. The constraint that the unknown* x*'s are probabilities limits their allowed numerical value to* $0 \leq x_1, x_2 \leq 1$*. The only solution that satisfies this constraint is* $x_1 = 0.519, x_2 = 0.217$*, which implies that* $x_3 = 0.264$*. The mixing probability* x_1 *for the observed DNA bases to be drawn from the CG rich distribution is twice as big as the other mixing probabilities. Once can therefore conclude that the observed DNA sequence is more likely to come from a CpG island.*

6.3 Ideal Membership problem

As already shown in Chapter 3, solving the ideal membership problem for a non-homogeneous polynomial p is checking the equality

$$\text{rank}(\begin{pmatrix} M(d) \\ p \end{pmatrix}) = \text{rank}(M(d)), \tag{6.10}$$

for a sufficiently large degree d_I. We can now express d_I in terms of d_G in the following way.

Theorem 6.4 *Consider the ideal membership problem as described in Problem 3.1. Let $G = \{g_1, \ldots, g_k\}$ be a Gröbner basis of $\langle f_1, \ldots, f_s \rangle$ and*

$$G_p = \{g \in G : LM(g) \mid LM(p)\} \text{ and } d_0 = \max_{g \in G_p} deg(g).$$

Then

$$d_I = d_G + deg(p) - d_0. \tag{6.11}$$

Proof. Since G is a Gröbner basis $\exists\, g \in G : \text{LM}(g) \mid \text{LM}(p)$ and G_p is therefore never empty. Determining whether $p \in \langle f_1, \ldots, f_s \rangle$ is equivalent with checking whether the remainder of p on division by G is zero. The determination of this remainder is equivalent with the reduction of the matrix

$$\begin{pmatrix} M(d) \\ p \end{pmatrix} Q$$

to triangular form for a sufficiently large d with Q the right-to-left column permutation as described in Section 5.1. Suppose that $g \in G_p$ and $\deg(g) = d_0$. The degree d_I as in (6.11) is then such that it guarantees that $\frac{\text{LM}(p)}{\text{LM(g)}}\, g \in \mathcal{M}_{d_I}$. In the first division of the multivariate division algorithm to compute the remainder, p will be updated to

$$p \leftarrow p - \frac{\text{LM}(p)}{\text{LM(g)}}\, g.$$

The multivariate division algorithm guarantees that the new p will have a smaller multidegree (according to the monomial ordering) [22, p.65]. In the next division step, another $g \in G$ such that $\text{LT}(g)|\text{LT}(p)$ is required. Since p has a smaller multidegree, the new g is also guaranteed to lie in $\mathcal{M}_{d_I}$. Therefore, all remaining steps of the division algorithm can be performed within $\mathcal{M}_{d_I}$ and the ideal membership problem can be solved. □

Remark 6.7 *In computer algebra, the ideal membership problem is solved by computing a Gröbner basis G for $f_1, \ldots, f_s$ and dividing p by G. If the remainder of this division is 0, then $p \in \langle f_1, \ldots, f_s \rangle$. In the PNLA framework it is not necessary to compute the Gröbner basis and do this division. Instead, the degrees d_G, d_0, d_I are all determined from the set of reduced leading monomials $A^\star(d)$.*

Theorem 6.4 means that in practice one can recursively compute the reduced leading monomials $A^\star(d)$ of $M(d)$ using Algorithm 5.2, do the rank test for the ideal membership problem and increase the degree as long as the rank test fails. At some point d_G can be determined and the iterations can stop as soon as $d = d_G + \deg(p) - d_0$.

Example 6.7 *We know from Chapter 3 that the polynomial*

$$\begin{aligned} p &= 867\,x_1^5 - 1560\,x_3\,x_2\,x_1 - 2312\,x_2^2\,x_1 + 1560\,x_3\,x_1^2 + 2104\,x_2\,x_1^2 - 1526\,x_1^3 \\ &\quad +4896\,x_2 - 2295\,x_1, \end{aligned}$$

lies in $\mathcal{M}_{11}$ of the corresponding polynomial system

$$\left\{ \begin{array}{rcrcrcrcrcl} -9 & - & x_2^2 & - & x_3^2 & - & 3\,x_2^2x_3^2 & + & 8\,x_2x_3 & = & 0, \\ -9 & - & x_3^2 & - & x_1^2 & - & 3\,x_1^2x_3^2 & + & 8\,x_1x_3 & = & 0, \\ -9 & - & x_1^2 & - & x_2^2 & - & 3\,x_1^2x_2^2 & + & 8\,x_1x_2 & = & 0. \end{array} \right.$$

Testing equality (6.10) *fails for all degrees $d = 4$ up to 10. At $d = 11$ we have that*

$$A^\star(11) = \{x_1x_3^2, x_2^3, x_2^2x_3, x_2x_3^2, x_3^3, x_1^3x_2, x_1^3x_3, x_1^2x_2^2, x_1^2x_2x_3, x_1^5\},$$

which contains the pure powers x_1^5, x_2^3, x_3^3. This implies that the polynomial system has a finite affine solution set and that $d_G = 11$. Since $LM(p) = x_1^5$, G_p contains only one polynomial, the one with leading monomial x_1^5 and therefore $d_0 = 5$. Applying (6.11) *then results in*

$$d_I = d_G + deg(p) - d_o = 11 + 5 - 5 = 11.$$

The rank test of (6.10) *for $d = d_I = 11$ succeeds, the numerical rank for both matrices is 300.*

6.4 Iterative algorithm for finding $l(d)$ and $d^\star$

The analysis of the syzygies in Chapter 3 relied on checking each row of the Macaulay matrix for linear dependence together with the Inclusion-Exclusion principle. This approach has the disadvantage that the total number of binomial terms that needed to be computed grows combinatorially, while in fact the majority

of them cancel one another. In this section we will present an iterative algorithm that does not need to check each row of the Macaulay matrix, nor has to use the Inclusion-Exclusion Principle to find the final expression of $l(d)$ and corresponding degree of regularity $d^\star$. Instead of finding basis syzygies and calculating how these will propagate to higher degrees, we will simply update $l(d)$ recursively. The algorithm is presented in pseudo-code in Algorithm 6.3. The main idea is to compute the numerical value $p(d) - r(d)$ and compare it with the evaluation of $l(d)$ for each degree. If $p(d) - r(d) > l(d)$, then the polynomial $l(d)$ needs to count $p(d) - r(d) - l(d)$ additional linearly dependent rows. Furthermore, each of these additional rows will propagate to higher degrees and give rise to extra binomial terms. Similarly, if $p(d) - r(d) < l(d)$, then too many linearly dependent rows were counted and $l(d)$ needs to be adjusted with $l(d) - p(d) + r(d)$ negative binomial contributions. The $p(d) - r(d) < l(d)$ degrees for which positive contributions to $l(d)$ are made are stored in the vector d_+ and likewise for the $l(d) - p(d) + r(d)$ negative contributions in d_-. All information on how to express $l(d)$ in terms of binomial coefficients is hence coded in d_+ and d_-.

Algorithm 6.3 *Find $l(d)$ and degree of regularity $d^\star$*
Input: polynomial system $f_1, \ldots, f_n \in \mathcal{C}^n$
Output: $l(d)$ and degree of regularity $d^\star$

$d \leftarrow max(deg(f_1), deg(f_2), \ldots, deg(f_s))$
$d_+ \leftarrow \varnothing$
$d_- \leftarrow \varnothing$
$l(d) \leftarrow 0$
$r(d) \leftarrow$ *rank* $M(d)$
while $d \leq \sum_i^s deg(f_i)$ ***do***
  if $p(d) - r(d) > l(d)$ ***then***
    add d $p(d) - r(d) - l(d)$ *times to* d_+
  else if $p(d) - r(d) < l(d)$ ***then***
    add d $l(d) - p(d) + r(d)$ *times to* d_-
  end if
  $l(d) \leftarrow \sum_{i=1}^{|d_+|} \binom{d-d_+(i)+n}{n} - \sum_{i=1}^{|d_-|} \binom{d-d_-(i)+n}{n}$
  $d \leftarrow d + 1$
  $r(d) \leftarrow$ *update* $r(d)$ *using Theorem 4.1*
end while
$d^\star \leftarrow max(d_+, d_-) - n$

Since the algorithm iterates over the degrees, our recursive orthogonalization algorithm can be used to determine $r(d)$ recursively. The most economic way is by the determination of the orthogonal basis for the null space $N(d)$ of $M(d)$. Indeed, the total number of columns of $N(d)$ is $c(d)$ and we can therefore easily compute the rank as $r(d) = q(d) - c(d)$. It is then neither necessary to update the orthogonal basis $U(d)$ for the row space of $M(d)$, nor store it in memory. The binomial term in $l(d)$ that appears at the highest degree $d_{\max} = \max(d_+, d_-)$

determines the degree of regularity $d^\star$. Indeed,

$$\binom{d - d_{\max} + n}{n}$$

has zeros for $d = \{d_{\max} - n, d_{\max} - n + 1, \ldots, d_{\max} - 1\}$ and therefore the final expression for $l(d)$ is valid for all $d \geq d^\star = d_{\max} - n$. An upper bound for $d_{\max}$ comes from the theory of resultants. Macaulay showed in [60] that it is possible to determine whether a homogeneous polynomial system $f_1^h, \ldots, f_n^h$ of degrees $d_1, \ldots, d_n$ has a common root by computing the determinant of a submatrix of $M(d)$ for $d = \sum_{i=1}^n d_i - n$. This essentially means that $d^\star = \sum_{i=1}^n d_i - n$, which results in a maximal degree of $\sum_{i=1}^n d_i$ in Algorithm 6.3.

Remark 6.8 *Since Theorem 4.1 can be used to recursively compute orthogonal bases for the row and null space of $M(d)$, it is therefore tempting to apply the same theorem on $M(d)^T$ to directly determine $l(d)$. This is however no possible since $M(d)$ does not exhibit the same structure column-wise as it has row-wise.*

Remark 6.9 *In computer algebra, one would need to compute a Gröbner basis G of $f_1, \ldots, f_s$ in order to describe all basis syzygies and find the degree of regularity. In the PNLA framework, no computation of a Gröbner basis is required. Instead, one needs to determine the numerical rank of $M(d)$ for increasing degrees d.*

Example 6.8 *We illustrate Algorithm 6.3 with the polynomial system from Example 3.13:*

$$\begin{cases} f_1: & x_2^2\,x_3 + 2\,x_1\,x_2\,x_4 - 2\,x_1 - x_3 = 0, \\ f_2: & -x_1^3\,x_3 + 4\,x_1\,x_2^2\,x_3 + 4\,x_1^2\,x_2\,x_4 + 2\,x_2^3\,x_4 + 4\,x_1^2 - 10\,x_2^2 \\ & +4\,x_1\,x_3 - 10\,x_2\,x_4 + 2 = 0, \\ f_3: & 2\,x_2\,x_3\,x_4 + x_1\,x_4^2 - x_1 - 2\,x_3 = 0, \\ f_4: & -x_1\,x_3^3 + 4\,x_2\,x_3^2\,x_4 + 4\,x_1\,x_3\,x_4^2 + 2\,x_2\,x_4^3 + 4\,x_1\,x_3 \\ & +4\,x_3^2 - 10\,x_2\,x_4 - 10\,x_4^2 + 2 = 0. \end{cases}$$

The expression for $l(d)$ is initialized to 0. The Macaulay matrix is of full row rank for degrees 4 and 5. For $d = 6$, we have that $p(6) - r(6) = 100 - 99 = 1 > l(6) = 0$. We therefore set $d_+ = 6$ and

$$l(d) = \binom{d - 6 + 4}{4}.$$

After incrementing the degree we find that $p(7) - r(7) = 210 - 201 = 9 > l(7) = 5$ and we therefore update d_+ and $l(d)$ to $d_+ = \{6, 7, 7, 7, 7\}$ and

$$l(d) = \binom{d - 6 + 4}{4} + 4\binom{d - 7 + 4}{4}$$

respectively. The algorithm finishes at $d = 14$ *with*

$$d_+ = \{6, 7, 7, 7, 7, 8, 13, 13, 13, 13\},$$

and

$$d_- = \{11, 11, 11, 11, 11, 11, 14\},$$

which indeed corresponds with the final expression for $l(d)$

$$\begin{aligned} l(d) &= \tbinom{d-6+4}{4} + 4\tbinom{d-7+4}{4} + \tbinom{d-8+4}{4} + 4\tbinom{d-13+4}{4} - 6\tbinom{d-11+4}{4} - \tbinom{d-14+4}{4} \\ &= \tfrac{1}{8}d^4 - \tfrac{13}{12}d^3 - \tfrac{5}{8}d^2 + \tfrac{19}{12}d + 105. \ (d \geq 10) \end{aligned}$$

Observe that Algorithm 6.3 finds the desired expression for $l(d)$ *at* $d = 14$*. In contrast, the basis syzygies analysis of Chapter 3 required the construction of* $M(15)$ *and the computation of 10241 binomial terms.*

6.5 Multivariate Polynomial Elimination

Gaussian elimination is probably the most known form of elimination. It involves the manipulation of linear equations such that the solution set does not change and one of the resulting equations is univariate. The same idea is generalized by a Gröbner basis using a lexicographic monomial ordering. The problem of multivariate elimination can be stated as follows:

Problem 6.2 *Given a system of multivariate polynomials* $f_1, \ldots, f_s$ *and a proper subset of variables* $x_e \subsetneq \{x_i : i = 1, \ldots, n\}$*, find a polynomial* $g = \sum_i^s h_i f_i$ *in which all variables* x_e *are eliminated.*

The polynomial g lies therefore obviously in $\mathcal{M}_d$ for some degree d. In addition, g also lies in the vector space spanned by all monomials which do not contain any element of x_e up to the same degree d.

Definition 6.1 *Given a proper subset of variables* $x_e \subsetneq \{x_i : i = 1, \ldots, n\}$ *then the elimination vector space* $\mathcal{E}_d$ *is the vector space of polynomials with maximal degree* d *spanned by all monomials that do not contain any of the variables of* x_e*.*

A canonical basis $E(d)$ for the vector space $\mathcal{E}_d$ is easily obtained. The following example illustrates this.

Example 6.9 *Suppose $x_e = \{x_2, x_4\} \in \mathcal{C}_2^4$. A canonical basis for $\mathcal{E}_2$ is then given by*

$$\begin{array}{c} \begin{array}{cccccccccccccc} 1 & x_1 & x_2 & x_3 & x_4 & x_1^2 & x_1x_2 & x_1x_3 & x_1x_4 & x_2^2 & x_2x_3 & x_2x_4 & x_3^2 & x_3x_4 & x_4^2 \end{array} \\ \left(\begin{array}{cccccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array} \right). \end{array}$$

Again, notice that the $q \times k$ matrix $E(d)^T$ can always be written as

$$E(d)^T \;=\; P \begin{pmatrix} I_k \\ 0 \end{pmatrix}. \tag{6.12}$$

This means that it is also possible to encode $E(d)$ completely as a vector j of indices. The intersection of $\mathcal{M}_d$ and $\mathcal{E}_d$ brings us naturally to the geometry of multivariate elimination.

6.5.1 The Geometry of Polynomial Elimination

The polynomial g which is to be found lies both in $\mathcal{M}_d$ and $\mathcal{E}_d$. Polynomial elimination therefore corresponds with finding an intersection of these two vector spaces. This is depicted in Figure 6.1. Here, both the row space of $M(d)$ and $E(d)$ are represented as two-dimensional planes. The polynomial g corresponds with the vector lying in the one-dimensional intersection.

6.5.2 Algorithm & Numerical Implementation

In this section, we present and explain Algorithm 6.4 for doing multivariate elimination. Since $\deg(g)$ is unknown, the algorithm iterates over the degree d with an initial value given by the maximal degree of the given polynomials $f_1, \ldots, f_s$. In each iteration, the intersection between $\mathcal{M}_d$ and $\mathcal{E}_d$ is computed (Appendix A, Sections A.9 and A.10). If there is no intersection, then the degree is increased and the vector of indices j and orthogonal basis N are updated. As soon as there is an intersection between $\mathcal{M}_d$ and $\mathcal{E}_d$, the first principal vector v_m that lies in this intersection is returned as g. Again, we use the same tolerance τ for both the determination of the zero principal angles and the rank.

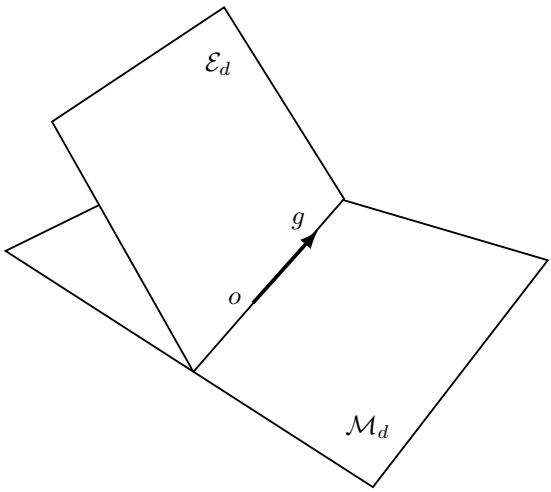

Figure 6.1: **The polynomial $g = \sum_i^s h_i f_i$ with all variables x_e eliminated lies in the intersection of $\mathcal{M}_d$ and $\mathcal{E}_d$.**

Algorithm 6.4 *Multivariate Elimination*
***Input**: polynomials $f_1, \ldots, f_s \in \mathcal{C}_d^n$, monomial set x_e, tolerance τ*
***Output**: $g \in \mathcal{M}_d \cap \mathcal{E}_d$*

```
d ← max(deg(f_1), deg(f_2), ..., deg(f_s))
g ← ∅
N ← orthogonal basis for null(M(d)) using Algorithm 4.2
j ← vector of nonzero row indices of canonical basis for E_d
while g = ∅ do
  [W S Z] ← SVD( N(j,:)^T )
  if arcsin(μ_m) < τ then
    g ← v_m^T
  else
    d ← d + 1
    update j
    update N using Theorem 4.1
  end if
end while
```

Remark 6.10 *It is possible that the dimension of the intersection is $s > 1$. One could then return all s orthogonal basis vectors $v_{m-s+1}, \ldots, v_m$, since in this case there are an infinite amount of solutions $g \in span\{v_{m-s+1}, \ldots, v_m\}$.*

Remark 6.11 *In computer algebra, multivariate elimination is achieved by computing a Gröbner basis of $f_1, \ldots, f_s$ using a lexicographic monomial ordering. This results in a Gröbner basis that has a distinct "triangular" form. In the PNLA framework, no Gröbner basis needs to be computed. Instead, the desired polynomial g is retrieved as the basis vector of the intersection of two subspaces.*

6.5.3 Numerical Experiments

From the following polynomial system [86, p. 823] in $\mathcal{C}_4^6$

$$\left\{\begin{array}{rcl} x_1^2 + x_3^2 - 1 &=& 0, \\ x_2^2 + x_4^2 - 1 &=& 0, \\ x_5 x_3^3 + x_6 x_4^3 - 1.2 &=& 0, \\ x_5 x_1^3 + x_6 x_2^3 - 1.2 &=& 0, \\ x_5 x_3^2 x_1 + x_6 x_4^2 x_2 - 0.7 &=& 0, \\ x_5 x_3 x_1^2 + x_6 x_4 x_2^2 - 0.7 &=& 0, \end{array}\right.$$

we eliminate $x_e = \{x_1, x_2, x_3, x_4, x_5\}$ using Algorithm 6.4. For all $d < 10$ we have that $\arcsin(\mu_m) > \tau$. For $d = 10, \tau = 1.46 \times 10^{-10}$ and

$$\arcsin(\mu_m) = 4.44 \times 10^{-16} < \tau.$$

The Macaulay matrix $M(10)$ is a 9702×8008 matrix with 29106 nonzero elements which corresponds with a density of 3.7%. The first principal vector is

$$g = 0.9011 - 0.0\, x_6 - 0.4335\, x_6^2 + 0.0\, x_6^3 + 0.0\, x_6^4 - 0.0\, x_6^5 - 0.0\, x_6^6 + 0.0\, x_6^7 - 0.0\, x_6^8.$$

All 0.0 coefficients are bounded from above by τ and can therefore be considered to be numerically zero. We therefore write g as

$$g = 0.9011 - 0.4335\, x_6^2.$$

This implies that the roots of this particular polynomial system have only 2 distinct x_6 components, 1.4417 and -1.4417. A Gröbner basis according to the lexicographic monomial ordering with $x_1 > x_2 > x_3 > x_4 > x_5 > x_6$ is

$$G = \left\{\begin{array}{rcl} -6859 + 3300\, x6^2 &=& 0, \\ -6859 + 3300\, x5^2 &=& 0, \\ 133 - 330\, x6\, x4 + 361\, x4^2 &=& 0, \\ 3300\, x5\, x4\, x6 - 6270\, x5 + 6859\, x3 &=& 0, \\ -330\, x6 + 361\, x4 + 361\, x2 &=& 0, \\ -3300\, x5\, x4\, x6 + 6859\, x1 &=& 0. \end{array}\right.$$

Observe its "triangular form", the first polynomial is univariate in x_6, the second polynomial is univariate in x_5. The monomial x_4 is only present in the third

Gröbner basis polynomial, etc... From this Gröbner basis G, the exact solutions for x_6 are easily computed as

$$x_6 \;=\; \pm\frac{19}{330}\sqrt{627}.$$

These allows us to determine the absolute forward error of our numerical solutions: 7.40×10^{-11}.

When eliminating $x_e = \{x_1, x_2, x_3, x_4\}$ from the same polynomial system, we have

$$\arcsin(\mu_m) = 6.34 \times 10^{-16} < \tau = 2.52 \times 10^{-11} \;\text{ for } d = 8.$$

The principal vector is then

$$h = -0.0305 + 0.7141\,x_5^2 - 0.6994\,x_6^2 - 0.0004\,x_5^4 + 0.0009\,x_5^2x_6^2 - 0.0004\,x_6^4.$$

The polynomial h also contains 22 other nonzero coefficients, each of which is bounded from above by 10^{-19}. We have omitted these numerical zeros for the sake of presentation.

We now add perturbations of 10^{-6} to the original polynomial system to obtain

$$\begin{cases} 1.000001\,x_1^2 + x_3^2 - 1 &= 0, \\ x_2^2 + x_4^2 - 1 &= 0, \\ x_5x_3^3 + x_6x_4^3 - 1.200001 &= 0, \\ x_5x_1^3 + x_6x_2^3 - 1.2 &= 0, \\ x_5x_3^2x_1 + 1.000001\,x_6x_4^2x_2 - 0.7 &= 0, \\ 1.000001\,x_5x_3x_1^2 + x_6x_4x_2^2 - 0.700001 &= 0. \end{cases}$$

When eliminating $x_e = \{x_1, x_2, x_3, x_4, x_5\}$ we have again for $d = 10$ that

$$\arcsin(\mu_m) = 6.34 \times 10^{-14} < \tau = 1.46 \times 10^{-10}.$$

Since $d = 10$ the Macaulay matrix will have the same size and structure as for the original polynomial system. The univariate polynomial in x_6 is now given by

$$\hat{g} = 0.9011 - 0.4335\,x_6^2,$$

for which $||g - \hat{g}||_2 < 10^{-7}$. This reflects the loss of precision due to the added noise.

6.6 Approximate LCM and GCD

In this section we present an algorithm to numerically compute both the least common multiple (LCM) and greatest common divisor (GCD) of two multivariate

polynomials. For the univariate case, this problem has been studied already quite intensively with different formulations [19, 36, 51, 68, 73, 94]. The most common framework is the one of the approximate LCM/GCD, which are usually defined as the exact LCM/GCD of polynomials $\tilde{f}_1, \tilde{f}_2$ that satisfy

$$\frac{||f_1 - \tilde{f}_1||_2}{||f_1||_2} \leq \epsilon, \text{ and } \frac{||f_2 - \tilde{f}_2||_2}{||f_2||_2} \leq \epsilon,$$

where ϵ is some user-defined tolerance. In this section, we will propose a geometric definition. The connection with the traditional definition will be made in Section 6.6.3. Numerical linear algebra plays a stronger role in the context of approximate GCDs, mainly since many numerical methods are based on the SVD of either Sylvester matrices [19, 20] or Bézout matrices [30, 41]. This is because the determination of the degree of the (approximate) GCD corresponds with the detection of a rank-deficiency of the Sylvester/Bézout matrix. Computing the LCM has not received as much attention. Although most GCD-methods based on the Sylvester matrix make use of the LCM implicitly, this is never really mentioned. Existing literature on numerical SVD-based methods to compute multivariate approximate GCD includes [50, 90]. These methods generalize the univariate SVD-based methods to the multivariate case in the sense that they also detect a numerical rank deficiency of a (multivariate) Sylvester matrix. The numerical algorithm presented in this section differs from these multivariate algorithms in the sense that the Sylvester matrix is not used.

The following theorem interrelates the LCM of multivariate polynomials with their GCD and will be of crucial importance to our method.

Theorem 6.5 *([22, p. 190]) Let $f_1, f_2 \in \mathcal{C}^n$ and l, g their LCM and GCD respectively, then*

$$l\, g \;=\; f_1\, f_2. \tag{6.13}$$

This theorem provides a way to find the LCM or GCD once either of them has already been found. The algorithm we propose will first compute a LCM and derive a GCD as a least-squares solution of (6.13).

6.6.1 Computing the LCM

Finding an approximate LCM is be the first step in the proposed algorithm. As mentioned in the previous section, the corresponding GCD will then be computed using Theorem 6.5. We first rewrite Theorem (6.5) as

$$l = f_1\, k_1 = f_2\, k_2 \tag{6.14}$$

where $k_1, k_2 \in \mathcal{C}^n$ and of degrees $\deg(l) - d_1$ and $\deg(l) - d_2$ respectively. From (6.14) it can be immediately deduced that the LCM of f_1 and f_2 lies in the row space of both multiplication matrices $M_{f_1}(d)$ and $M_{f_2}(d)$. We therefore define an approximate LCM as the polynomial that lies in the intersection $\mathcal{M}_{f_1} \cap \mathcal{M}_{f_2}$ for a certain degree d and with a certain tolerance τ.

Definition 6.2 *Let $f_1, f_2 \in \mathcal{C}^n$ and suppose a tolerance $\tau > 0$. If the smallest principal angle θ_k between $\mathcal{M}_{f_1}$ and $\mathcal{M}_{f_2}$ satisfies*

$$\theta_k \leq \tau,$$

then the corresponding principal vector v_k is the approximate LCM with tolerance τ.

This definition is completely in line with the literature, in the sense that the tolerance τ is an explicit part of the definition. Algorithm 6.5 is a direct translation of Definition 6.2. Since $\deg(l)$ is not known a priori, iterations over the degree are necessary. An upper bound for $\deg(l) \triangleq d_l$ is given by $\deg(f_1)+\deg(f_2)$ since in this case the approximate GCD $g = 1$. Deciding whether an intersection exists between $\mathcal{M}_{f_1}$ and $\mathcal{M}_{f_2}$ is done by inspecting the smallest principal angle between these two vector spaces. $Q_1^T N_2$ is a $p_1(d) \times c_2(d)$ matrix. One can keep the dimensions of this matrix minimal by computing the orthogonal basis Q_1 for the polynomial with the highest degree, or equivalently computing N_2 for the polynomial of lowest degree.

Algorithm 6.5 *Computing an approximate LCM*
***Input**: polynomials $f_1, f_2 \in \mathcal{C}^n$, tolerance τ*
***Output**: approximate LCM l of f_1, f_2*

```
d ← max(deg(f1), deg(f2))
l ← ∅
Q1 ← orthogonal basis row(Mf1(d))
N2 ← orthogonal basis null(Mf2(d))
while l = ∅ & d ≤ deg(f1) + deg(f2) do
  [W S Z] ← SVD( Q1^T N2 )
  if arcsin(μm) < τ then
    l ← vm^T
  else
    d ← d + 1
    update Q1 using Theorem 4.1
    update N2 using Theorem 4.1
  end if
end while
```

6.6.2 Computing the GCD

Once the approximate LCM l is found a corresponding approximate GCD g is easily retrieved from Theorem 6.5. This implies that no extra tolerance needs to be defined anymore. One could compute the vector corresponding with $f_1\, f_2$ and divide this by l. This division is achieved by constructing the Macaulay matrix of l and solving the sparse overdetermined system of equations

$$M_l^T(d_1 + d_2)\, g^T \;=\; M_{f_2}^T(d_1 + d_2)\, f_1^T.$$

This can be done in the least squares sense [19] using a sparse Q-less QR decomposition [24]. The approximate GCD g is defined in this case as the solution of

$$g \;=\; \underset{w}{\operatorname{argmin}}\, ||f_1\, M_{f_2}(d_1 + d_2) - wM_l(d_1 + d_2)||_2^2.$$

The 2-norm of the residual $||f_1\, M_{f_2}^T(d_1 + d_2) - gM_l(d_1 + d_2)||_2$ then provides a measure on how well the computed GCD satisfies Theorem 6.5 with the LCM. The problem with this approach however is that computing the product $f_1\, f_2$ can be quite costly in terms of storage. The need to do this multiplication can be circumvented by first doing the division $h_2 = l/f_2$. This is also done by solving another sparse overdetermined system

$$M_{f_2}(d_l)^T h_2^T \;=\; l^T, \tag{6.15}$$

where d_l is the degree of l. $M_{f_2}(d_l)$ will typically have much smaller dimensions than $M_l(d_1 + d_2)$. From Theorem 6.5 it is easy to see that $h_2 = f_1/g$ and hence an alternative for the computation of the approximate GCD g is solving the sparse linear system

$$M_{h_2}(d_1)^T g^T \;=\; f_1^T. \tag{6.16}$$

We therefore define the approximate GCD as the least squares solution of (6.16).

Definition 6.3 *Let $f_1, f_2 \in \mathcal{C}^n$ and let l be their approximate LCM as in Definition 6.2, then their approximate GCD g is the solution of*

$$g \;=\; \underset{w}{\operatorname{argmin}}\, ||f_1^T - M_{h_2}(d_1)^T\, w^T||_2^2.$$

where h_2 is the least squares solution of (6.15).

For each of the divisions described above, the 2-norm of the residual provides a natural measure on how well the division succeeded. Since g is defined up to a scalar, one can improve the 2-norm of the residual by normalizing the right-hand side f_1^T. The residual thus improves with a factor $||f_1||_2$ and will typically be of the same order as $||l - w\, M_{f_2}(d_l)||_2$. Algorithm 6.6 summarizes the high-level algorithm of finding the approximate GCD. The computational complexity of the

entire method is dominated by the cost of solving the 2 linear systems (6.15) and (6.16). These are both $O(qp^2)$ where p and q stand for the number of rows and columns of the matrices involved. The large number of zero elements however make the solving of these systems still feasible.

Algorithm 6.6 *Computing an approximate GCD*
Input*: polynomials $f_1, f_2, l \in \mathcal{C}^n$ with l an approximate LCM of f_1, f_2*
Output*: approximate GCD g of f_1, f_2*

$$h_2 \leftarrow \underset{w}{\operatorname{argmin}} \, ||l - wM_{f_2}(d_l)||_2^2$$
$$g \leftarrow \underset{w}{\operatorname{argmin}} \, ||\tfrac{f_1}{||f_1||_2} - wM_{h_2}(d_1)||_2^2$$

6.6.3 Choosing the numerical tolerance τ

In this section we will derive the relationship between the tolerance τ of Algorithm 6.5 and the ϵ which is commonly used in other methods [30]. Let $e_1 = f_1 - \tilde{f}_1, e_2 = f_2 - \tilde{f}_2$ with $||e_1||_2 \leq \epsilon_1 ||f_1||_2, ||e_2||_2 \leq \epsilon_2 ||f_2||_2$. Then both $M_{f_1}(d)$ and $M_{f_2}(d)$ are perturbed by structured matrices E_1 and E_2, which can be interpreted as the multiplication matrices of the perturbations e_1, e_2. Now suppose that

$$\frac{||E_1||_2}{||M_{f_1}(d)||_2} \leq \bar{\epsilon}_1, \quad \frac{||E_2||_2}{||M_{f_2}(d)||_2} \leq \bar{\epsilon}_2,$$

then in [14, p. 585] the following expression is proved

$$|\Delta\theta_k| \leq \sqrt{2}\,(\bar{\epsilon}_1\,\kappa_1 + \bar{\epsilon}_2\,\kappa_2) + O(\delta^2), \tag{6.17}$$

where κ_1, κ_2 are the condition numbers of $M_{f_1}(d)$ and $M_{f_2}(d)$ respectively and $\delta = (\bar{\epsilon}_1\,\kappa_1 + \bar{\epsilon}_2\,\kappa_2)$. Since in the exact case $\theta_k = 0$, the left hand side of (6.17) is actually $|\tilde{\theta}_k|$, the absolute value of the perturbed principal angle due to the perturbations e_1 and e_2. We can therefore take the right-hand side of (6.17) as the tolerance τ from Algorithm 6.5. We now show that we can set $\epsilon_1 = \bar{\epsilon}_1$. It is clear that

$$\begin{aligned} & ||E_1||_2 \leq \bar{\epsilon}_1\,||M_{f_1}(d)||_2 \\ \Leftrightarrow \quad & \sigma_1(E_1) \leq \bar{\epsilon}_1\,\sigma_1(M_{f_1}(d)) \\ \Leftrightarrow \quad & ||e_1||_1 \leq \bar{\epsilon}_1\,||f_1||_1 \\ \Leftrightarrow \quad & ||e_1||_2 \leq \bar{\epsilon}_1\,||f_1||_2 \end{aligned}$$

from which the desired follows. Applying the same reasoning for f_2 means that we can also set $\bar{\epsilon}_2 = \epsilon_2$. Defining $\epsilon = \max(\epsilon_1, \epsilon_2)$ and substituting for ϵ_1, ϵ_2 in (6.17) results in

$$|\Delta\theta_k| \leq \sqrt{2}\,\epsilon\,(\kappa_1 + \kappa_2). \tag{6.18}$$

Replacing the condition numbers by their upper bounds for the nontrivial case, we get

$$|\Delta\theta_k| \leq \sqrt{2}\,\epsilon\,\left(\frac{|||f_1||_1}{2|m_{00,1}| - |||f_1||_1} + \frac{|||f_2||_1}{2|m_{00,2}| - |||f_2||_1}\right), \tag{6.19}$$

where extra subscripts are introduced to m_{00} to distinguish between the constant term of f_1 and f_2. If we assume that all principal angles have perturbations of the same order of magnitude, then choosing the right-hand side of (6.19) as τ should enable us to recover an approximate GCD of the same degree as the GCD of the unperturbed polynomials. Observe also from (6.18) that the well-conditioning of the multiplication matrices implies that the principal angles are well-determined.

6.6.4 Numerical Experiments

In this section we discuss some numerical examples and compare the results with those obtained from NAClab, the MATLAB counterpart of ApaTools, by Zeng [91]. The 'mvGCD' method from NAClab improves the accuracy of its result by Gauss-Newton iterations and will hence always produce results with lower relative errors. We therefore use these results as a reference.

Example 1

First, consider the following two bivariate polynomials with exact coefficients

$$\begin{aligned} f_1 &= (x_1x_2 + x_2^2 + 200)\,(x_1x_2 + x_2 + 200)\,(100 + 2x_1^3 - 2x_1x_2 + 3x_2^2) \\ f_2 &= (x_1x_2 + x_2^2 + 200)\,(x_1x_2 + x_2 + 200)\,(100 - 2x_1^3 + 2x_1x_2 - 3x_2^2). \end{aligned}$$

Both f_1 and f_2 satisfy the nontrivial case. The relatively large coefficients of the constant terms in the 3 factors have as a consequence that the absolute value of the coefficients of f_1, f_2 vary between 1 and 4000000. Since we assume the coefficients are exact, we set the tolerance τ to the tolerance when doing the determination of the numerical rank as discussed in Section 4.5. The exact GCD g is obviously the product of the first 2 factors of f_1 and hence the exact LCM l is

$$l = g\,(100 + 2x_1^3 - 2x_1x_2 + 3x_2^2)\,(100 - 2x_1^3 + 2x_1x_2 - 3x_2^2).$$

The smallest principal angle θ_k drops from 0.04 to 8.83×10^{-16} when going from $d = 9$ to $d = 10$. The condition numbers of $M_{f_2}(10)$ and $M_{h_2}(10)$ are 1.05 and 1.06 respectively. The relative error between our computed LCM and the exact answer is 8.01×10^{-13}. The 2-norm of the residual for calculating h_2 is 2.12×10^{-14} and is for solving (6.16) 6.11×10^{-15}. For the computed GCD, the relative error is 3.48×10^{-11}. The GCD computed from NAClab has a relative error of 1.24×10^{-20}.

The 2-norm of the absolute difference between the GCD of our method compared to one from NAClab is 5.13×10^{-13}. In order to investigate how our method performs when the polynomials have inexact coefficients we now add perturbations of the order 10^{-3} to f_1, f_2 and obtain

$$\begin{aligned}\tilde{f}_1 &= (x_1x_2 + x_2^2 + 200)(x_1x_2 + x_2 + 200 + 10^{-3}x_1)(100 + 2x_1^3 - 2x_1x_2 + 3x_2^2)\\ \tilde{f}_2 &= (x_1x_2 + x_2^2 + 200)(x_1x_2 + x_2 + 200 - 10^{-3}x_1)(100 - 2x_1^3 + 2x_1x_2 - 3x_2^2).\end{aligned}$$

Again, both $\tilde{f}_1$ and $\tilde{f}_2$ satisfy the nontrivial case. The perturbation of 10^{-3} in one of the factors corresponds with $\epsilon_1 \approx \epsilon_2 \approx 5.00 \times 10^{-6}$, so we set $\epsilon = 10^{-5}$. Now, the smallest principal angle θ_k drops from 0.04 to 7.58×10^{-9} when going from $d = 9$ to $d = 10$. From (6.18) we have $\tau = 2.77 \times 10^{-5}$ and from (6.19) $\tau = 3.29 \times 10^{-5}$. Both tolerances recover an approximate GCD of degree 4. The 2-norm of the residual when calculating h_2 and the approximate GCD is 7.60×10^{-9} and 4.22×10^{-9} respectively. The absolute difference between our approximate GCD and the one from NAClab is of the order 10^{-5}, which agrees with $\epsilon = 10^{-5}$.

Example 2

In this second numerical experiment, the capability of Algorithms 6.5 and 6.6 to handle an increasing number of variables is tested. Suppose we have the following polynomials

$$\begin{aligned}u_k &= (x_1 + x_2 + \ldots + x_k + 1)^2,\\ v_k &= (x_1 - x_2 - \ldots - x_k - 2)^2,\\ w_k &= (x_1 + x_2 + \ldots + x_k + 2)^2.\end{aligned}$$

We then set $f_1 = u_k\, v_k$ and $f_2 = u_k\, w_k$ and compute the GCD for $k = 2, \ldots, 10$. We will denote the exact GCD by g, the GCD found with NAClab by g_n and the GCD found by Algorithm 6.6 by g_τ. Table 6.1 lists the relative forward errors

$$e_n = \frac{||g - g_n||_2}{||g||_2}, \quad e_\tau = \frac{||g - g_\tau||_2}{||g||_2},$$

for NAClab and Algorithm 6.6 respectively. As expected, g_n lies slightly closer to the exact result due to the additional Gauss-Newton iterations.

Table 6.1: Relative forward errors Example 2

	$k=2$	$k=3$	$k=4$	$k=5$
e_n	5.55×10^{-17}	2.32×10^{-16}	1.24×10^{-16}	2.42×10^{-16}
e_τ	5.02×10^{-16}	1.14×10^{-15}	1.31×10^{-15}	3.10×10^{-15}

$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
3.09×10^{-15}	2.93×10^{-16}	2.20×10^{-16}	1.81×10^{-16}	2.61×10^{-16}
8.41×10^{-15}	1.12×10^{-14}	1.08×10^{-14}	1.56×10^{-14}	8.28×10^{-14}

Example 3

Next, the capability of Algorithms 6.5 and 6.6 to handle high degrees is tested. Let

$$\begin{aligned} p &= x_1 - x_2\, x_3 + 1 \\ q &= x_1 - x_2 + 3\, x_3 \\ f_1 &= p^6\, q^{12} \\ f_2 &= p^{12}\, q^6. \end{aligned}$$

For this case the exact GCD is $p^6\, q^6$. This is reminiscent of f_1 and f_2 having multiple common roots for the univariate case. It took NAClab several runs to find a result. In most cases it returned an error message. This is probably somehow related to the high degrees since for the case $f_1 = p^4\, q^6$ and $f_2 = p^6\, q^4$ a GCD could always be computed with NAClab. For the high degree case, when NAClab could return a computed result, the relative forward errors were $e_n = 2.46 \times 10^{-14}$ and $e_\tau = 3.37 \times 10^{-13}$.

Example 4

The next example demonstrates the robustness of Algorithm 6.6 with respect to noisy coefficients. We revisit the polynomials of Example 2 and set $k = 3$. We therefore have

$$\begin{aligned} u &= (x_1 + x_2 + x_3 + 1)^2 \\ v &= (x_1 - x_2 - x_3 - 2)^2 \\ w &= (x_1 + x_2 + x_3 + 2)^2 \end{aligned}$$

and again set $f_1 = u\,v$ and $f_2 = u\,w$. Every nonzero coefficient of f_1, f_2 is then perturbed with noise, uniformly drawn from $\left[0, 10^{-k}\right]$ for $k = 1, 3, 5, 7$. Table 6.2 lists the relative forward errors e_n, e_τ. Again, the approximate GCD from NAClab lies slightly closer to the exact result. However, NAClab cannot find a GCD anymore for $k = 1$.

Table 6.2: Errors Example 4

	$k=1$	$k=3$	$k=5$	$k=7$
e_n	NA	7.83×10^{-05}	4.45×10^{-07}	3.86×10^{-09}
e_τ	1.17×10^{-2}	1.01×10^{-04}	5.73×10^{-07}	2.23×10^{-08}

Example 5

The final example is the blind image deconvolution problem from Chapter 1. Since the desired image $p(i, j)$ is 150×150 pixels, its z-transform $P(z_1, z_2)$ is a bivariate polynomial of degree 298. The filters $D_1(z_1, z_2), D_2(z_1, z_2)$ are both of degree 2, which means that $F_1(z_1, z_2), F_2(z_1, z_2)$ are both of degree 300. Since $p(i, j)$ is a colour image, it is represented by 3 polynomials, one for each of its colour channels (red-green-blue). Running Algorithm 6.6 with $\epsilon = 0.02$ on each of the polynomials corresponding with $F_1(z_1, z_2)$ and $F_2(z_1, z_2)$ results in the image of Figure 6.2. The relative forward errors for each colour channel are $0.0119, 0.0199, 0.0300$ respectively.

Figure 6.2: **The image retrieved as the approximate GCD of $F_1(z_1, z_2)$ and $F_2(z_1, z_2)$.**

6.7 Removing Multiplicities

In this section, we will present the theorem that removes the multiplicities of affine roots and consequently also removes all root at infinity. Unlike the procedure outlined in Chapter 3 Section 3.6.3, no knowledge on the roots is required and they do not need to have the exact same multiplicity structure. The key ingredient will be the square-free parts of specific univariate polynomials. Remember that the square-free part p_{red} of a univariate polynomial p is the same polynomial with all multiplicities of its roots removed (Appendix B Section B.6). The following lemma provides a way of computing p_{red} from p without the need of computing its roots.

Lemma 6.2 *Let $p \in C_d^1$ and $p' = p\,D_1$ be its first derivative then*

$$p_{red} \;=\; \frac{p}{GCD(p,p')}. \tag{6.20}$$

Proof. Since

$$p \;=\; c\,(x-z_1)^{m_1}\,(x-z_2)^{m_2}\;\ldots\;(x-z_r)^{m_r}$$

then

$$p' \;=\; \prod_{j=1}^{r} r_j\,(x-z_j)^{m_j-1}\;H(x)$$

with

$$H(x) \;=\; c\sum_{k=1}^{r}\prod_{j\neq k}(x-z_j)$$

a polynomial in C_d vanishing at none of the $z_1,\ldots,z_m$. Clearly

$$\mathrm{GCD}(p,p') \;=\; \prod_{j=1}^{r}(x-z_j)^{m_j-1}$$

which proves (6.20). □

The zero-dimensionality of the projective variety implies that we can find for each variable $x_i\,(i=1,\ldots,n)$ a univariate polynomial $p(x_i)=p_i\in\mathcal{M}_d$. The following theorem tells us how we can remove the multiplicities of the affine roots and obtain the radical ideal.

Theorem 6.6 *([21, p. 41]) Let $I = \langle f_1, \ldots, f_s \rangle$ be a zero-dimensional ideal. For each $i = 1, \ldots, n$, let p_i be the unique univariate polynomial that lies in $\mathcal{M}_d$, and let $p_{i,red}$ be the square-free part of p_i. Then*

$$\sqrt{I} \;=\; \langle f_1, \ldots, f_s, p_{1,red}, \ldots, p_{n,red} \rangle.$$

Theorem 6.6 is indeed quite similar to Section 3.6.3, extra polynomials are added such that the resulting polynomial ideal is radical and hence all multiplicities of the roots are removed. It is also quite straightforward to see that all roots at infinity will be removed as well. Indeed, from Lemma 3.4 we know that the roots at infinity are found from the polynomial system that is obtained from making all generators of the radical ideal homogeneous and setting $x_0 = 0$. The addition of the square-free parts of the univariate polynomials then ensures that the only root at infinity is $0 = (0, \ldots, 0)$, which is not a valid solution. Hence, the new polynomial system has no more roots at infinity. Converting Theorem 6.6 into an algorithm is rather straightforward. The first step will be to apply our elimination algorithm, Algorithm 6.4, to obtain for each variable x_i $(1 \leq i \leq n)$ its univariate polynomial $p_i \in \mathcal{M}_d$. The next step is to compute its square-free part $p_{i,\mathrm{red}}$ by using Lemma 6.2. Since the GCD of p_i and p_i' needs to be computed, we first compute its LCM l using Algorithm 6.5. We then have from Theorem 6.5 that

$$l \;=\; h_1\, p_i \;=\; h_2\, p_i',$$

with

$$h_1 \;=\; \frac{p_i'}{\mathrm{GCD}(p_i, p_i')} \;\text{ and }\; h_2 \;=\; \frac{p_i}{\mathrm{GCD}(p_i, p_i')}.$$

This means that h_2 is exactly the desired square-free part, which can be computed as the least-squares solution of

$$\underset{w}{\mathrm{argmin}}\; ||l - wM_{p_i'}(d_l)||_2^2 \,.$$

The whole procedure is summarized in Algorithm 6.7.

Algorithm 6.7 *Compute square-free generators of radical ideal $\sqrt{I} \neq I$*
***Input**: generators $f_1, \ldots, f_s$ of zero-dimensional ideal I*
***Output**: square-free parts $p_{1,red}, \ldots, p_{n,red}$*

```
for i = 1, ..., n do
    p_i ← univariate polynomial from Algorithm 6.4
    p_i' ← compute first derivative of p_i
    l_i ← LCM(p_i, p_i') from Algorithm 6.5
    p_{i,red} ← argmin_w ||l − w M_{p_i'}(d_l)||_2^2
end for
```

Example 6.10 *We illustrate Algorithm 6.7 on the following polynomial system*

$$\left\{\begin{array}{rcl} x_2^4\,x_1 + 3\,x_1^3 - x_2^4 - 3\,x_1^2 & = & 0, \\ x_1^2\,x_2 - 2\,x_1^2 & = & 0, \\ 2\,x_2^4\,x_1 - x_1^3 - 2\,x_2^4 + x_1^2 & = & 0, \end{array}\right.$$

that has 13 projective roots: 4 at infinity, the affine origin $(1,0,0)$ *with a multiplicity of 8 and the point* $(1,1,2)$*. From Algorithm 6.4 we obtain the following univariate polynomials*

$$\begin{aligned} p_1 &= 0.707107\,x_1^2 - 0.707107\,x_1^3, \\ p_2 &= 0.894427\,x_2^4 - 0.447213\,x_2^5, \end{aligned}$$

with respective relative forward errors of 1.18×10^{-16} *and* 7.32×10^{-16}*. The derivatives of each of the univariate polynomials* p_1, p_2 *are computed by right-multiplying their coefficient vectors with the appropriate derivative operator. Algorithm 6.5 then obtains the following LCMs*

$$\begin{aligned} l_1 &= 0.324443\,x_1^2 - 0.811107\,x_1^3 + 0.486664\,x_1^4, \\ l_2 &= -0.650492\,x_2^4 + 0.731804\,x_2^5 - 0.203279\,x_2^6, \end{aligned}$$

with respective relative forward errors of 3.02×10^{-16} *and* 5.57×10^{-16}*. The square-free generators for the radical ideal are then the least-squares solutions*

$$\begin{aligned} p_{1,red} &= 0.707107\,x_1 - 0.707107\,x_1^2, \\ p_{2,red} &= 0.894427\,x_2 - 0.447213\,x_2^2, \end{aligned}$$

with relative forward errors of 4.82×10^{-16} *and* 1.77×10^{-16}*. Applying Algorithm 6.2 on the enlarged polynomial system then returns only 2 affine roots:* $(0,0)$ *and* $(1,2)$*.*

Observe that $p_{i,\mathrm{red}} = p_i$ implies that the roots have no multiplicities and therefore that the ideal $I = \langle f_1, \ldots, f_s \rangle$ is already radical. It is therefore easy to adjust Algorithm 6.7 such that it checks whether a polynomial ideal I is radical.

Chapter 7

Conclusions and Future Work

In this chapter, we provide a summary of the thesis and give an overview of open questions for future research.

7.1 Concluding Remarks

In this thesis the connection between numerical linear algebra and algebraic geometry is established. As a result, we now have a numerical linear algebra framework that allows us to solve a whole range of different kinds of problems from algebraic geometry.

In Chapter 2, the numerical linear algebra framework for multivariate polynomials was set up by describing 3 fundamental operations: the addition, multiplication and division of multivariate polynomials. It was shown how addition simply corresponds with vector addition and multiplication with computing a matrix vector product. For the multiplication operator, an upper bound for its condition number was derived. The division of multivariate polynomials is much more involved. A divisor matrix was introduced, which specifies two important subspaces. This lead to the insight that polynomial division corresponds with a vector decomposition over these two subspaces. Introducing the oblique projectors to compute this decomposition also lead to an elegant geometric interpretation of multivariate polynomial division. The nonuniqueness of both the quotient and remainder were also discussed and illustrated.

In Chapter 3, the most important matrix for applications, the Macaulay matrix, was introduced. Its size and density were discussed and illustrated. Furthermore,

3 important fundamental subspaces associated with the Macaulay matrix were introduced. The row space naturally lead to the ideal membership problem. The left null space was shown to be linked with the notion of syzygies and from their analysis the degree of regularity was derived. The final fundamental subspace discussed was the right null space, which was shown to be linked with the number of projective roots of a polynomial system.

Orthogonal bases for both the row space and right null space of the Macaulay matrix played an important part in all of our algorithms. Therefore, a fast recursive orthogonalization scheme for the Macaulay matrix was introduced in Chapter 4. This scheme allows to recursively update orthogonal bases for both the row space and right null space of the Macaulay matrix. Furthermore, it was shown how the resulting orthogonal basis for the row space retains a similar structure as the Macaulay matrix. Two implementations of the orthogonalization scheme were discussed: one using a sparse rank-revealing QR decomposition and one using a full SVD. The computational complexity for both implementations was investigated and it was shown that the sparse QR implementation has about d^3 times less operations. The sparse QR implementation was shown to be far superior to the full SVD with a 15 up to 119 times speedup and a gain of 12 up to 500 in required storage.

In Chapter 5, we introduced both the canonical and reduced canonical decomposition and the notion of a pure power. These concepts are linked with both a Gröbner basis and the number of roots of a polynomial system. An algorithm was presented that produces these decompositions through the repetitive computations of intersections of subspaces. In addition, the effect of noise on the coefficients of the polynomials on the resulting canonical decompositions was investigated. It was shown that computing canonical decompositions is in fact an ill-posed problem and that border bases do not suffer from this ill-posedness.

In Chapter 6, we discussed 7 different applications in the PNLA framework. The first application was the numerical computation of a Gröbner basis. Its relation with the reduced canonical decomposition was revealed for the zero-dimensional case. This also lead to a simple stop-criterion in terms of pure powers for the algorithm that computes the Gröbner basis. The next application discussed was affine root-finding. It was shown how the particular Vandermonde structure of the kernel of the Macaulay matrix together with a column compression allows to compute all affine roots from an eigenvalue problem. Again, the appearance of pure powers provides a stop-criterion in the affine root-finding algorithm. Next, the link between the occurrence of a Gröbner basis in $\mathcal{M}_d$ and solving the ideal membership problem was demonstrated. For the zero-dimensional case, the degree at which all pure powers are found also plays an important part in the stop-criterion. The next application was a new recursive method to do the full syzygy analysis and determine the polynomial expressions for $l(d), r(d)$ and

$c(d)$. The generalization of Gaussian elimination to the multivariate case was discussed next. The geometric interpretation of multivariate elimination as finding intersections of subspaces was also provided. This lead to the formulation of an elimination-algorithm that primarily uses principal angles between subspaces. We also discussed the computation of approximate LCMs and GCDs. We provided our own geometric definition of these approximate polynomials and made the link with the ϵ-GCD, which is more commonly used in the literature. Computation of the approximate LCM was shown to be equivalent with finding an intersection between two subspaces and hence an algorithm using principal angles was presented. The corresponding approximate GCD is then found from a sparse least-squares problem. Finally, we presented an algorithm that allows us to remove the multiplicities of the affine roots of a polynomial system. Its two main ingredients are multivariate elimination and the computation of an approximate LCM.

7.2 Future Research

Since this thesis constitutes the humble beginning of the PNLA framework, still many open theoretical and computational problems remain.

7.2.1 Numerical Analysis

Many numerical algorithms were formulated in this thesis. No formal numerical analysis was performed however. The seminal work by Stetter [78] can be rightfully seen as a first step of bringing numerical analysis into the domain of multivariate polynomials. Now, with the development of the PNLA framework, many existing tools of numerical analysis [44, 82] can be used to extend the work of Stetter.

Multivariate Polynomial Division. The link between multivariate polynomial division and oblique projections was established in this thesis. The main conceptual tool is the divisor matrix, from which two important subspaces can be defined. This lead to the new insight that the division of a multivariate polynomial by a set of divisors is in fact a vector decomposition into these two subspaces. This decomposition needs to be computed through the use of oblique projectors since the two subspaces are not orthogonal to one another. Although a numerical algorithm was proposed and implemented, a formal numerical analysis is still lacking. The recently published numerical results in [81] are a useful starting point in this respect. It is expected that this analysis will lead to a deeper understanding of the conditioning of multivariate polynomial division.

Intersection of subspaces. Like in the case of polynomial division, a geometrical concept plays a vital role in many algorithms: finding the intersection of subspaces.

The numerical algorithm that we developed depends heavily on the notion of principal angles, for which also perturbation results are available [14]. This in combination with the preliminary results for nonsingular matrices in the section on computing an approximate LCM/GCD can serve as a starting point for further studying the numerical properties of our algorithm.

Numerical rank. The notion of the numerical rank of a matrix is a well-defined concept through the SVD. The determination of the numerical rank is for all applications an important first step, since it specifies the dimension of the fundamental subspaces. Measurements from real world applications are noisy and hence the elements of the matrices will always be subject to some uncertainty. Starting from some well-known perturbation results on singular values [82], further research will need to investigate whether these uncertainties can be used to determine a suitable tolerance.

7.2.2 Curse of Dimensionality

A major bottleneck in solve large problems is the polynomial growth of the Macaulay matrix. One of the key research points is therefore obviously to look for ways to further exploit the sparsity and structure of the Macaulay matrix.

Exploiting the structure of the Macaulay matrix. The Macaulay matrix is extremely structured, a fact that has been partially exploited in Chapter 4, which resulted in gains of execution time of at least one order of magnitude. This first optimization however does not take into account the fact that always the same numerical values appear throughout the matrix. Methods for structured matrices such as Hankel and Toeplitz-matrices are already able to exploit the structure even further to reduce the computational complexity from $O(n^3)$ to $O(n^2)$ or even to $O(n\log^2(n))$ [11, 70]. These methods can serve as a starting point to further investigate the possibilities of exploiting the particular structure of the Macaulay matrix.

Exploiting sparsity of the Macaulay matrix. When multivariate polynomials are used as a modelling tool, they typically have only a few nonzero coefficients. This sparsity in the model translates itself to a very sparse Macaulay matrix. Indeed, its density drops very rapidly to less than 1%, which means that more than 99% of the matrix consists of zero elements. Not storing these zeros therefore constitutes a huge reduction in required storage space. A whole body of sparse matrix theory and methods [23] has been developed over the past five decades. This can also serve as a starting point to determine whether it is possible to exploit the a priori known sparsity pattern of the Macaulay matrix. This would not only reduce the amount of required storage space but is also expected to result in a significant gain in run time as well.

Appendix A

Numerical Linear Algebra

In this appendix we give a quick overview of the main concepts of numerical linear algebra that are used throughout the thesis. Good reference books on numerical linear algebra are [29, 40, 47].

A.1 Matrix Notation

Matrices are denoted by an upper case letter such as M and its (i, j)th element is by m_{ij}. We also use the MATLAB [72] notation $M(i : j, k : l)$ to denote the submatrix of M lying in rows i through j and columns k through l. A lower-case letter like p denotes both a multivariate polynomial and its corresponding coefficient vector. Coefficient vectors are always be row vectors. $\mathbb{R}$ and $\mathbb{C}$ denote the set of real and complex numbers respectively; $\mathbb{R}^n$, the set of n-dimensional real vectors, and $\mathbb{R}^{m\times n}$, the set of m-by-n real matrices. $\mathbb{N}_0^n$ denotes the set of n-tuples of natural numbers, including zero. M^T denotes the transpose of the matrix $M : (M^T)_{ij} = m_{ji}$ and M^{-1} its inverse. I_m denotes a square unit matrix of size m and the notation $\mathrm{diag}(a_1, \ldots, a_m)$ stands for a diagonal matrix with diagonal elements $a_1, \ldots, a_m$.

Remark A.1 *Assumption 1.1 implies that the entries of all matrices in this thesis are real numbers. Hence, there is no need for the conjugate transpose and all matrix factorizations will involve orthogonal matrices. One could allow complex entries at the cost of replacing the notion of orthogonality by the notion of unitarity and replacing the transpose by the conjugate transpose.*

A.2 Vector and Matrix Norms

Norms prove to be very useful in our framework. They are used to express bounds for the computation of certain quantities, like the condition number of the Macaulay matrix or the product of multivariate polynomials.

Definition A.1 *([40, p.52]) A vector norm on $\mathbb{R}^n$ is a function $||\cdot|| : \mathbb{R}^n \to \mathbb{R}$ that satisfies the following properties*

1. $||x|| \geq 0$, *and* $||x|| = 0$ *if and only if* $x = 0$,
2. $||\alpha x|| = |\alpha|\,||x||$ *for any real scalar* α,
3. $||x + y|| \leq ||x|| + ||y||$.

The two norms commonly used in this thesis are:

$$\begin{aligned} \text{1-norm:} \quad ||x||_1 &= |x_1| + \cdots + |x_n|, \\ \text{2-norm:} \quad ||x||_2 &= (|x_1|^2 + \cdots + |x_n|^2)^{1/2} = (x^T x)^{1/2}. \end{aligned}$$

A unit vector with respect to the norm $||\cdot||$ is a vector x that satisfies $||x|| = 1$. Sometimes it is necessary to express bounds in the 1-norm in terms of the 2-norm. Then the following relationship

$$||x||_2 \leq ||x||_1,$$

is used. Each of these two vector norms induces a corresponding matrix norm. A matrix norm is defined by simply stating that it is a function of $\mathbb{R}^{m\times n} \to \mathbb{R}$ that satisfies the same 3 properties as a vector norm. In this thesis, only one particular matrix norm is used.

Definition A.2 *Let $A \in \mathbb{R}^{m\times n}$, and $||\cdot||_2$ be the vector 2-norm. Then*

$$||A||_2 = \max_{x\neq 0,\, x\in\mathbb{R}^n} \frac{||Ax||_2}{||x||_2} \tag{A.1}$$

is called the spectral norm of the matrix A.

This matrix norm is given an interpretation in terms of bounds for multiplication of multivariate polynomials in Chapters 2 and 3.

A.3 Four Fundamental Subspaces

Given a matrix $A \in \mathbb{R}^{m \times n}$, then its associated fundamental subspaces [83] are:

$$\begin{array}{lrcl} \text{column space:} & \text{col}(A) & = & \{y \in \mathbb{R}^m \,|\, \exists\, x \in \mathbb{R}^n \,:\, y = A\,x\}, \\ \text{null space:} & \text{null}(A) & = & \{x \in \mathbb{R}^n \,|\, A\,x = 0\}, \\ \text{row space:} & \text{row}(A) & = & \{x \in \mathbb{R}^n \,|\, \exists\, z \in \mathbb{R}^m \,:\, x = z\,A\}, \\ \text{left null space:} & \text{null}(A^T) & = & \{z \in \mathbb{R}^m \,|\, z\,A = 0\}. \end{array}$$

From this definition one can see that the vector space $\text{null}(A)$ is orthogonal with respect to $\text{row}(A)$ and $\text{null}(A^T)$ is orthogonal with respect to $\text{col}(A)$. Except for the column space, all of these subspaces are given an interpretation in the context of multivariate polynomials in chapter 3. The following relations between the dimensions of the different subspaces hold:

$$\dim\left(\text{col}(A)\right) + \dim\left(\text{null}(A^T)\right) \;=\; m, \tag{A.2}$$

and

$$\dim\left(\text{row}(A)\right) + \dim\left(\text{null}(A)\right) \;=\; n. \tag{A.3}$$

The expressions (A.2) and (A.3) are called the rank-nullity theorem of A^T and A respectively.

A.4 Dual vector space

In Chapter 3, dual vector spaces play an important role in describing the roots of a polynomial system. Every finite dimensional vector space has a dual. The dual of a vector space $\mathcal{V}$ over a field k is a vector space $\mathcal{V}'$ of linear functionals $l : \mathcal{V} \to k$ that map each vector of $\mathcal{V}$ to an element of the field k. In our case, the field in question will always be $\mathbb{R}$. The following theorem relates the dimension of the vector space $\mathcal{V}$ with the dimension of its dual $\mathcal{V}'$.

Theorem A.1 *([43, p. 23]) If $\mathcal{V}$ is an n-dimensional vector space with a basis $\{v_1, \ldots, v_n\}$ then there is a uniquely determined basis $\{l'_1, \ldots, l'_n\}$ in $\mathcal{V}'$, with the property that for all $1 \leq i \leq n$ $l'_i(v_i) = 1$ and $l'_j(v_i) = 0$ for all $i \neq j$. Consequently the dual space of an n-dimensional space is n-dimensional.*

The basis $\{l'_1, \ldots, l'_n\}$ in $\mathcal{V}'$ is then called the dual of $\{v_1, \ldots, v_n\}$. The most interesting and useful dual vector space in this thesis will be the annihilator.

Definition A.3 *The annihilator $\mathcal{V}^o$ of a vector space $\mathcal{V}$ is the set of vectors $l \in \mathcal{V}'$ such that $l(v) = 0$ for all $v \in \mathcal{V}$.*

The dimension of the annihilator is given by the following theorem.

Theorem A.2 *[43, p. 26] If $\mathcal{M}$ is an r-dimensional subspace of an n-dimensional vector space $\mathcal{V}$, then $\mathcal{M}^o$ is an $(n-r)$-dimensional subspace of $\mathcal{V}'$.*

A.5 Moore-Penrose pseudoinverse

The Moore-Penrose pseudoinverse $A^\dagger$ of a matrix A is a generalization of the inverse matrix A^{-1}. The pseudoinverse is typically defined as follows.

Definition A.4 *([40, p. 257]) Let A be a $m \times n$ matrix, then its pseudoinverse $A^\dagger$ is the unique matrix satisfying*

$$\begin{aligned} AA^\dagger A &= A, \\ A^\dagger AA^\dagger &= A^\dagger, \\ (AA^\dagger)^T &= AA^\dagger, \\ (A^\dagger A)^T &= A^\dagger A. \end{aligned}$$

These four conditions are called the Moore-Penrose conditions.

A.6 Condition Number for matrix inversion and least squares

The condition number of a problem is a measure for the sensitivity of the solution to perturbations in the data. Some applications in this thesis, like for example polynomial division in Chapter 2 and the computation of an approximate GCD in Chapter 6, require solving (linear) least squares problems. This problem can hence be written as: given $A \in \mathbb{R}^{m \times n}$ with $m \geq n$ and $b \in \mathbb{R}^m$, find $x \in \mathbb{R}^n$ such that

$$A\,x \;=\; b.$$

If A is square and invertible, then the solution is given by

$$x \;=\; A^{-1}\,b.$$

The condition number κ for solving square linear systems is hence the condition number for matrix inversion. It is defined as

$$\kappa \;=\; ||A^{-1}||_2\,||A||_2.$$

Suppose now that the input of the problem, A and b, are perturbed with ΔA and Δb. The condition number κ then relates the relative error in the solution x to the relative error in the input by

$$\frac{||\Delta x||_2}{||x||_2} \leq \frac{\kappa}{1 - \kappa \frac{||\Delta A||_2}{||A||_2}} \left(\frac{||\Delta A||_2}{||A||_2} + \frac{||\Delta b||_2}{||b||_2} \right),$$

under the assumption that $\kappa \frac{||\Delta A||_2}{||A||_2} < 1$. This assumption, in fact, guarantees that $A + \Delta A$ is invertible, which we need for Δx to exist. Furthermore, if $\kappa \frac{||\Delta A||_2}{||A||_2} \ll 1$, then the relative error in the solution is indeed proportional to the relative error in the input.

When A is rectangular, things are a bit more complicated. First, its condition number for matrix inversion is now given by

$$\kappa \;=\; ||A^\dagger||_2 \, ||A||_2.$$

The condition number κ_{LS} for the least squares problem

$$\operatorname*{argmin}_{x \in \mathbb{R}^n} ||Ax - b||_2$$

is then given by the following theorem.

Theorem A.3 *([29, p. 117]) Suppose that $A \in \mathbb{R}^{m \times n}$ with $m \geq n$ and has full rank. Suppose that x minimizes $||Ax - b||_2$. Let $r = Ax - b$ be the residual. Let $\hat{x}$ minimize $||(A + \Delta A)\hat{x} - (b + \Delta b)||_2$. Assume $\epsilon = max(\frac{||\Delta A||_2}{||A||_2}, \frac{||\Delta b||_2}{||b||_2}) \leq \frac{1}{\kappa}$. Then*

$$\frac{||\Delta x||_2}{||x||_2} \leq \epsilon \left(\frac{2\,\kappa}{cos\theta} + tan\theta\, \kappa^2 \right) + O(\epsilon^2) = \epsilon\, \kappa_{LS} + O(\epsilon^2),$$

where $sin\theta = \frac{||r||_2}{||b||_2}$. In other words, θ is the angle between the vectors b and Ax and measures whether the residual norm $||r||_2$ is large (near $||b||_2$) or small (near 0).

The assumption that $\epsilon\,\kappa < 1$ is necessary, like in the square case, to guarantee that $A + \Delta A$ is of full rank, such that $\hat{x}$ is uniquely determined. The theorem can be interpreted as follows. If θ is 0 or very small, then the residual is small and the effective condition number is about $2\,\kappa$. This is similar to the square matrix case. If θ is not small but not close to $\pi/2$, the residual is moderately large, and the effective condition number can be much larger, κ^2. If θ is close to $\pi/2$, so the true solution is nearly zero, then the effective condition number becomes unbounded even if κ is small.

A.7 QR Decomposition

The first important matrix decomposition that we discuss is the QR decomposition. This is basically a numerically stable implementation of the Gram-Schmidt process.

Theorem A.4 *([29, p. 107]) Let A be a $m \times n$ matrix with $m \geq n$. Suppose that A has full column rank. Then there exists a unique $m \times m$ orthogonal matrix $Q\,(Q^T Q = I_m)$ and a unique $m \times n$ upper triangular matrix R with positive diagonals $r_{ii} > 0$ such that*

$$A = Q\,R.$$

An important application of the QR decomposition is that it provides orthogonal bases for the column space and its orthogonal complement. Since A is of full column rank the QR decomposition can be partitioned as

$$A = (Q_1\ Q_2) \begin{pmatrix} R_1 \\ R_2 \end{pmatrix},$$

where Q_1 consists of the first n columns of Q. It can then be shown that

$$\begin{array}{rcl} \mathrm{col}(Q_1) & = & \mathrm{col}(A), \\ \mathrm{col}(Q_2) & = & \mathrm{null}(A^{\perp}). \end{array}$$

Or in other words, the columns of Q_1 are an orthogonal basis for $\mathrm{col}(A)$ and the columns of Q_2 are an orthogonal basis for $\mathrm{null}(A^{\perp})$. When A is not of full column rank then Theorem A.4 needs a small adjustment. In exact arithmetic, if A had rank $r < n$ and its first r columns were independent, then its QR decomposition would look like

$$A = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix},$$

where R_{11} is $r \times r$ and nonsingular and R_{12} is $r \times (n-r)$. With roundoff, we might hope to compute

$$R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

with $||R_{22}||_2$ very small, on the order of $\epsilon\,||A||_2$. Since the first r columns are not necessarily linearly independent some column pivoting scheme is required. This means that we factorize $A\,P = Q\,R$, P being a permutation matrix. The aim of the pivoting scheme is to keep R_{11} as well-conditioned as possible and R_{22} as small as possible. Algorithms that implement such a pivoting scheme are called rank-revealing QR algorithms.

A.8 Singular Value Decomposition

The Singular Value Decomposition (SVD) can rightfully be called the mother of all matrix factorizations. Not only does it provide orthogonal bases for all fundamental subspaces, it also allows us to determine the rank and the spectral norm of a matrix, in addition to providing a way to compute principal angles.

Theorem A.5 *([29, p. 109]) For every real $m \times n$ matrix A with $m \geq n$ there exist orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ ($U^TU = UU^T = I_m$ and $V^TV = VV^T = I_n$) such that*

$$A \;=\; U\,\Sigma\,V^T,$$

and the $m \times n$ matrix $\Sigma = diag(\sigma_1, \ldots, \sigma_n)$ with $\sigma_1 \geq \ldots \geq \sigma_n \geq 0$. The columns of U are called the left singular vectors and the columns of V are called the right singular vectors. The σ_i's are called the singular values.

A first major application of the SVD is that it allows to determine the rank of a matrix and it provides orthogonal bases for all fundamental subspaces. If A has rank r, then it has r nonzero singular values and its SVD can be partitioned as

$$A \;=\; (U_1\; U_2)\begin{pmatrix} S_1 & 0 \\ 0 & 0 \end{pmatrix}(V_1\; V_2)^T. \tag{A.4}$$

We then have that

$$\begin{aligned} \mathrm{row}(A) &= \mathrm{col}(V_1), \\ \mathrm{null}(A) &= \mathrm{col}(V_2), \\ \mathrm{col}(A) &= \mathrm{col}(U_1), \\ \mathrm{null}(A^\perp) &= \mathrm{col}(U_2). \end{aligned}$$

Note from (A.4) that

$$\begin{aligned} A &= U_1\,S_1\,V_1^T, \\ &= \textstyle\sum_{i=1}^{r}\; \sigma_i\,u_i\,v_i^T. \end{aligned}$$

Or in other words, A can be written as a linear combination of r rank one matrices $u_i\,v_i^T$. This is usually called the reduced or compact SVD of A. The pseudoinverse of A is then easily computed as

$$A^\dagger = V_1\,S_1^{-1}\,U_1^T, \tag{A.5}$$

with $S^{-1} = \mathrm{diag}(1/\sigma_1, \ldots, 1/\sigma_n)$. This is easily verified by substituting (A.5) into the Moore-Penrose conditions. Another application of the SVD is the computation of the spectral norm of a matrix.

Lemma A.1 *Let $A \in \mathbb{R}^{m \times n}$ and $\sigma_1(A)$ denote the largest singular value of A then*

$$||A||_2 \;=\; \sigma_1(A).$$

If we choose x in the definition of the spectral norm such that $||x||_2 = 1$, then the largest singular value of A can also be interpreted as

$$\sigma_1(A) \;=\; \max_{||x||_2=1} ||Ax||_2.$$

This will be useful for Section 2.3.2 and Section 3.3, where bounds for the products of multivariate polynomials are derived. Furthermore, the SVD also allows us to compute principal angles between subspaces. This brings us to the next section.

A.9 Principal Angles

Here we provide the definition of principal angles [40, 49] between subspaces and their associated principal directions. These concepts play a fundamental role for the algorithms in this thesis.

Definition A.5 *The principal angles $0 \leq \theta_1 \leq \theta_2 \leq \ldots \theta_{min(d_1,d_2)} \leq \pi/2$ between the vector spaces S_1 and S_2 of dimension d_1 and d_2 respectively, and the corresponding principal directions $u_i \in S_1$ and $v_i \in S_2$ are defined recursively as*

$$cos(\theta_k) \;=\; \max_{u \in S_1} \max_{v \in S_2} u\, v^T = u_k\, v_k^T$$

subject to

$$\begin{aligned} ||u|| &= ||v|| = 1, \\ u\, u_i^T &= 0, \quad i = 1, \ldots, k-1, \\ v\, v_i^T &= 0, \quad i = 1, \ldots, k-1. \end{aligned}$$

Several numerical algorithms have been proposed for the computation of the principal angles and the corresponding principal directions [14]. The following theorem shows how one can determine the cosines of the principal angles and the corresponding principal directions by means of an SVD.

Theorem A.6 *([14, p. 582]) Assume that the columns of Q_1 and Q_2 are orthogonal bases for two subspaces of $\mathbb{R}^m$. Let*

$$A \;=\; Q_1^T\, Q_2,$$

and let the SVD of this $r_1 \times r_2$ matrix be

$$A \;=\; Y\,C\,Z^T, \quad C \;=\; diag(\sigma_1, \ldots, \sigma_{r_2}). \tag{A.6}$$

If we assume that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{r_2}$, then the principal angles and principal vectors associated with this pair of subspaces are given by

$$cos(\theta_k) = \sigma_k(A), \quad U = Q_1 Y, \quad V = Q_2 Z.$$

Computing principal angles smaller than 10^{-8} in double precision is impossible using Theorem A.6. This is easily seen from the second order approximation of the cosine of its Maclaurin series: $\cos(x) \approx 1 - x^2/2$. If $x < 10^{-8}$ then the $x^2/2$ term will be smaller than the machine precision $\epsilon \approx 2 \times 10^{-16}$ and hence $\cos(x)$ will be exactly 1. For small principal angles it is numerically better to compute the sines using the following Theorem.

Theorem A.7 *([14, p. 582-583] and [54, p. 6]) The singular values $\mu_1, \ldots, \mu_m$ of the matrix $Q_2 - Q_1\,Q_1^T\,Q_2$ are given by $\mu_k = \sqrt{1 - \sigma_k^2}$ where the σ_k are defined in* (A.6)*. Moreover, the principal angles satisfy the equalities $\theta_k = \arcsin(\mu_k)$. The right principal vectors can be computed as*

$$v_k \;=\; Q_2^T\,z_k, \quad k = 1, \ldots, r_2,$$

where z_k are the corresponding right singular vectors of $Q_2 - Q_1\,Q_1^T\,Q_2$.

Note that $Q_2 - Q_1\,Q_1^T\,Q_2$ is the orthogonal projection of $\mathrm{col}(Q_2)$ onto the orthogonal complement of $\mathrm{col}(Q_1)$. If an orthogonal basis N_1 for $\mathrm{null}(Q_1)$ is available, then one could compute the singular values and right singular vectors of $N_1^T Q_2$ instead.

A.10 Intersection of Subspaces

The main application of principal angles in this thesis is to find the intersection of subspaces. Theorem A.7 can be used to compute an orthogonal basis for $\mathrm{col}(Q_1) \cap \mathrm{col}(Q_2)$.

Theorem A.8 *([40, p. 604]) Let $sin(\theta_k), u_k, v_k$ be as described in Definition A.5 and Theorem A.7. Suppose the last s singular values $\mu_{m-s+1} = \ldots = \mu_m = 0$, then we have*

$$col(Q_1) \cap col(Q_2) \;=\; span\{u_{m-s+1}, \ldots, u_m\} \;=\; span\{v_{m-s+1}, \ldots, v_m\}.$$

In this thesis, it will almost always be that

$$Q_2 \;=\; P\,E$$

where P is a permutation matrix and $E = (I_{r_2}\ 0)^T$. This allows us to simplify the computation of the orthogonal basis for the intersection. Remember that the principal directions are given by the right singular vectors of $N_1^T Q_2$ where the columns of N_1 are an orthogonal basis for null(Q_1). We can therefore write

$$\begin{aligned} N_1^T Q_2 &= N_1^T P\,E, \\ &= (P^T N_1)^T\,E. \end{aligned}$$

Applying P^T to N_1 moves the r_2 rows of N_1 corresponding with nonzero rows of Q_2 to the top. We now partition $P^T\,N_1$ and substitute E by $(I_{r_2}\ 0)^T$

$$\begin{aligned} N_1^T Q_2 &= \begin{pmatrix} N_{11}^T & N_{12}^T \end{pmatrix} \begin{pmatrix} I_{r_2} \\ 0 \end{pmatrix}, \\ &= N_{11}^T. \end{aligned}$$

Remark A.2 *If we denote the vector containing the indices of these nonzero rows of Q_2 by j, then we can write $N_1^T Q_2$ in MATLAB notation as $N_1(j,:)^T$. Likewise using Theorem A.6, one needs to compute the SVD of $Q_1(j,:)^T$ to determine the cosines of the principal angles.*

Remark A.3 *Usually we will only need the principal vector v_m corresponding with the last zero singular value μ_m. An alternative for the computation of the full SVD of $N1^T\,Q_2$ would be to use an iterative approach (e.g. Krylov subspace methods).*

A.11 CS Decomposition

The CS decomposition relates the SVDs of the blocks of an orthogonal matrix partitioned into 2-by-2 form. There is also a connection between the CS decomposition and computing the intersection of subspaces. We first present the CS decomposition.

Theorem A.9 *([2, 79]) Let $Q \in \mathbb{R}^{m\times m}$ have orthonormal columns. Partition Q in the form*

$$Q = \begin{array}{c} \\ k \\ m-k \end{array} \begin{array}{c} \begin{array}{cc} k & m-k \end{array} \\ \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix} \end{array},$$

where $2k \leq m$. *Then there are orthogonal matrices* $U = diag(U_{11}, U_{22})$ *and* $V = diag(V_{11}, V_{22})$ *such that*

$$U^T Q V = \begin{array}{c} \\ k \\ k \\ m-2k \end{array} \begin{array}{c} \begin{array}{ccc} k & k & m-2k \end{array} \\ \begin{pmatrix} C & S & 0 \\ -S & C & 0 \\ 0 & 0 & I \end{pmatrix} \end{array},$$

where C *and* S *are nonnegative diagonal matrices satisfying*

$$C^2 + S^2 = I.$$

It is now easy to see that the C and S matrices contain the cosines and sines of the principal angles as discussed in the previous section. This follows from applying Theorem A.9 to $P\,Q = (Q_1\ N_1)$.

A.12 Projectors

A projector Π is an idempotent matrix ($\Pi^2 = \Pi$). It is completely determined by its column space $\mathcal{X}$ and its row space $\mathcal{Y}$.

Theorem A.10 *([81, p. 312]) Let* Π *be a projector. If the columns of* X *and* Y *are a basis for* $\mathcal{X} = col(\Pi)$ *and* $\mathcal{Y} = row(\Pi)$ *respectively, then* $Y^T X$ *is nonsingular and*

$$\Pi \;=\; X(Y^T X)^{-1}\, Y^T. \tag{A.7}$$

From (A.7) it is easy to see that when $X = Y$ then $\Pi = X(X^T X)^{-1} X^T$, which is the familiar expression for an orthogonal projector. Also observe that $||\Pi||_2 = 1$ for an orthogonal projector. Furthermore, if X is orthogonal then the expression for the projector simplifies to $\Pi = XX^T$. As soon as $\mathcal{X} \neq \mathcal{Y}$, then Π will be an oblique projector. Note that for an oblique projector we always have that $||\Pi||_2 > 1$. If the representation of (A.7) is used for a projector Π, then $\Pi\, a$ stands for the projection of the column vector a onto $\mathrm{col}(X) = \mathcal{X}$. The complement of $\Pi\, a$, defined as $(I - \Pi)a$, then lies in $\mathrm{null}(Y)$. From this it follows that the column vector a can be decomposed into $a = \Pi\, a + (I - \Pi)a$. It is also possible to write the oblique projector in terms of X and an orthogonal basis Z for $\mathrm{null}(Y)$:

$$\Pi \;=\; X\, \left[(I - ZZ^T)X\right]^{\dagger}\, (I - ZZ^T). \tag{A.8}$$

The use of the Moore-Penrose pseudoinverse is required in (A.8) since X might be rank deficient.

A.13 Sparse Matrices

A matrix is sparse when it has a large number of zero elements. What is meant by "large" is normally not very relevant. Indeed, for the definition of a sparse matrix a more computational point of view is adopted. Wilkinson defined a sparse matrix as "any matrix with enough zeros that it pays to take advantage of them" [39]. Or in other words, a matrix that allows special techniques to take advantage of the large number of zero elements. By avoiding arithmetic operations on zero elements, sparse matrix algorithms require less computer time. And, perhaps more importantly, by not storing many zero elements, sparse matrix data structures require less computer memory. MATLAB uses the compressed column format [23, p.8], which stores a sparse matrix as a collection of compressed columns. A compressed column is a list of row indices Ai and corresponding nonzero values Ax. The matrix

$$A = \begin{pmatrix} 4.5 & 0 & 3.2 & 0 \\ 3.1 & 2.9 & 0 & 0.9 \\ 0 & 1.7 & 3.0 & 0 \\ 3.5 & 0.4 & 0 & 1.0 \end{pmatrix},$$

is then stored as

$$\begin{array}{lccccccccccc} Ap: & [& 1, & & & 4, & & & 7, & & 9, & 10 &], \\ Ai: & [& 1, & 2, & 4, & 3, & 4, & 2, & 1, & 3, & 2, & 4 &], \\ Ax: & [& 4.5, & 3.1, & 3.5, & 1.7, & 0.4, & 2.9, & 3.2, & 3.0, & 0.9, & 1.0 &] & . \end{array}$$

The second column of A is then given by the row indices $Ai[Ap[2], \ldots, Ap[3]-1] = [3, 4, 2]$ with corresponding nonzero values $Ax[Ap[2] \ldots Ap[3]-1] = [1.7, 0.4, 2.9]$. Note that the row indices Ai do not even need to be sorted, as long as they point to the correct numerical values. The 10 entry in Ap is the total number of nonzero elements of A. Storing a matrix in compressed column format means that working with columns is easy, since they can be retrieved very fast, while working with rows is hard. Indeed, one needs to iterate over all columns and check whether the row has a particular nonzero element there.

In this thesis, we frequently need to determine the numerical rank of a given sparse matrix and compute orthogonal bases for both its row and null space. Ideally, one should use the SVD for this purpose but unfortunately, no sparse methods are available for computing the matrices Σ and V of the SVD. This is the reason a sparse rank-revealing multifrontal QR decomposition algorithm, SuiteSparseQR [24], will be used when the given matrix is stored in column compressed form. This method computes the QR factorization of $A \in \mathbb{R}^{m \times n}$ of rank r

$$A\,P \;=\; Q\,R,$$

where P is a permutation matrix such that $\mathrm{col}(Q(:, 1:r)) = \mathrm{col}(A)$ and the number of nonzero elements of R is minimized. Finding a column ordering optimal in this

sense is an NP-complete problem, i.e., it cannot be solved in polynomial time. Hence, a heuristic ordering algorithm is used. More details on the fill-reducing column ordering used in SuiteSparseQR, the column approximate minimum degree ordering, can be found in [25, 26]. This sparse QR algorithm, like all other sparse methods, combines numerics together with graph theory and consist of 2 phases: a symbolical and numerical phase. In the symbolical phase, the nonzero pattern of the R factor is predicted using graph theory and in the numerical phase, the numerical result is computed. SuiteSparseQR is also a multifrontal method. This means that the QR decomposition is reorganized into a sequence of partial factorizations of small dense matrices, called frontal matrices, and is well suited for parallelism. Indeed, the reduction of these dense matrices into upper triangular form can be done independently of one another. This results in a significant speedup of the algorithm, as is be demonstrated in Chapter 4.

Appendix B

Algebraic Geometry

In this appendix we give a quick overview of the main concepts of algebraic geometry that are used throughout the thesis. Excellent introductions to algebraic geometry are [21, 22].

B.1 Polynomials & Monomial Ordering

Definition B.1 *A monomial in $x_1, \ldots, x_n$ is a product of the form*

$$x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n},$$

where all of the exponents $\alpha_1, \ldots, \alpha_n$ are nonnegative integers. The total degree of this monomial is the sum $\alpha_1 + \ldots + \alpha_n$.

A simplification of notation can be introduced by writing α as n-tuples in $\mathbb{N}_0^n$, so that $\alpha = (\alpha_1, \ldots, \alpha_n)$. Then we set

$$x^\alpha \;=\; x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}.$$

In addition, the total degree of a polynomial will be denoted by $|\alpha|$.

Definition B.2 *A polynomial p in $x_1, \ldots, x_n$ with coefficients in $\mathbb{C}$ is a finite linear combination (with coefficients in $\mathbb{C}$) of monomials. We write a polynomial p in the form*

$$p = \sum_\alpha a_\alpha x^\alpha, a_\alpha \in \mathbb{C}, \tag{B.1}$$

where the sum is over a finite number of n-tuples $\alpha = (\alpha_1, \ldots, \alpha_n)$.

The ring of multivariate polynomials in n variables is denoted by $\mathcal{C}^n$. The subset of all multivariate polynomials of total degrees from 0 up to d forms a vector space, which is denoted by $\mathcal{C}_d^n$. A canonical basis for this vector space consists of all monomials from degree 0 up to d. A monomial $x^a = x_1^{a_1} \ \ldots \ x_n^{a_n}$ has a multidegree $(a_1, \ldots, a_n) \in \mathbb{N}_0^n$ and (total) degree $|a| = \sum_{i=1}^n a_i$. The degree of a polynomial p then corresponds with the highest total degree of all monomials of p. Note that we can reconstruct the monomial x^a from its multidegree a. Furthermore, any ordering $>$ we establish on the space $\mathbb{N}_0^n$ will give us an ordering on monomials: if $a > b$ according to this ordering, we will also say that $x^a > x^b$. Throughout this thesis the following monomial ordering will be used.

Definition B.3 *Degree negative lexicographic. Let a and $b \in \mathbb{N}_0^n$. We say $a >_{dnlex} b$ if*

$$|a| = \sum_{i=1}^{n} a_i > |b| = \sum_{i=1}^{n} b_i, \text{ or } |a| = |b| \text{ and } a >_{nlex} b$$

where $a >_{nlex} b$ if, in the vector difference $a - b \in \mathbb{Z}^n$, the leftmost nonzero entry is negative.

Example B.1 $(2,0,0) >_{dnlex} (0,0,1)$ *because* $|(2,0,0)| > |(0,0,1)|$ *which implies* $x_1^2 >_{dnlex} x_3$. *Likewise,* $(0,1,1) >_{dnlex} (2,0,0)$ *because* $(0,1,1) >_{nlex} (2,0,0)$ *and this implies that* $x_2 x_3 >_{dnlex} x_1^2$.

The ordering is graded because it first compares the degrees of the two monomials and applies the negative lexicographic ordering when there is a tie. The ordering is also multiplicative, which means that if $a <_{\text{dnlex}} b$ this implies that $ac <_{\text{dnlex}} bc$ for all $c \in \mathbb{N}_0^n$. Monomial orderings are important since results will typically depend on which ordering is used. Once a monomial ordering is specified, then the following useful concepts can be defined.

Definition B.4 *Let $f = \sum_\alpha a_\alpha x^\alpha$ be a nonzero polynomial in $\mathcal{C}^n$ and let $>$ be a monomial order.*

1. *The multidegree of f is*

$$multideg(f) \ = \ max\,(\alpha \in \mathbb{N}_0^n : a_\alpha \neq 0)$$

   where the maximum is taken with respect to $>$.

2. *The leading monomial of f is*

$$LM(f) \ = \ x^{multideg(f)}$$

   with coefficient 1.

B.2 Homogeneous Polynomials and Coordinates

A polynomial of degree d is homogeneous when every term is of degree d. A non-homogeneous polynomial can easily be made homogeneous by introducing an extra variable x_0.

Definition B.5 *Let $f \in \mathcal{C}_d^n$ of degree d, then its homogenization $f^h \in \mathcal{C}_d^{n+1}$ is the polynomial obtained by multiplying each term of f with a power of x_0 such that its degree becomes d.*

Example B.2 *Let*

$$f = x_1^2 + 9x_3 - 5,$$

then its homogenization is

$$f^h = x_1^2 + 9x_0x_3 - 5x_0^2.$$

The vector space of all homogeneous polynomials in $n+1$ variables and of degree d is denoted by $\mathcal{P}_d^n$. This vector space is spanned by all monomials in $n+1$ variables of degree d and hence

$$\dim\left(\mathcal{P}_d^n\right) = \binom{d+n}{n}. \tag{B.2}$$

In order to describe solution sets of systems of homogeneous polynomials, the projective space needs to be introduced. First, an equivalence relation $\sim$ on the nonzero points of $\mathbb{C}^{n+1}$ is defined by setting

$$(x_0', \ldots, x_n') \sim (x_0, \ldots, x_n)$$

if there is a nonzero $\lambda \in \mathbb{C}$ such that $(x_0', \ldots, x_n') = \lambda\,(x_0, \ldots, x_n)$.

Definition B.6 *The n-dimensional projective space $\mathbb{P}^n$ is the set of equivalence classes of $\sim$ on $\mathbb{C}^{n+1} - \{0\}$. Each nonzero $(n+1)$-tuple $(x_0, \ldots, x_n)$ defines a point p in $\mathbb{P}^n$, and we say that $(x_0, \ldots, x_n)$ are homogeneous coordinates of p.*

Note that the origin $(0, \ldots, 0) \in \mathbb{C}^{n+1}$ is not a point in the projective space. Because of the equivalence relation $\sim$, an infinite number of projective points $(x_0, \ldots, x_n)$ can be associated with 1 affine point $(x_1, \ldots, x_n)$. The affine space $\mathbb{C}^n$ can be retrieved as a 'slice' of the projective space:

$$\mathbb{C}^n \;=\; \{(x_0, x_1, \ldots, x_n) \in \mathbb{P}^n \;:\; x_0 = 1\}.$$

This means that given a projective point $p = (x_0, \ldots, x_n)$ with $x_0 \neq 0$, its affine counterpart is $(1, \frac{x_1}{x_0}, \ldots, \frac{x_n}{x_0})$. The projective points for which $x_0 = 0$ are called points at infinity.

B.3 Ideals

Another important algebraic concept is that of a polynomial ideal.

Definition B.7 *Let $f_1, \ldots, f_s \in \mathcal{C}^n$. Then we set*

$$\langle f_1, \ldots, f_s \rangle \;=\; \left\{ \sum_{i=1}^{s} h_i f_i : h_1, \ldots, h_s \in \mathcal{C}^n \right\} \tag{B.3}$$

and call it the ideal generated by $f_1, \ldots, f_s$.

The ideal hence contains all polynomial combinations $\sum_{i=1}^{s} h_i f_i$ without any constraints on the degrees of $h_1, \ldots, h_s$. For this reason, the polynomials $f_1, \ldots, f_s$ are also called the generators of the polynomial ideal. We will denote all polynomials of the ideal $\langle f_1, \ldots, f_s \rangle$ with a degree from 0 up to d by $\langle f_1, \ldots, f_s \rangle_d$. Observe that this implies that

$$\langle f_1, \ldots, f_s \rangle_d \subset \mathcal{C}_d^n$$

and $\langle f_1, \ldots, f_s \rangle_d$ is therefore also a vector space. The set of generators is not unique for a given polynomial ideal. An important set of generators for a given polynomial ideal $\langle f_1, \ldots, f_s \rangle$ is the Gröbner basis, which is explained below. For discussing the multiplicities of polynomial roots, the concept of a radical ideal is needed.

Definition B.8 *Let $I = \langle f_1, \ldots, f_s \rangle$ be a polynomial ideal. The radical of I, denoted $\sqrt{I}$, is the set*

$$\{f \,:\, f^m \in I \text{ for some integer } m \geq 1\}.$$

It can be shown that $\sqrt{I}$ is also an ideal and that $I \subset \sqrt{I}$. The link with multiplicities of roots is clear from the definition. If I has roots with multiplicities larger than 1, so-called multiple roots, then $\sqrt{I}$ will contain polynomials from which these multiplicities are removed. One therefore has to enlarge the polynomial ideal I to $\sqrt{I}$ in order to get rid of multiple roots. This is discussed in more detail in Chapters 3 and 6.

B.4 Varieties

Varieties are a geometrical concept and in a sense the dual of the notion of a polynomial ideal.

Definition B.9 *Let $f_1, \ldots, f_s \in \mathcal{C}^n$, then we set*

$$V = \{(a_1, \ldots, a_n) \in \mathbb{C}^n : f_i(a_1, \ldots, a_n) = 0 \text{ for all } 1 \leq i \leq s\}$$

and call it the affine variety of $f_1, \ldots, f_s$.

Or in other words, the affine variety is the set of affine roots of $f_1, \ldots, f_s$. It is easy to see that if V is a variety of a polynomial system $f_1, \ldots, f_s$, then it is also the variety of the whole polynomial ideal $\langle f_1, \ldots, f_s \rangle$. In the same vein, we can define a projective variety.

Definition B.10 *Let $f_1^h, \ldots, f_s^h \in \mathcal{P}^n$. Then*

$$V = \{(x_0, \ldots, x_n) \in \mathbb{P}^n : f_i^h(x_0, \ldots, x_n) = 0 \text{ for all } 1 \leq i \leq s\}$$

and call it the projective variety of $f_1^h, \ldots, f_s^h$.

Again, the projective variety is the set of roots of the homogeneous polynomials $f_1^h, \ldots, f_s^h$. Note that, since a projective variety is defined over homogeneous coordinates, it will never contain the origin in $\mathbb{P}^n$. Given a polynomial system $f_1, \ldots, f_s$ with corresponding affine variety V_a, one can easily transform everything into a projective setting. This is done by making each of the polynomials of $f_1, \ldots, f_s$ homogeneous. The homogeneous polynomial system $f_1^h, \ldots, f_s^h$ will then have a projective variety V. This process can be seen to extend V_a, such that it also contains roots at infinity. The affine part of V can be retrieved as the subset of projective roots for which $x_0 = 1$. For this reason we will sometimes say 'a polynomial system $f_1, \ldots, f_s$ with a projective variety'. By this then we mean the process of making $f_1, \ldots, f_s$ homogeneous and considering the projective variety of $f_1^h, \ldots, f_s^h$.

The notion of dimensionality of a variety can also be introduced. We will call a variety zero-dimensional if it consists of a finite number of points. The corresponding polynomial ideal is then also called zero-dimensional.

B.5 Gröbner Basis

As mentioned above, Gröbner bases are a particular set of generators of a polynomial ideal $\langle f_1, \ldots, f_s \rangle$. Their most useful property is also their defining property.

Definition B.11 *Given a set of multivariate polynomials $f_1, \ldots, f_s$ and a monomial ordering, then a finite set of polynomials $G = \{g_1, \ldots, g_t\} \in \langle f_1, \ldots, f_s \rangle$ is a Gröbner basis of $\langle f_1, \ldots, f_s \rangle$ if*

$$\forall\, p \in \langle f_1, \ldots, f_s \rangle, \exists\, g \in G \text{ such that } LM(g) \mid LM(p).$$

In addition, a Gröbner basis is called reduced if no monomial in any element of the basis is divisible by the leading monomials of the other elements of the basis.

Note from the definition that a Gröbner basis depends on the monomial ordering. The defining property of a Gröbner basis is hence that the leading monomial of every polynomial in $I = \langle f_1, \ldots, f_s \rangle$ is divisible by at least one of the leading monomials of the Gröbner basis. This can also be written as

$$\langle \mathrm{LM}(g_1), \ldots, \mathrm{LM}(g_t) \rangle \;=\; \langle \mathrm{LM}(I) \rangle,$$

where $\mathrm{LM}(I)$ is the set of all leading monomials in I. In order to determine whether a set of polynomials is a Gröbner basis, one needs the notion of an S-polynomial.

Definition B.12 *Let f_1, f_2 be nonzero multivariate polynomials and x^γ the least common multiple of their leading monomials. The S-polynomial of f_1, f_2 is the combination*

$$S(f_1, f_2) \;=\; \frac{x^\gamma}{LT(f_1)}\, f_1 - \frac{x^\gamma}{LT(f_2)}\, f_2$$

where $LT(f_1), LT(f_2)$ are the leading terms of f_1, f_2 with respect to a monomial ordering.

It is clear from this definition that an S-polynomial is designed to produce cancellation of the leading terms and that it has a degree of at most $\deg(x^\gamma)$. We can now state Buchberger's Criterion.

Theorem B.1 *([22, p.85]) Let I be a polynomial ideal. Then a basis $G = \{g_1, \ldots, g_t\}$ for I is a Gröbner basis for I if and only if for all pairs $i \neq j$, the remainder on division of $S(g_i, g_j)$ by G is zero.*

The notion of divisibility of polynomials is very important in relation to a Gröbner basis. Generalizing the polynomial long division to the multivariate case is rather straightforward and is discussed in [22, p. 61]. We discuss polynomial division in detail in Chapter 2. Now all ingredients are available to state Buchberger's Algorithm to find a Gröbner basis.

Algorithm B.1 *Buchberger's Algorithm*
Input: *polynomial system* $F = f_1, \ldots, f_s$
Output: *Gröbner basis* G *of* $\langle f_1, \ldots, f_s \rangle$

```
G ← F
G' ← ∅
while G' ≠ G do
  G' ← G
  for each pair p, q, p ≠ q in G' do
    S ← remainder on division of S(p, q) by G'
    if S ≠ 0 then
      G ← G ∪ S
    end if
  end for
end while
```

The algorithm is given in pseudo-code in Algorithm B.1. The candidate Gröbner basis G is first initialized to F. Then, for each pair of distinct polynomials in G, the S-polynomials are constructed and these are divided by G. Every nonzero remainder S on division by G is then added to G. The algorithm terminates when all S-polynomials of all pairs of G have zero remainders. This is effectively Buchberger's Criterion. The algorithm will always terminate after a finite number of steps due to the Ascending Chain Condition [22, p. 90]. The notion of a Gröbner basis plays a prominent role in most of the applications in Chapter 6.

B.6 Least Common Multiples and Greatest Common Divisors

The least common multiple (LCM) and greatest common divisor (GCD) of two multivariate polynomials are natural extensions of their univariate cousins. They appear in Chapter 6 where algorithms are presented on how each of them can be computed numerically.

Definition B.13 *A polynomial* $l \in \mathcal{C}^n$ *is called a LCM of* $f_1, f_2 \in \mathcal{C}^n$ *if*

1. f_1 *divides* l *and* f_2 *divides* l *and*
2. l *divides any polynomial which both* f_1 *and* f_2 *divide.*

Definition B.14 *A polynomial* $g \in \mathcal{C}^n$ *is called a GCD of* $f_1, f_2 \in \mathcal{C}^n$ *if*

1. g *divides* f_1 *and* f_2 *and*

2. if p is any polynomial which divides both f_1 and f_2, then p divides g.

As explained in the blind deconvolution problem, GCDs are an important concept. Another use of GCDs is in the computation of reduced polynomials. Only the concept for univariate problems is needed in this thesis. Remember that the fundamental theorem of algebra states that every non-zero univariate polynomial with complex coefficients has exactly as many complex roots as its degree, counting multiplicities. This means that any polynomial $p \in \mathcal{C}_d^1$ with r distinct roots $z_1, \ldots, z_r$ can be factorized as

$$p \;=\; c\,(x - z_1)^{m_1}\,(x - z_2)^{m_2}\;\ldots\;(x - z_r)^{m_r},$$

with $c \in \mathbb{C}$ and $m_1 + \ldots + m_r = d$. The reduced part of the polynomial p is then found by 'stripping away' the multiplicities of the roots.

Definition B.15 *The square-free (or reduced) part of a univariate polynomial $p \in \mathcal{C}_d^1$ with distinct roots $z_1, \ldots, z_r$ and multiplicities $m_1, \ldots, m_r$ respectively is the polynomial*

$$p_{red} \;=\; c\,(x - z_1)\,(x - z_2)\;\ldots\;(x - z_r).$$

These square-free parts of univariate polynomials play an important role in Chapters 3 and 6 when discussing multiplicities of roots.

References

[1] W. Auzinger and H. J. Stetter. An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations. In *Int. Conf. on Numerical Mathematics, Singapore 1988, Birkhäuser ISNM 86*, pages 11–30, 1988.

[2] Z. Bai. The CSD, GSVD, their Applications and Computations. *IMA preprint series 958,Institute for Mathematics and its Applications, University of Minnesota, MN*, 1992.

[3] K. Batselier, P. Dreesen, and B. De Moor. Maximum likelihood estimation and polynomial system solving. Proc of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning 2012 (ESANN 2012), pages 1–6, 2012.

[4] K. Batselier, P. Dreesen, and B. De Moor. Prediction Error Method Identification is an Eigenvalue Problem. Proc 16th IFAC Symposium on System Identification (SYSID 2012), pages 221–226, 2012.

[5] K. Batselier, P. Dreesen, and B. De Moor. A fast iterative orthogonalization scheme for the Macaulay matrix. Submitted, 2013.

[6] K. Batselier, P. Dreesen, and B. De Moor. A geometrical approach to finding multivariate approximate LCMs and GCDs. *Linear Algebra and its Applications*, 438(9):3618–3628, 2013.

[7] K. Batselier, P. Dreesen, and B. De Moor. Numerical Polynomial Algebra: The Canonical Decomposition and Numerical Gröbner Bases. Submitted, 2013.

[8] K. Batselier, P. Dreesen, and B. De Moor. The Geometry of Multivariate Polynomial Division and Elimination. *SIAM Journal on Matrix Analysis and Applications*, 34(1):102–125, 2013.

[9] D. Bayer and M. Stillman. On the complexity of computing syzygies. *Journal of Symbolic Computation*, 6(2-3):135 – 147, 1988.

[10] B. Beauzamy, E. Bombieri, P. Enflo, and H. L. Montgomery. Products Of Polynomials in Many Variables. *Journal Of Number Theory*, 36(2):219–245, October 1990.

[11] D. Bini and V. Y. Pan. *Polynomial and Matrix Computations (Vol. 1): Fundamental Algorithms*. Birkhauser Verlag, Basel, Switzerland, Switzerland, 1994.

[12] D. A. Bini and P. Boito. Structured matrix-based methods for polynomial ϵ-gcd: analysis and comparisons. In *Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation*, ISSAC '07, pages 9–16, New York, NY, USA, 2007. ACM.

[13] A. P. Bird. Cpg Islands as Gene Markers in the Vertebrate Nucleus. *Trends in Genetics*, 3(12):342–347, December 1987.

[14] Å. Björck and G. H. Golub. Numerical Methods for Computing Angles Between Linear Subspaces. *Mathematics of Computation*, 27(123):pp. 579–594, 1973.

[15] P. Boito. *Structured Matrix Based Methods for Approximate Polynomial GCD*. Edizioni della Normale, 2011.

[16] B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*. PhD thesis, Mathematical Institute, University of Innsbruck, Austria, 1965.

[17] B. Buchberger. A criterion for detecting unnecessary reductions in the construction of groebner bases. In *EUROSAM*, pages 3–21, 1979.

[18] P. Businger and G. H. Golub. Linear Least Squares Solutions by Householder Transformations. *Numerische Mathematik*, 7:269–276, 1965.

[19] R. M. Corless, P. M. Gianni, B. M. Trager, and S. M. Watt. The Singular Value Decomposition for Polynomial Systems. In *ACM International Symposium on Symbolic and Algebraic Computation*, pages 195–207, 1995.

[20] R. M. Corless, S. A. Watt, and L. H. Zhi. QR factoring to compute the GCD of univariate approximate polynomials. *IEEE Transactions on Signal Processing*, 52(12):3394–3402, December 2004.

[21] D. A. Cox, J. B. Little, and D. O'Shea. *Using Algebraic Geometry*. Graduate Texts in Mathematics. Springer-Verlag, Berlin-Heidelberg-New York, March 2005.

[22] D. A. Cox, J. B. Little, and D. O'Shea. *Ideals, Varieties and Algorithms*. Springer-Verlag, third edition, 2007.

[23] T. A. Davis. *Direct Methods for Sparse Linear Systems (Fundamentals of Algorithms 2)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2006.

[24] T. A. Davis. Algorithm 915, SuiteSparseQR: Multifrontal Multithreaded Rank-Revealing Sparse QR Factorization. *ACM Transactions on Mathematical Software*, 38(1), November 2011.

[25] T. A. Davis, J. R. Gilbert, S. I. Larimore, and E. G. Ng. Algorithm 836: Colamd, a column approximate minimum degree ordering algorithm. *ACM Trans. Math. Softw.*, 30(3):377–380, September 2004.

[26] T. A. Davis, J. R. Gilbert, S. I. Larimore, and E. G. Ng. A column approximate minimum degree ordering algorithm. *ACM Trans. Math. Softw.*, 30(3):353–376, September 2004.

[27] T. A. Davis and Y. Hu. The University of Florida Sparse Matrix Collection. *ACM Trans. Math. Softw.*, 38(1):1:1–1:25, December 2011.

[28] B. H. Dayton and Z. Zeng. Computing the multiplicity structure in solving polynomial systems. In *Proc. of ISSAC '05*, pages 116–123. ACM Press, 2005.

[29] J. W. Demmel. *Applied Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997.

[30] G. M. Diaz-Toca and L. Gonzalez-Vega. Computing greatest common divisors and squarefree decompositions through matrix methods: The parametric and approximate cases. *Linear Algebra and its Applications*, 412(2-3):222 – 246, 2006.

[31] J. A. Dieudonné. *History of Algebraic Geometry*. The Wadsworth mathematics series. Chapman & Hall, 1985.

[32] P. Dreesen. *Back to the Roots: Polynomial System Solving Using Linear Algebra*. PhD thesis, Faculty of Engineering, KU Leuven (Leuven, Belgium), 2013.

[33] P. Dreesen, K. Batselier, and B. De Moor. Back to the roots: Polynomial system solving, linear algebra, systems theory. Proc 16th IFAC Symposium on System Identification (SYSID 2012), pages 1203–1208, 2012.

[34] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis*. eleventh edition, 2006.

[35] I. Z. Emiris. *Sparse Elimination and Applications in Kinematics*. PhD thesis, Faculty of Computer Science, University of California (Berkeley), 1994.

[36] I. Z. Emiris, A. Galligo, and H. Lombardi. Certified approximate univariate GCDs. *Journal of Pure and Applied Algebra*, 117-118(0):229 – 251, 1997.

[37] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1–3):61–88, June 1999.

[38] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *Proceedings of the 2002 international symposium on Symbolic and algebraic computation*, ISSAC '02, pages 75–83, New York, NY, USA, 2002. ACM.

[39] J. R. Gilbert, C. Moler, and R. Schreiber. Sparse matrices in matlab: Design and implementation. *SIAM Journal of Matrix Analysis and Applications*, 13:333–356, 1992.

[40] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, October 1996.

[41] L. Gonzalez-Vega. An elementary proof of barnett's theorem about the greatest common divisor of several univariate polynomials. *Linear Algebra and its Applications*, 247(0):185 – 202, 1996.

[42] J. Grabmeier, E. Kaltofen, and V. Weispfenning. *Computer Algebra Handbook: Foundations, Applications, Systems*. With Cd-Rom, Demo Versions. Springer-Verlag GmbH, 2003.

[43] P. R. Halmos. *Finite-Dimensional Vector Spaces*. Undergraduate Texts in Mathematics. Springer, 1974.

[44] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2002.

[45] D. Hilbert. *Ueber die Theorie der algebraischen Formen*. Springer, 1890.

[46] H. Hironaka. Resolution of singularities of an algebraic variety over a field of characteristic zero: I. *Annals of Mathematics*, 79(1):pp. 109–203, 1964.

[47] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.

[48] C. R. Johnson. A Gersogrin-type Lower Bound for the Smallest Singular Value. *Linear Algebra and its Applications*, 112:1–7, January 1989.

[49] C. Jordan. Essai sur la géométrie à n dimensions. *Bulletin de la Société Mathématique*, 3:103–174, 1875.

[50] E. Kaltofen, J. P. May, Z. Yang, and L. Zhi. Approximate factorization of multivariate polynomials using singular value decomposition. *Journal of Symbolic Computation*, 43(5):359–376, May 2008.

[51] N. Karmarkar and Y. N. Lakshman. On approximate gcds of univariate polynomials. *Journal of Symbolic Computation*, 26(6):653–666, 1998.

[52] A. Kehrein and M. Kreuzer. Characterizations of border bases. *Journal of Pure and Applied Algebra*, 196(2-3):251 – 270, 2005.

[53] A. Kehrein and M. Kreuzer. Computing border bases. *Journal of Pure and Applied Algebra*, 205(2):279 – 295, 2006.

[54] A. V. Knyazev and M. E. Argentati. Principal Angles between Subspaces in an A-based Scalar Product: Algorithms and Perturbation Estimates. *SIAM Journal on Scientific Computing*, 23(6):2008–2040, May 17 2002.

[55] M. Kreuzer and L. Robbiano. *Computational Commutative Algebra 2*. Springer, 2005.

[56] D. Lazard. Gröbner-Bases, Gaussian elimination and resolution of systems of algebraic equations. In *EUROCAL*, pages 146–156, 1983.

[57] D. Lazard. A note on upper bounds for ideal-theoretic problems. *Journal of Symbolic Computation*, 13(3):231 – 233, 1992.

[58] T. Y. Li and Z. Zeng. A rank-revealing method with updating, downdating, and applications. *SIAM Journal on Matrix Analysis and Applications*, 26(4):918–946, 2005.

[59] L. Ljung. *System Identification: Theory for the User (2nd Edition)*. Prentice Hall, 2 edition, January 1999.

[60] F. S. Macaulay. On some formulae in elimination. *Proc. London Math. Soc.*, 35:3–27, 1902.

[61] F. S. Macaulay. *The algebraic theory of modular systems*. Cambridge University Press, 1916.

[62] E. W. Mayr and A. R. Meyer. The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in Mathematics*, 46(3):305 – 329, 1982.

[63] H. M. Moller and H. J. Stetter. Multivariate Polynomial Equations with Multiple Zeros solved by Matrix Eigenproblems. *Numerische Mathematik*, 70(3):311–329, May 1995.

[64] A. Morgan. *Solving polynomial systems using continuation for engineering and scientific problems*. Prentice-Hall, Englewood Cliffs, N.J., 1987.

[65] B. Mourrain. A new criterion for normal form algorithms. In *Proc. AAECC, volume 1719 of LNCS*, pages 430–443. Springer, 1999.

[66] B. Mourrain and V. Y. Pan. Multivariate polynomials, duality, and structured matrices. *Journal of Complexity*, 16(1):110 – 180, 2000.

[67] K. Nagasaka. Backward Error Analysis of Approximate Gröbner Basis. Preprint, 2012.

[68] M. T. Noda and T. Sasaki. Approximate gcd and its application to ill-conditioned equations. *Journal of Computational and Applied Mathematics*, 38(1-3):335 – 351, 1991.

[69] L. Pachter and B. Sturmfels, editors. *Algebraic Statistics for Computational Biology*. Cambridge University Press, August 2005.

[70] V. Y. Pan. *Structured matrices and polynomials: unified superfast algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[71] S. U. Pillai and B. Liang. Blind image deconvolution using a robust gcd approach. *Image Processing, IEEE Transactions on*, 8(2):295 –301, feb 1999.

[72] MATLAB R2012a. The Mathworks Inc., 2012. Natick, Massachusetts.

[73] A. Schönhage. Quasi-gcd computations. *Journal of Complexity*, 1(1):118 – 137, 1985.

[74] I. Schur. Bemerkungen zur Theorie der beschränkten Bilinearformen mit unendlich vielen Veränderlichen. *Journal für die Reine und Angewandte Mathematik*, 140:1 – 28, 1911.

[75] A. J. Sommese, J. Verschelde, and C. W. Wampler. Numerical decomposition of the solution sets of polynomial systems into irreducible components. *SIAM Journal on Numerical Analysis*, 38(6):2022–2046, 2001.

[76] A. J. Sommese and C. W. Wampler. Numerical Algebraic Geometry. In The Mathematics of Numerical Analysis. In M. Shub J.Renegar and S. Smale, editors, *Lectures in Applied Mathematics. Proceedings of the AMS-SIAM Summer Seminar in Applied Mathematics*, pages 749–763. AMS, 1996.

[77] H. J. Stetter. Matrix eigenproblems are at the heart of polynomial system solving. *SIGSAM Bulletin*, 30(4):22–5, December 1996.

[78] H. J. Stetter. *Numerical Polynomial Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2004.

[79] G. W. Stewart. Computing the CS decomposition of a partitioned orthonormal matrix. *Numerische Mathematik*, 40(3):297–306, October 1982.

[80] G. W. Stewart. Perturbation Theory for the Singular Value Decomposition. In *SVD and Signal Processing, II: Algorithms, Analysis and Applications*, pages 99–109. Elsevier, 1990.

[81] G. W. Stewart. On the Numerical Analysis of Oblique Projectors. *SIAM Journal on Matrix Analysis and Applications*, 32(1):309–348, 2011.

[82] G. W. Stewart and J.-G. Sun. *Matrix Perturbation Theory (Computer Science and Scientific Computing)*. Academic Press, June 1990.

[83] G. Strang. The fundamental theorem of linear algebra. *The American Mathematical Monthly*, 100(9):pp. 848–855, 1993.

[84] J. J. Sylvester. On the Intersections, Contacts, and Other Correlations of Two Conics Expressed by Indeterminate Coordinates. *Cambridge and Dublin Mathematical Journal*, pages 262–282, 1850.

[85] J. J. Sylvester. On a theory of syzygetic relations of two rational integral functions, comprising an application to the theory of Sturm's function and that of the greatest algebraical common measure. *Trans. Roy. Soc. Lond.*, 1853.

[86] P. Van Hentenryck, D. McAllester, and D. Kapur. Solving Polynomial Systems Using a Branch and Prune Approach. *SIAM Journal on Numerical Analysis*, 34(2):797–827, April 1997.

[87] J. Verschelde. Algorithm 795: PHCpack: a general-purpose solver for polynomial systems by homotopy continuation. *ACM Trans. Math. Softw.*, 25(2):251–276, 1999.

[88] H. Weyl. Das asymptotische Verteilungsgesetz der Eigenwerte linearer partieller Differentialgleichungen (mit einer Anwendung auf die Theorie der Hohlraumstrahlung). *Mathematische Annalen*, 71:441–479, 1912.

[89] C. K. Yap. A New Lower Bound Construction for Commutative Thue Systems with Applications. *Journal Of Symbolic Computation*, 12(1):1–27, July 1991.

[90] Z. Zeng. The approximate GCD of inexact polynomials part II: a multivariate algorithm. In *Proceedings of the 2004 international symposium on Symbolic and algebraic computation*, pages 320–327, 2004.

[91] Z. Zeng. ApaTools: A Software Toolbox for Approximate Polynomial Algebra. In Michael Stillman, Jan Verschelde, and Nobuki Takayama, editors, *Software for Algebraic Geometry*, volume 148 of *The IMA Volumes in Mathematics and its Applications*, pages 149–167. Springer New York, 2008.

[92] Z. Zeng. A numerical elimination method for polynomial computations. *Theor. Comput. Sci.*, 409:318–331, December 2008.

[93] Z. Zeng. The closedness subspace method for computing the multiplicity structure of a polynomial system. In *Interactions of Classical and Numerical Algebraic*, 2009.

[94] Z. Zeng and B. H. Dayton. The approximate GCD of inexact polynomials Part I: a univariate algorithm, 2004. preprint.

Publications by the author

Journal papers

- K. Batselier, P. Dreesen, and B. De Moor, "The Geometry of Multivariate Polynomial Division and Elimination", SIAM Journal on Matrix Analysis and Applications, vol. 34, no. 1, February 2013, pp. 102-125.
- K. Batselier, P. Dreesen, and B. De Moor, "A geometrical approach to finding multivariate approximate LCMs and GCDs", Linear Algebra and its Applications, vol. 438, no. 9, May 2013, pp. 3618-3628.
- K. Batselier, P. Dreesen, and B. De Moor, "The Canonical Decomposition of $\mathcal{C}_d^n$ and Numerical Gröbner and Border Bases", submitted.
- K. Batselier, P. Dreesen, and B. De Moor, "A fast recursive orthogonalization scheme for the Macaulay matrix", submitted.

International Conference papers

- D. Geebelen, K. Batselier, P. Dreesen, M. Signoretto, J.A.K. Suykens, B. De Moor, and J. Vandewalle, "Joint Regression and Linear Combination of Time Series for Optimal Prediction", in Proc. of the European Symposium on Artificial Neural Networks (ESANN 2012), Brugge, Belgium, April 2012.
- P. Dreesen, K. Batselier, and B. De Moor, "Weighted/Structured Total Least Squares Problems and Polynomial System Solving", in Proc. of the 20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2012), Brugge, Belgium, Apr. 2012, pp. 351-356.
- K. Batselier, P. Dreesen, and B. De Moor, "Prediction Error Method Identification is an Eigenvalue Problem", in Proc. of the 16th IFAC

Symposium on System Identification (SYSID 2012), Brussels, Belgium, July 2012.

- P. Dreesen, K. Batselier, and B. De Moor, "Back to the Roots: Polynomial System Solving, Linear Algebra, Systems Theory", in Proc. of the 16th IFAC Symposium on System Identification (SYSID 2012), Brussels, Belgium, July 2012, pp. 1203-1208.

- K. Batselier, P. Dreesen, and B. De Moor, "Maximum Likelihood Estimation and Polynomial System Solving", in Proc. of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning 2012 (ESANN 2012), Brugge, Belgium, April 2012.

Curriculum vitae

Kim Batselier was born in Geraardsbergen in 1981. He received his Master of Science degree in Electrical Engineering from the KU Leuven, Belgium in 2005. He then worked for BIORICS, a spin-off of the research unit Measure, Model and Manage Bio Responses (M3-BIORES), of the Department Biosystems of the KU Leuven. His main work within BIORICS was the development of real-time monitoring algorithms that use system identification techniques to monitor professional football players during their training. This was in collaboration with Milan Lab, the research centre of AC Milan. In 2009 he started to pursue a Ph.D. degree at the STADIUS research unit of the Department of Electrical Engineering of the KU Leuven, under the supervision of Prof. Bart De Moor. His Ph.D. constituted the development of a numerical linear algebra framework in which problems with multivariate polynomials are solved. His main research interests are numerical linear and polynomial algebra, systems theory, system identification and signal processing.