KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT WETENSCHAPPEN
FACULTEIT TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT COMPUTERWETENSCHAPPEN
Celestijnenlaan 200 A — 3001 Leuven

HOMOTOPY CONTINUATION METHODS FOR SOLVING POLYNOMIAL SYSTEMS

Promotor :
Prof. Dr. ir. A. Haegemans
Co-promotor :
Prof. Dr. H. Van de Vel

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de Informatica

door

Jan VERSCHELDE

May 1996

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT WETENSCHAPPEN
FACULTEIT TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT COMPUTERWETENSCHAPPEN
Celestijnenlaan 200 A — 3001 Leuven

HOMOTOPY CONTINUATION METHODS FOR SOLVING POLYNOMIAL SYSTEMS

Jury :
Voorzitter Prof. Dr. ir. Y.D. Willems
Prof. Dr. ir. A. Haegemans, promotor
Prof. Dr. H. Van de Vel, co-promotor
Prof. Dr. ir. R. Cools
Prof. Dr. J. Denef
Prof. Dr. T.Y. Li
(Michigan State University, U.S.A.)

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de Informatica

door

Jan VERSCHELDE

U.D.C. 681.3*F21

May 1996

Jan VERSCHELDE — **HOMOTOPY CONTINUATION METHODS** FOR SOLVING POLYNOMIAL SYSTEMS — May 1996

Instructies voor de zetter:

Formaat: 16×24 cm
Totaal aantal vellen papier
(incl. 2 schutbladen, excl. harde kaft): 144
Op voorste kaft, blauwe zone bovenaan van 6 cm

Reductie van origineel: 80 %
Nederlandse samenvatting, blz. vii-xxxii, op
"blauw pastel" papier

Aantal exemplaren: 60

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT WETENSCHAPPEN
FACULTEIT TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT COMPUTERWETENSCHAPPEN
Celestijnenlaan 200 A — 3001 Leuven

HOMOTOPY CONTINUATION METHODS FOR SOLVING POLYNOMIAL SYSTEMS

Jury:
Voorzitter: Prof. Dr. ir. Y.D. Willems
Prof. Dr. A. Haegemans, promotor
Prof. Dr. H. Van de Vel, co-promotor
Prof. Dr. ir. R. Cools
Prof. Dr. J. Denef
Prof. Dr. T.Y. Li
(Michigan State University, U.S.A.)

Proefschrift voorgedragen tot het behalen van het doctoraat in de Informatica

door

Jan VERSCHELDE

U.D.C. 681.3*F21

May 1996

Homotopy Continuation Methods for Solving Polynomial Systems

Jan Verschelde

abstract

Homotopy continuation methods have proven to be reliable for computing numerically approximations to all isolated solutions of polynomial systems. The performance of homotopy continuation methods depends heavily on the root count used for estimating the number of isolated solutions. Therefore, we have directed our attention to the development of root counting methods.

Two different classes of homotopy methods have been investigated, based on different notions of complexity: degree and geometric complexity. Product homotopy methods are based on the highest degree terms of each monomials. Polyhedral homotopies make use of the Newton polytopes of the polynomials in the system. While the first class is well suited for dense systems, sparse systems have to be solved by polyhedral homotopy methods.

To improve the performance of the classical homotopy based on the total degree, we proposed to apply nonlinear reduction. Hereby the system is reformulated into an equivalent one with lower degree polynomials by computing S-polynomials. By casting Bézout's theorem in multi-homogeneous space, frequently a more accurate root count than the total degree can be obtained. To exploit the multi-homogeneous structure, we first constructed interpolating multi-homogeneous homotopies. Later we derived more efficient Bézout number computations by exploiting the analogy with the solution of multi-homogeneous linear-product systems. This insight led us to a generalization of Bézout's theorem, where we used a set structure as basis for the root count. This root counting method proved to be more flexible and effective, especially for symmetric polynomial systems.

We have developed several polyhedral homotopy methods by elaborating four different lifting algorithms. First we implemented the algorithm Bernshteĭn gave in his proof. Nowadays, we call this implicit lifting. Later we applied the static lifting method of Huber and Sturmfels to symmetric polynomial systems. Symmetry forced us to extend static lifting with a recursive mechanism. To control the height of the lifting and to obtain consequently a stable polyhedral continuation, we developed the dynamic lifting algorithm. This algorithm enables incremental polynomial system solving.

We have implemented all investigated methods in Ada. The result is an integrated work bench whose tools offer various root counting and homotopy methods. We have tested the package extensively on a wide range of polynomial systems derived from various application fields. Hereby we have provided practical evidence that the investigated homotopy continuation methods are effective in solving polynomial systems.

Preface

The most plain question is always why? Why make a PhD? Why become a researcher? In my teens I was attracted by the pure beauty of mathematical ideas and concepts. To be able to make a computer obtain results that went beyond our human possibilities fascinated me. Bringing mathematics alive and at work became my vocation.

I thank my adviser Prof. Ann Haegemans for the opportunity to work in her group. I also want to thank my co-adviser Prof. Hugo Van de Vel. I am grateful to the other members of the jury, Prof. Ronald Cools, Prof. Jan Denef, Prof. Tien-Yien Li and Prof. Yves Willems for the interest they showed in my work.

I wish to thank all my co-authors for many inspiring collaborations.

As counceler of my Bachelor's thesis, Marc Beckers introduced me to homotopy continuation and explained multi-homogenization to me. From him I learned about the problems with homotopy methods. He did a splendid job, thanks to him, I could start this project.

I am much indebted to Ann Haegemans. She encouraged my projects and left me the freedom to undertake this work. Her help was invaluable on starting to write up this dissertation. Without her thorough reading, this thesis would have contained a lot more mistakes. I thank Ronald Cools for his continuing friendly assistance concerning software. I am grateful that he brought me into contact with Karin Gatermann. His comments on the first draft of the dissertation were most useful.

I am largely indebted to Pierre Verlinden. The years we shared the office were most fruitful with many discoveries. Pierre explained Bernshteĭn's paper to me and helped me with lots of theoretical problems. We had many inspiring discussions on the stability of reduction, symmetric lifting and the connectivity conjecture.

Karin Gatermann brought the paper of Canny and Rojas to my attention, which led to a major turn in my work. Together we explored polyhedral homotopy methods. I thank her for the patience she had with me reading all my 'proofs' of the connectivity conjecture. Our collaboration greatly influenced my mathematical writing style.

I wish to thank Tom Michiels for his motivation to continue the line of this work. We had lots of interesting discussions on the secondary polytope. I also want to thank Peter Kravanja for joining our group.

I was fortunate for the many stimulating contacts with fellow researchers.

I am largely indebted to Tien-Yien Li, for his interest in my research. The short visits I could make to MSU always stimulated further research. I learned a lot from his thorough and profound questioning. I also like to thank Xiaoshen Wang for the discussions at MSU. We had lots of inspiring e-mail conversations.

Special thanks go to Ioannis Emiris, Birk Huber and Maurice Rojas for the mixed-volume sessions we organized at several conferences. I thank them for being great colleagues. I learned a lot from them. Ioannis explained me his pruning methods. From Birk I learned about the Cayley trick. We also had lots of stimulating discussions on software. I am grateful to Maurice for his patience to make his mathematical ideas more clearly to me.

I thank Charles Wampler for his interest in polyhedral homotopy continuation. In particular, his use of my program was very stimulating to me.

I am much indebted to Paul Pedersen who introduced me to fans and who allowed me to use his idea in the writing of dynamic lifting. I wish to thank Bernd Sturmfels for explaining the polyhedral homotopy methods to us. I am grateful to Peter Gritzmann and Alexander Hufnagel, they introduced me to the class of $\#P$-hard problems.

I wish to thank Jan Janssen, Karl Meerbergen, Pierre Verlinden and Hilde Vanaenroyde for the cordial and stimulating atmosphere during the time I shared the office with them.

I am much indebted to Dirk Craeynest for his enthusiasm and dedication to Ada and for introducing my to the Ada world. Thanks to him, I started to use the gnu-ada compiler. I also want to thank Karel De Vlaminck and Yvan Barbaix for putting their Ada technology at my disposal. I am grateful to the technical staff of our department, in particular to Anita Ceulemans for the quick installations of the latest versions of the gnu-ada compiler. Thanks to our secretaries Denise Brams, Karin Michiels, Margot Peeters, and Lieve Swinnen for taking over administrative burdens.

I could benefit from financial support of the Nationaal Fonds voor Wetenschappelijk Onderzoek and Vlaamse Leergangen to attend conferences. As graduate student I was fortunate to receive support from the American Mathematical Society to attend their conferences. I also acknowledge the support of the European Human Capability Program to attend a Euroconference in Crete.

Finally, I want to thank my parents for their continuing support.

Jan Verschelde
Heverlee, April 1996

contents

List of Symbols

The following list contains the notations used. The page numbers refer to the page where the symbol occurs for the first time. In case we use the same symbol to denote another object, the context is totally different. Next we give the only two cases where this occurs. The S stands for an array of sets in Chapter 4 and denotes in Chapter 5 a subdivision. In Chapter 4, the symbol $\mathfrak{S}$ is used for a set structure, while in Chapter 5, it denotes the collection of mixed cells.

symbol	explanation	page
$\mathbb{N}$	set of natural numbers	2
$\mathbb{C}$	field of complex numbers	2
$\mathbb{C}_0$	$\mathbb{C} \setminus \{0\}$	2
n	$n \in \mathbb{N}$, dimension of a vector space	2
m	multiplicity of an isolated solution, cyclic number	2
d	$d \in \mathbb{N}$, exponent	2
c	$c \in \mathbb{C}$, coefficient of a monomial	2
$\mathbf{d}$	$\mathbf{d} = (d_1, d_2, \ldots, d_n) \in \mathbb{N}^n$	2
$\mathbf{x}$	$\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{C}^n$	2
$\mathbb{C}[\mathbf{x}]$	ring of complex multivariate polynomials	2
$\mathbf{x}^\mathbf{d}$	$\mathbf{x}^\mathbf{d} = x_1^{d_1} x_2^{d_2} \cdots x_n^{d_n}$, monomial in multi-index notation	2
f	multivariate polynomial in $\mathbb{C}[\mathbf{x}]$	2
F	polynomial system	2
J_F	Jacobian matrix of F	2
$\det(J_F)$	Jacobian of F	2
$\mathbf{x}^*$	solution of a polynomial system	2
$\mathbb{N}_0$	$\mathbb{N} \setminus \{0\}$	6
ϵ	$\epsilon > 0$, accuracy requirement	6
$\lvert\lvert\mathbf{x}\rvert\rvert$	vector norm for the vector $\mathbf{x}$	6
H	homotopy	7
t	continuation parameter	7
$F^{(0)}$	start system	7
$F^{(1)}$	target system	7
$\mathbf{z}$	$\mathbf{z} = (z_0, z_1, \ldots, z_n) \in \mathbb{P}^n$	9
$\mathbb{P}$	projective space	9
$\deg(f)$	degree of f	10
D	total degree of a polynomial system	10

g	group action, $g \in G$	64
(G, V, W)	(G, V, W)-symmetric structure	64
V, W	matrix representations of G	64
V_g, W_g	$V_g, W_g \in \mathbb{C}^{n \times n}$ matrix representations of g	64
$\mathcal{O}_{\mathbf{x}}$	orbit of the vector $\mathbf{x}$	64
$D(T, \mathfrak{S}, G)$	$D(T, \mathfrak{S}, G) = [(k, l)]$, dependency structure of T	66
$C(\mathfrak{S}, G)$	covering of the set structure	66
$\mathfrak{T}$	$\mathfrak{T} = (\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_n)$, template, $\mathcal{T}_k = [\mathbf{a}^{(kl)}]$	68
$n_{\mathfrak{T}}$	number of free coefficients	68
$A = \mathrm{supp}(f)$	support of Laurent polynomial f	78
$P = \mathrm{conv}(A)$	Newton polytope of f	78
$\mathcal{A}$	$\mathcal{A} = (A_1, A_2, \dots, A_r)$, supports of F	78
$\mathcal{P}$	$\mathcal{P} = (P_1, P_2, \dots, P_r)$, Newton polytopes of F	78
$\mathrm{vol}_n(P)$	volume of P	78
$V_n(P)$	mixed volume of $\mathcal{P}$	78
p	support function of polytope P	80
$\langle \cdot, \cdot \rangle$	standard inner product of two vectors	80
$\mathbf{v}$	$\mathbf{v} = [v_1, v_2, \dots, v_n]^T$, inner or outer normal	81
$\partial_{\mathbf{v}} A$	spans face of polytope	81
$\partial_{\mathbf{v}} P$	$\partial_{\mathbf{v}} P = \mathrm{conv}(\partial_{\mathbf{v}} A)$, face of polytope P	81
$\dim(A)$	dimension of the set A	81
$\partial_{\mathbf{v}} f$	face polynomial	81
$\gcd(a, b)$	greatest common divisor of $a, b \in \mathbb{Z}$	83
U	$U \in \mathbb{Z}^{n \times n}$, $\det(U) = \pm 1$, unimodular matrix	83
$\mathbf{x} = \mathbf{y}^U$	unimodular transformation of monomials	85
$\mathbb{R}$	field of real numbers	85
$\mathcal{V}$	list of outer normals contributing to $V_n(\mathcal{P})$	86
S	$S = \{C_1, C_2, \dots, C_m\}$, a subdivision of A	96
C	$C \subset A$, $C \in S$, a cell in a subdivision	96
ω	lifting function	96
$\widehat{\mathbf{a}}$	lifted vector, $a_{n+1} = \omega(\mathbf{a})$	96
$\widehat{A}$	lifted support	96
$\widehat{P}$	lifted polytope, $\widehat{P} = \mathrm{conv}(\widehat{A})$	96
S_ω	regular subdivision induced by ω	96
$\widehat{S}_\omega$	collection of lifted cells in S_ω	96
$C_{\mathbf{v}}$	cell characterized by inner normal $\mathbf{v}$, $\widehat{C}_{\mathbf{v}} = \partial_{\mathbf{v}} \widehat{A}$	96
$\widehat{C}_{\mathbf{v}}$	lifted cell with inner normal $\mathbf{v}$	96
$\triangle$	a triangulation of A	97
$\triangle_\omega$	a regular triangulation of A, induced by ω	97
r	number of different supports of F	98
$\mathbf{k} \in \mathbb{N}^r$	occurrences of the supports of F, $n = \sum_{i=1}^r k_i$	98
$\mathcal{C}$	$\mathcal{C} = (C_1, C_2, \dots, C_r)$, $\mathcal{C} \subset \mathcal{A}$, $\mathcal{C} \in S$, cell in subdivision of $\mathcal{A}$	98
$\mathrm{type}(\mathcal{C})$	vector in $\mathbb{N}^r$ indicating the dimensions of C_i	98
$\widehat{\mathcal{A}}$	$\widehat{\mathcal{A}} = (\widehat{A}_1, \widehat{A}_2, \dots, \widehat{A}_r)$, lifted supports of F	99
$\widehat{\mathcal{C}}$	lifted cell $\mathcal{C} \in S$	99

$\widehat{\mathcal{P}}$	$\widehat{\mathcal{P}} = (\widehat{P}_1, \widehat{P}_2, \ldots, \widehat{P}_r)$, lifted Newton polytopes of F	99
$\widehat{\mathcal{C}}_{\mathbf{v}}$	cell characterized by inner normal $\mathbf{v}$, $\widehat{\mathcal{C}}_{\mathbf{v}} = \partial_{\mathbf{v}} \widehat{\mathcal{A}}$	99
$\mathrm{vol}_n(P, \mathbf{k})$	volume of P, w.r.t. type of mixture	99
$V_n(P, \mathbf{k})$	mixed volume of $\mathcal{P}$, w.r.t. type of mixture	99
$\mathfrak{S}_\omega$	collection of mixed cells in S_ω	101
$\widehat{\mathfrak{S}}_\omega$	collection of lifted mixed cells in $\widehat{S}_\omega$	101
$\widehat{\mathcal{F}}$	tuple of k_i-faces of the lower hull of $\widehat{P}_i$	101
$\widehat{f}(\mathbf{x}, t)$	lifted Laurent polynomial	102
$\widehat{F}(\mathbf{x}, t)$	polyhedral homotopy	102
$F_{\mathcal{C}_{\mathbf{v}}}$	initial form system, $F_{\mathcal{C}_{\mathbf{v}}}(\mathbf{x}) = \partial_{\mathbf{v}} \widehat{F}(\mathbf{x}, 1)$	102
$\widehat{F}_{\mathbf{v}}$	the polyhedral homotopy for $\mathbf{v}$	103
$K(A)$	normal cone on A	105
$K(A, B)$	normal cone on A w.r.t. B	105
$\mathcal{N}^j(A)$	normal cone complex of j-faces of $\mathrm{conv}(A)$	106
$\mathcal{N}^j_\vee(A)$	normal cone complex of j-faces of lower hull of $\mathrm{conv}(A)$	106
$\mathcal{N}(A)$	normal fan of A	106
$\mathcal{K} \wedge \mathcal{K}'$	common refinement of two normal cone complexes $\mathcal{K}$ and $\mathcal{K}'$	106
$S^{\mathbf{x}}_\omega$	$S^{\mathbf{x}}_\omega = \{C \in S_\omega \mid \mathbf{x} \in C\}$, cells in S_ω that contain $\mathbf{x}$	116
$\widehat{S^{\mathbf{x}}_\omega}$	$\widehat{S^{\mathbf{x}}_\omega} = \{\widehat{C} \in \widehat{S}_\omega \mid \widehat{\mathbf{x}} \in \widehat{C}\}$, lifted cells in $S^{\mathbf{x}}_\omega$	116
$\triangle^{\mathbf{x}}_\omega$	$\triangle^{\mathbf{x}}_\omega = \{C \in \triangle_\omega \mid \mathbf{x} \in C\}$, cells in $\triangle_\omega$ that contain $\mathbf{x}$	116
$\widehat{\triangle^{\mathbf{x}}_\omega}$	$\widehat{\triangle^{\mathbf{x}}_\omega} = \{\widehat{C} \in \widehat{\triangle}_\omega \mid \widehat{\mathbf{x}} \in \widehat{C}\}$, lifted cells in $\triangle^{\mathbf{x}}_\omega$	116
$K_\vee(\widehat{\mathbf{a}}, \widehat{A})$	normal cone to $\widehat{\mathbf{a}}$ in the lower hull of $\mathrm{conv}(\widehat{A})$	131
$\widehat{S}^{(i)}_\omega$	subdivision of A_i induced by $\omega = (\omega_1, \omega_2, \ldots, \omega_r)$	134
$\widehat{\triangle}^{(i)}_\omega$	triangulation of A_i induced by $\omega = (\omega_1, \omega_2, \ldots, \omega_r)$	137
$\widehat{\mathcal{F}^{\mathbf{x}}_i}$	k_i-faces of the lower hull of $\widehat{P}_i$ that contain $\widehat{\mathbf{x}}$	137
$\widehat{\triangle}^{(i)\mathbf{x}}_\omega$	the cells in a regular triangulation of P_i that contain $\widehat{\mathbf{x}}$	137
$\widehat{\mathfrak{S}^{\mathbf{x}}_\omega}$	the mixed cells in a regular subdivision that contain $\widehat{\mathbf{x}}$	137
Z_2	$Z_2 = \{0, 1\}$	154
(G, V^F, W^F)	symmetry on Laurent polynomial system	154
$(G, V^{\mathcal{A}}, W^{\mathcal{A}})$	symmetry on tuple of supports	154
$\widehat{V^{\mathcal{A}}}(g)$, $\widehat{W^{\mathcal{A}}}(g)$	group representation on lifted tuples of supports	155
$G_{\mathcal{C}}$	isotropy group of the cell $\mathcal{C}$	158
S_n	full permutation group	159
$G_{\mathbf{x}}$	isotropy group of the vector $\mathbf{x}$	160
$\mathrm{Fix}(G, V^F)$	fixed-point space of G	160
F^{red}	system reduced to fixed-point space	160
$\mathcal{O}_{(\mathbf{a}, i)}$	orbit of the point $\mathbf{a} \in A_i$	161
$\widetilde{W}^{\mathcal{A}}(g^{-1})$	permutation matrix corresponding to $V^{\mathcal{A}}(g)$	161
$G_{(\mathbf{a}, i)}$	isotropy subgroup of $\mathbf{a} \in A_i$	163

Chapter 1

Introduction

Polynomial systems occur frequently in various fields of science and engineering, such as, e.g., formula construction, geometric intersection problems, inverse kinematics and the computation of equilibrium states. During the last two decades, homotopy continuation methods have proven to be reliable and efficient numerical algorithms for computing approximations to all isolated solutions of polynomial systems.

Very roughly, the history of homotopy continuation for polynomial systems can be divided into two eras, each spanning about one decade. The first decade was focussed on applying Bézout's theorem for counting the solutions, while during the latter ten years, more sophisticated root counting methods have been developed, which allow to exploit more of the structure of a polynomial system. This chapter is mainly concerned with the description of the results established in the first decade, whereas results of the second decade are treated in the rest of the dissertation.

The development of homotopy continuation methods started around the mid seventies with the work of Drexler [36] and Garcia & Zangwill [51]. In [50], Garcia & Li presented Bézout's theorem as root counting method for polynomial systems. The homotopy method proposed by Chow, Mallet-Paret & Yorke [27] was simplified by Li [81], Morgan [108] and further modified by Brunovský & Meravý [19]. In [182], practical experiences with the homotopy continuation method were reported by Wright. Regularity results were presented by Li & Sauer [83]. The first decade concluded with the appearance of the introductory paper of Li [82], the book of Morgan [104] and the survey of Watson [179], along with publicly available software packages, such as CONSOL [104] and HOMPACK [118, 181].

The purpose of this introduction is to define the problem, to outline the solution method and to provide a motivation for this work. Section 1.1 contains the problem definition and some examples of polynomial systems. The principle of homotopy continuation methods is explained in Section 1.2. In Section 1.3, the basic homotopy method is presented, based on Bézout's theorem. Complexity models are discussed in Section 1.4. In Section 1.5 we sketch other solution methods and related work. This chapter concludes with the outline of the thesis.

1.1 Solving polynomial systems

Algorithms will be developed for the following problem: compute numerically approximations to all complex isolated solutions of systems of polynomial equations.

Definition 1.1.1 A *complex multivariate polynomial* $f(\mathbf{x}) \in \mathbb{C}[\mathbf{x}]$, with unknowns $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, is given in multi-index notation by

$$f(\mathbf{x}) = \sum_{k=0}^{l} c_{\mathbf{d}_k} \mathbf{x}^{\mathbf{d}_k}, \quad \text{with} \quad c_{\mathbf{d}_k} \in \mathbb{C}_0, \quad \mathbf{d}_k \in \mathbb{N}^n \quad \text{and} \quad \mathbf{x}^{\mathbf{d}_k} = x_1^{d_{k1}} x_2^{d_{k2}} \cdots x_n^{d_{kn}}, \tag{1.1}$$

hereby $\mathbb{C}_0 = \mathbb{C} \setminus \{0\}$.

Definition 1.1.2 A *complex polynomial system* $F(\mathbf{x}) = \mathbf{0}$ is defined by a tuple of complex multivariate polynomials, $F = (f_1, f_2, \ldots, f_n)$, $f_k(\mathbf{x}) \in \mathbb{C}[\mathbf{x}]$, $k = 1, 2, \ldots, n$. Its *Jacobian matrix* $J_F(\mathbf{x})$ consists of all first partial derivatives and is given by the matrix

$$J_F(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}. \tag{1.2}$$

The *Jacobian of* F is the determinant $\det(J_F(\mathbf{x}))$ of the Jacobian matrix J_F of F.

Definition 1.1.3 Given a polynomial system F, $\mathbf{x}^*$ is a *solution* of F, if $F(\mathbf{x}^*) = \mathbf{0}$.

We say that $F(\mathbf{x}) = \mathbf{0}$ has an *approximate solution* $\mathbf{x}^*$ if $\mathbf{x}^*$ can be refined up to arbitrary precision by some locally convergent iterative method. In practical computations, approximate solutions are the best that can be obtained. Generally, we refer to a *root* of a polynomial (or a tuple of polynomials) and to a solution of an equation (or a system of equations). The process of estimating the number of solutions is called *root counting*[1].

Definition 1.1.4 A solution $\mathbf{x}^*$ is *regular* if $\det(J_F(\mathbf{x}^*)) \neq 0$, otherwise it is called *singular*. A solution is *isolated* if there exists a nonempty ball around it containing no other solution of the system. Let $\mathbf{x}^*$ be an isolated solution and consider a slight perturbation of the coefficients of F. Then $\mathbf{x}^*$ has *multiplicity* m if in the neighborhood of $\mathbf{x}^*$, the perturbed system has m isolated solutions.

The latter definition of multiplicity is the least complicated one (see [104]), justified by Sard's theorem. This theorem will be presented in Section 1.3. Next we list some very simple problem instances that come from various fields of science and engineering. Except for the first example, they are all taken from [104].

[1]The term *root counting* is due to Canny and Rojas. They presented in [25] the BKK bound as of important practical significance for solving polynomial systems. In Chapter 5 we explain this important root count method.

Example 1.1.5 *Formula construction.* Consider the problem of constructing interpolating quadrature formulae for numerical integration, see [146]. The integrand is a function $f(x)$, defined over an interval $[a, b]$. The problem is to approximate the integral of f over $[a, b]$ by a weighted sum $I(f)$ of function values: $I(f) = \sum w_k f(x_k)$, with w_k the weights and x_k the knots of the quadrature formula. The aim is to develop high accuracy formulae which integrate exactly all polynomials up to a certain degree. With a formula using two knots, every third degree polynomial can be approximated exactly, provided the knots and weights satisfy the conditions given in (1.3).

$$\int_a^b f dx \approx I(f) = w_1 f(x_1) + w_2 f(x_2) \quad \text{with} \quad I(x^k) = m_k = \int_a^b x^k dx, \quad k = 0, 1, 2, 3. \tag{1.3}$$

The knots x_1, x_2 and weights w_1, w_2, satisfying the polynomial system F in (1.4), define a quadrature formula $I(f)$ which integrates every third-degree univariate polynomial exact.

$$F(x_1, x_2, w_1, w_2) = \left\{ \begin{array}{l} I(x^0) = m_0 \\ I(x^1) = m_1 \\ I(x^2) = m_2 \\ I(x^3) = m_3 \end{array} \right. = \left\{ \begin{array}{c} w_1 + w_2 - m_0 = 0 \\ w_1 x_1 + w_2 x_2 - m_1 = 0 \\ w_1 x_1^2 + w_2 x_2^2 - m_2 = 0 \\ w_1 x_1^3 + w_2 x_2^3 - m_3 = 0 \end{array} \right. \tag{1.4}$$

The system F has two solutions, which are invariant under the permutation $(x_1, w_1) \leftrightarrow (x_2, w_2)$. Note that for constructing quadrature formulae, explicitly solving a polynomial system can be avoided. However, for multivariate integration rules, we inevitably have to deal with polynomial systems, see e.g. [31] and [156]. Continuation methods were applied in [157] to construct a family of cubature formulae.

Besides numerical integration rules, another formula construction problem for which the solution of a polynomial system is needed, is the construction of Runge-Kutta methods for numerically solving ordinary differential equations. ■

Example 1.1.6 *Geometric intersection problems.* Consider the intersection of two circles in the plane, see Figure 1.1.
The equations defining the intersection points are given in (1.5).

$$F(x, y) = \left\{ \begin{array}{l} (x-0)^2 + (y-0)^2 = 5^2 \\ (x-8)^2 + (y-0)^2 = 5^2 \end{array} \right. = \left\{ \begin{array}{l} x^2 + y^2 - 25 = 0 \\ x^2 + y^2 - 16x + 39 = 0 \end{array} \right. \tag{1.5}$$

In Figure 1.1, we see that the polynomial system $F(x, y) = \mathbf{0}$ has two real solutions.

Geometric intersection problems are one of the problems to be dealt with when modeling geometric objects. See e.g. [47] for an example of a computer vision application. ■

Example 1.1.7 *Inverse kinematics of robot manipulators.* In Figure 1.2, a planar robot arm with two links is pictured. The lengths of the links are denoted by a_1 and a_2. The problem is to determine the angles θ_1 and θ_2 at the joints of the arm, needed to move the hand of the arm into a given position.
By successive translations and rotations, the origin $(0, 0)$ of the coordinate system is moved to a given position of the "hand" of the robot arm. Denote $c_k = \cos(\theta_k)$ and $s_k = \sin(\theta_k)$, for $k = 1, 2$. In (1.6), the correspondence between the first and second coordinate system is given.

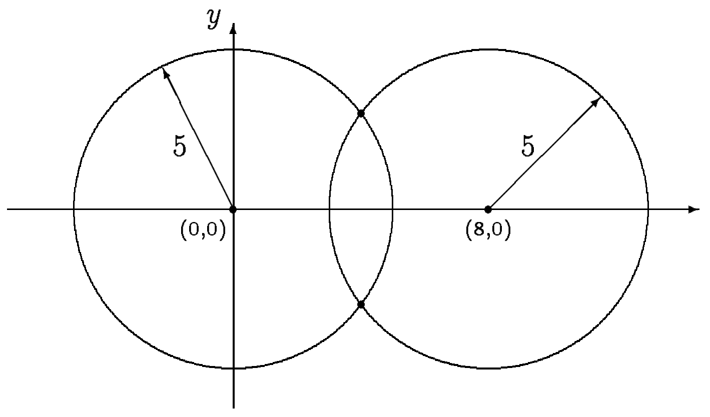


Figure 1.1: Two circles with radius 5 in the plane, centered at $(0,0)$ and $(8,0)$.

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} + \begin{bmatrix} a_1 c_1 \\ a_1 s_1 \end{bmatrix} \tag{1.6}$$

By using a Denavit-Hartenberg matrix, it is possible to express a rotation followed by a translation in one single matrix-vector product. This yields (1.7), as an equivalent formulation to (1.6):

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} c_1 & s_1 & a_1 c_1 \\ -s_1 & c_1 & a_1 s_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = T_1 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}. \tag{1.7}$$

Analogously, the correspondence between the second and third coordinate system can be expressed using T_2 as Denavit-Hartenberg matrix, representing the rotation and translation defined by θ_2 and a_2 respectively. In order to obtain a system with as many equations as unknowns, the equations $s_k^2 + c_k^2 - 1 = 0$, $k = 1, 2$, need to be added.

$$F(c_1, s_1, c_2, s_2) = \left\{ \begin{array}{c} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = T_1 T_2 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ c_1^2 + s_1^2 - 1 = 0 \\ c_2^2 + s_2^2 - 1 = 0 \end{array} \right. = \left\{ \begin{array}{r} a_2 c_1 c_2 + a_2 s_1 s_2 + a_1 c_1 = 0 \\ -a_2 c_2 s_1 + a_2 c_1 s_2 + a_1 s_1 = 0 \\ c_1^2 + s_1^2 - 1 = 0 \\ c_2^2 + s_2^2 - 1 = 0 \end{array} \right. \tag{1.8}$$

The four quadratic equations of the system $F(c_1, s_1, c_2, s_2) = \mathbf{0}$ in (1.8) determine the sines and cosines of the angles θ_1 and θ_2, from which these angles can be derived. For the hand position pictured in Figure 1.2, there are two solutions.

The inverse kinematics of robot manipulators is one of the problems in computational kinematics. A similar problem in this field occurs in the design of four-bar mechanisms. See [131] and the references cited therein for the importance of polynomial system solving in kinematics. ■

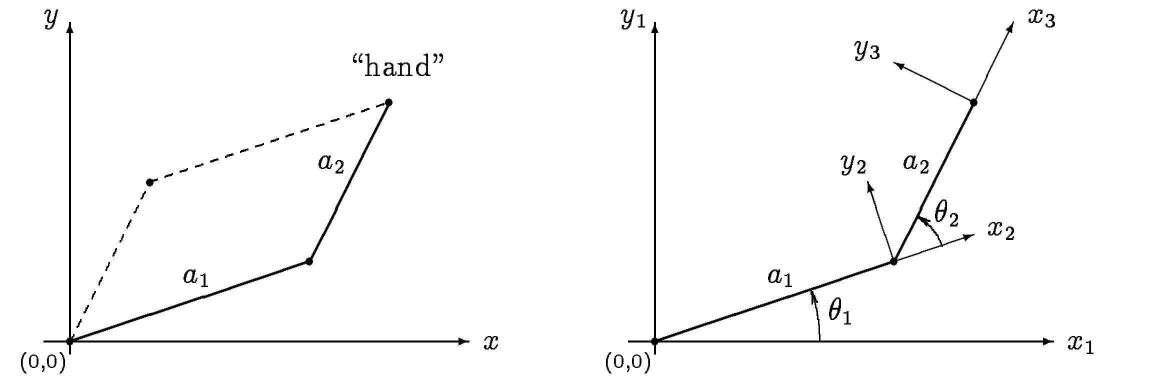


Figure 1.2: To put the hand of the arm (with lengths a_1 and a_2) in the desired position, the angles θ_1 and θ_2 are required. For the given position of the hand, there are two different values for the angles that reach this position. The alternative solution is presented with dashed lines.

Example 1.1.8 *Equilibrium states.* Consider a container with hydrogen (H), oxygen (O) and water (H_2O), heated at a constant temperature. On the left of (1.9) are the equilibrium reaction equation and the conservation laws. The relation between chemical and algebraic variables are given in the middle (1.9). On the right of (1.9) the corresponding polynomial system $F(\mathbf{x}) = \mathbf{0}$ is displayed.

$$\begin{array}{rcl} H_2O & \leftrightarrows & H + O \\ 2[H_2O] + [H] & = & \text{total hydrogen} \\ [H_2O] + [O] & = & \text{total oxygen} \end{array} \quad \begin{array}{l} x_1 = [H] \\ x_2 = [O] \\ x_3 = [H_2O] \end{array} \quad F(\mathbf{x}) = \left\{ \begin{array}{r} 2x_3 + x_1 = T_H \\ x_3 + x_2 = T_O \\ Kx_3 = x_1^2 x_2 \end{array} \right. \tag{1.9}$$

The total amounts of hydrogen and oxygen are given by the constants T_H and T_O respectively. K is the equilibrium constant, depending on the conditions of the reaction, in particular the temperature. The system can be reduced to a single univariate cubic equation.

The range of coefficients that is typical for chemical applications force the use of scaling before solving the system. See [97] for a numerical scaling approach. ■

Each polynomial system coming from an application field can be treated by appropriate ad hoc methods. It is though more tempting to develop a general purpose solver, taking only the polynomial system as input, assuming no particular knowledge from the application.

1.2 Homotopy continuation methods

The standard algorithm to deal with systems of nonlinear equations (see [125]) is Newton's method[2]. It is presented in Algorithm 1.2.1, following an Ada-like syntax. We use $\mathbb{N}_0$ to denote $\mathbb{N} \setminus \{0\}$.

Algorithm 1.2.1 Newton's method for refining an approximation to a solution.

Input:	$F(\mathbf{x}) = \mathbf{0}$, $\mathbf{x}^* \in \mathbb{C}^n$,	*system and approximate solution*
	$\epsilon > 0$, $max\_it \in \mathbb{N}_0$.	*accuracy and max. #iterations*
Output:	$\mathbf{x}^*$, success if $\|\|F(\mathbf{x}^*)\|\| \leq \epsilon$.	*approximate solution if success*

```
k := 0;                                                  initialize counter
while (||F(x*)|| > ε) and (k < max_it) loop
  y ∈ ℂ^n: J_F(x*)y = −F(x*);                            compute correction
  x* := x* + y;                                          correct solution
  k := k + 1;                                            augment counter
end loop;
success := (||F(x*)|| ≤ ε).                              report success or failure
```

Algorithm 1.2.1 only presents a crude version of Newton's method. More sophisticated stopping criteria should be applied, for instance, taking the relative precision into account, i.e., stop when the correction $\mathbf{y}$ becomes small enough. The condition of the Jacobian matrix should also be checked during computation.

The convergence of Newton's method is only *local*, it can only converge when the approximation to the solution lies already sufficiently close to the solution. And even when there is convergence, it might very well be that the method converges to a nonphysical solution. *Global convergence* can be achieved by the use of *continuation* methods. Continuation is based on the deformation principle: close systems have close solution sets. The applicability of Newton's method to a perturbed system using the solutions of the original system as approximations is guaranteed by Sard's theorem (see [67]). Theorem 1.2.2 provides a particular version of this theorem, formulated in the context of polynomial systems over the complex field.

Theorem 1.2.2 (Sard) *Consider a polynomial system $F(\mathbf{x}) = \mathbf{c}$. For almost all choices of vectors $\mathbf{c} \in \mathbb{C}^n$: $\det(J_F(\mathbf{x}^*)) \neq 0$, for all solutions $\mathbf{x}^* \in \mathbb{C}^n$, $F(\mathbf{x}^*) = \mathbf{c}$.*

In [67], a proof for Theorem 1.2.2 is given. More qualitative versions of this theorem for numerical analysis can be found in [183].

Homotopy continuation methods consist of two stages. First there is the embedding of the system to solve in a family of systems, the so-called homotopy. This embedding is done by *homotopy methods*. In the second stage, *continuation methods* are applied to follow numerically the solution paths.

[2] In the light of the historical development, we would better name this method the Newton-Raphson-Simpson method, see [184].

Definition 1.2.3 A *homotopy* $H(\mathbf{x}, t) = \mathbf{0}$ is a family of systems

$$H(\mathbf{x}, t) = c(1-t)^k F^{(0)}(\mathbf{x}) + t^k F^{(1)}(\mathbf{x}) = \mathbf{0}, \quad k \in \mathbb{N}_0,\ c \in \mathbb{C}_0, \quad t \in [0,1], \tag{1.10}$$

parametrized by a *continuation parameter*, denoted by t.

For $k = 1$, the usual *convex-linear* homotopy is obtained. Taking $k > 1$ induces smaller steps at the beginning and at the end of each solution path, see [8]. The homotopy contains the *start system* $F^{(0)}(\mathbf{x}) = H(\mathbf{x}, 0) = \mathbf{0}$, its solutions are called *start solutions*, and the *target system* $F^{(1)}(\mathbf{x}) = H(\mathbf{x}, 1) = \mathbf{0}$, whose solutions are desired.

The homotopy given in (1.10) is artificially constructed, without a physical meaning for t. This type of homotopy is called an *artificial-parameter homotopy*. Many polynomial systems arising in practical applications have parameters. Consider for example the system F in (1.8), representing the inverse kinematics of a robot arm. We could study the behavior of the solutions as the position of the hand or the lengths of the links vary. The polynomial system F then becomes a *natural-parameter homotopy*, with the position of the hand or the lengths of the links as continuation parameters. Instead of solution paths, this natural-parameter homotopy has four-dimensional solution sets, which can nevertheless be approximated numerically by applying continuation methods. So, in computing solutions of a polynomial system by homotopy continuation, both artificial-parameter and natural-parameter homotopies are used, each at different stages of the overall solving procedure.

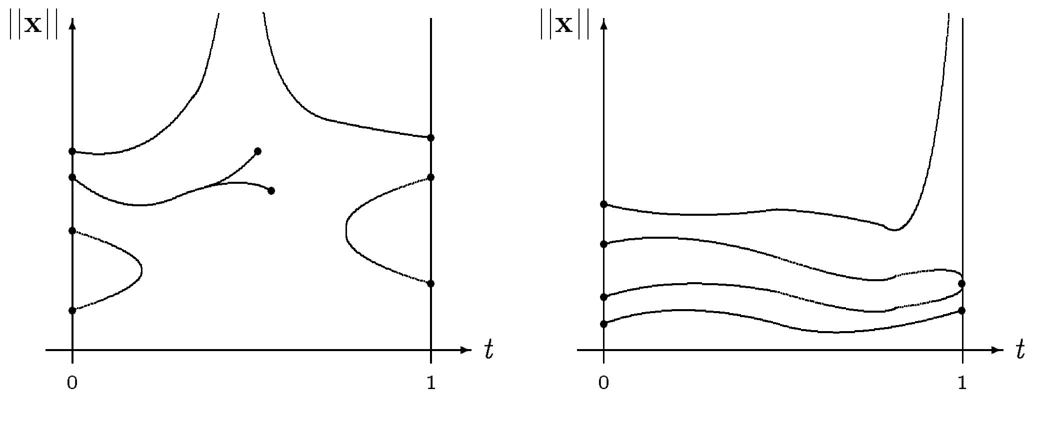


Figure 1.3: On the left a bad behavior of the solution paths is pictured: paths that turn back, bifurcate, stop or diverge to infinity for some t, $|t| < 1$. A typical behavior is drawn on the right: paths can only diverge at the end, when $t \approx 1$.

Definition 1.2.4 A *good homotopy* $H(\mathbf{x}, t)$ should possess the following properties (criteria taken from [82]):

1. (triviality) The solutions of $H(\mathbf{x}, 0) = \mathbf{0}$ are known.

2. (smoothness) The solution set of $H(\mathbf{x}, t) = \mathbf{0}$ for all t: $0 \leq |t| \leq 1$, consists of a finite number of smooth paths, each parameterized by t.

3. (accessibility) Every isolated solution of $H(\mathbf{x}, 1) = \mathbf{0}$ is reached by some path origination at a solution of $H(\mathbf{x}, t) = \mathbf{0}$ for $t = 0$. If m is the multiplicity of an isolated solution $\mathbf{x}^*$, then there are exactly m paths converging to $\mathbf{x}^*$.

A homotopy that does not satisfy all these properties is a *bad homotopy*.

In Figure 1.3, the behavior of the solution paths is pictured, with on the left a bad and on the right a good homotopy.

Continuation or path following methods are standard globally convergent solution methods for nonlinear systems, see [2, 3]. For polynomial systems there are some important differences. In general, we are only interested in the solutions at the end of the path. Singularities along the path can be avoided, eventually by allowing complex values for t. Hereby it is essential that path following is done over the complex numbers. See [104] for continuation methods for polynomial systems.

Path following methods are based on *predictor-corrector* methods. The *predictor* delivers at each step of the method a new value of the continuation parameter and predicts an approximate solution of the corresponding new system in the homotopy. Then, the predicted approximate solution is corrected by applying the *corrector*, e.g. by Newton's method. With a good homotopy, the solution paths never turn back as t increases. Therefore, the continuation parameter can remain fixed while correcting the predicted solution. This leads to so-called *increment-and-fix* path following methods. In practice, determining the step length during the prediction stage is done by a hit-or-miss method, which can be implemented by means of an *adaptive step size control*, as done in Algorithm 1.2.5.

Following all paths defined by a homotopy can be done *sequentially*, one path at a time, or in *parallel*, with for each solution path the same sequence of values of the continuation parameter. The sequential path following method has the advantage that it can be very well adapted to run on multi-processor machines, because there is a low overhead of communication. See [4] for granularity issues. By tracing the paths in parallel, it is possible to detect path jumping already in an early stage of the continuation. However, one difficult path can slow down the continuation of all other ones.

Algorithm 1.2.5 Following one solution path by an increment-and-fix predictor-corrector method with an adaptive step size control strategy.

Input:	$H(\mathbf{x}, t)$, $\mathbf{x}^* \in \mathbb{C}^n$: $H(\mathbf{x}^*, 0) = \mathbf{0}$,	*homotopy and start solution*
	$\epsilon > 0$, *max_it*, *max_steps*.	*accuracy and upper bounds*
Output:	$\mathbf{x}^*$, success if $\|\|H(\mathbf{x}^*, 1)\|\| \leq \epsilon$.	*approximate solution if success*

$t := 0$; $k := 0$; *initialization*
$h := max\_step\_size$; *step length*
$old\_t := t$; $old\_\mathbf{x}^* := \mathbf{x}^*$ *back up values for t and* $\mathbf{x}^*$
$previous\_\mathbf{x}^* := \mathbf{x}^*$; *previous approximate solution*
stop := false; *combines stopping criteria*

```
while t < 1 and not stop loop
   t := min(1, t + h);                                   secant predictor for t
   x* := x* + h(x* − previous_x*);                       secant predictor for x*
   Newton(H(x,t), x*, ε, max_it, success);               correct with Algorithm 1.2.1
   if success                                            step size control
    then h := min(Expand(h), max_step_size);             enlarge step length
         previous_x* := old_x*;                          go further along path
         old_t := t;  old_x* := x*;                      new back up values
    else h := Shrink(h);                                 reduce step length
         t := old_t;  x* := old_x*;                      step back and try again
   end if;
   k := k + 1;                                           augment counter
   stop := (h < min_step_size) or (k > max_steps);       stopping criteria
end loop;
success := (||H(x*, 1)|| ≤ ε).                           report success or failure
```

Algorithm 1.2.5 contains three key ingredients in its loop: the predictor, the corrector and the step size control. The step size h is controlled by the functions *Shrink* and *Expand* which respectively reduce and enlarge h, depending on the outcome of the corrector. It should also be clear that there are far more parameters to tune, than those explicitly mentioned in the input specification.

For solving polynomial systems by homotopy continuation methods, the construction of the homotopy is crucial for the performance of the solver. Therefore, in this work, emphasis is placed on the development of homotopy methods. For further information on continuation methods, see [2, 3] or [104].

1.3 The theorem of Bézout

Homotopy methods for polynomial systems consist in constructing a start system which allows to compute all isolated solutions of a given polynomial system. In order to be able to compute all isolated solutions, we first need to know an upper bound for the number of isolated solutions. To get a uniform root count, we need to compactify $\mathbb{C}^n$, by moving to projective space.

Definition 1.3.1 Consider the equivalence relation $\sim$ defined by

$$(z_0, z_1, \ldots, z_n) \sim (z'_0, z'_1, \ldots, z'_n) \quad \Leftrightarrow \quad \exists \lambda \in \mathbb{C}_0 : \; z_k = \lambda z'_k, \; k = 1, 2, \ldots, n. \tag{1.11}$$

By restricting to the equivalence classes of $\sim$, the *n-dimensional complex projective space* is defined by $\mathbb{P}^n = (\mathbb{C}^{n+1} \setminus \{\mathbf{0}\}) / \sim$.

Proposition 1.3.2 *$\mathbb{P}^n \setminus \{z_0 = 0\}$ is isomorphic to $\mathbb{C}^n$ by the map ϕ:*

$$\phi : \; \mathbb{P}^n \setminus \{z_0 = 0\} \to \mathbb{C}^n : \; (z_0, z_1, z_2, \ldots, z_n) \mapsto \phi(\mathbf{z}) = \left(\frac{z_1}{z_0}, \frac{z_2}{z_0}, \ldots, \frac{z_n}{z_0} \right). \tag{1.12}$$

Proposition 1.3.2 (proven in [32]) allows to view $\mathbb{P}^n$ as $\mathbb{C}^n$ extended with the hyperplane at infinity, represented by $z_0 = 0$. There are many alternative compactifications of $\mathbb{C}^n$, by taking different homogeneous linear forms to represent the hyperplane at infinity, see [104].

Definition 1.3.3 *The degree d of a multivariate polynomial f, $d = \deg(f)$, is de-* fined as the largest degree of the monomials with nonzero coefficients of f, i.e., $d = \max\{\sum_{j=1}^n d_{kj} \mid c_{\mathbf{d}_k} \neq 0\}$, with f denoted by $\sum_{k=0}^l c_{\mathbf{d}_k} \mathbf{x}^{\mathbf{d}_k}$. We define $\deg(0) = 0$.

Definition 1.3.4 Let $f(\mathbf{x}) \in \mathbb{C}[\mathbf{x}]$. The *projective transformation of* f is defined by the polynomial $z_0^d f(\phi(\mathbf{x}))$, where $d = \deg(f)$.

The importance of the projective transformation is that it allows to treat *solutions at infinity* just like any other *finite solution*. When moving from $\mathbb{C}^n$ to $\mathbb{P}^n$, the distinction between finite solutions and solutions at infinity ceases to exist, which allows to provide a uniform root count.

Definition 1.3.5 Given a polynomial system $F = (f_1, f_2, \ldots, f_n)$, *the total degree D of* F is the product of the degrees of f_k: $D = \prod_{k=1}^n \deg(f_k)$.

Theorem 1.3.6 (Bézout) *If the number of solutions of a polynomial system $F(\mathbf{z}) = \mathbf{0}$ is finite, then there are exactly D isolated solutions in $\mathbb{P}^n$, counted with multiplicities.*

See [120] for a proof of Theorem 1.3.6. Following [82], we can view Bézout's theorem as a generalization of two fundamental theorems in algebra, for univariate polynomial equations ($n = 1$) and linear systems (all $d_k = 1$, $k = 1, 2, \ldots, n$).

Definition 1.3.7 Let $\mathbf{d} = (d_1, d_2, \ldots, d_n)$ represent a vector of degrees. *A start system $F^{(0)}(\mathbf{x}) = \mathbf{0}$ based on the degrees* $\mathbf{d}$ is given by the following tuple $F^{(0)} = (f_1^{(0)}, f_2^{(0)}, \ldots, f_n^{(0)})$, $f_k^{(0)} = x_k^{d_k} - c_k$, $c_k \in \mathbb{C}_0^n$, $k = 1, 2, \ldots, n$.

Obviously, the start system based on the degrees $\mathbf{d}$, has as many isolated finite solutions as its total degree D. That is why it is more convenient to refer to the system $F^{(0)}$ in Definition 1.3.7 as a *start system based on the total degree.*

Theorem 1.3.8 *Consider a polynomial system $F(\mathbf{x}) = \mathbf{0}$. Embedding F as target system $F^{(1)}$ in the homotopy given by* (1.10)*, together with a start system $F^{(0)}$ based on the total degree of F, yields a good homotopy for computing all isolated solutions of $F(\mathbf{x}) = \mathbf{0}$.*

There are two different ways to prove Theorem 1.3.8. In [36], tools from algebraic geometry are used. A proof based on arguments from differential topology has been given in [51]. See also [104] for a proof based on differential topology.

Example 1.3.9 Consider again the examples presented in Section 1.1. Table 1.1 lists for the examples their total degrees and $N(F)$, the number of isolated solutions in $\mathbb{C}^n$, for nondegenerate instances of the problems, that is, for generic choices of the parameters. Table 1.1 shows that, except for the last example, there exists a gap between the total degree and the number of finite solutions. ■

system	description	D	$N(F)$
(1.4)	quadrature formula	24	2
(1.5)	geometric intersection	4	2
(1.8)	inverse kinematics	16	2
(1.9)	equilibrium states	3	3

Table 1.1: The total degree versus the number of finite solutions.

Example 1.3.9 illustrates the motivation of this thesis. For almost all applications, the total degree is a too large upper bound for the number of finite solutions. In that case, the polynomial systems are called *deficient* (see [84]). Because solutions at infinity are never interesting, we want to avoid computing them. We define the *performance of the homotopy* as the quotient of $N(F)$ and the number of paths that have to be followed to compute all solutions of $F(\mathbf{x}) = \mathbf{0}$. In the ideal case, each path leads to an isolated solution of $F(\mathbf{x}) = \mathbf{0}$ and the performance equals one.

1.4 Complexity models

The classical notion of arithmetic complexity will be used, that is the number of arithmetical operations needed in the optimal algorithm for solving a problem models the *complexity* of that problem. Hereby we assume a uniform length of the real numbers, as recently proposed in a new model of computation, see [16].

The following intuitive reasoning might shed a light on the hardness of the problem. For a treatment on complexity classes, we refer to [52]. Not surprisingly, evaluating a multivariate polynomial can be done in *polynomial time*, i.e., in time proportional to a polynomial function in its size. Computing one solution is *NP-hard*, because the following non-deterministic algorithm can be applied: guess a root by means of an oracle and verify whether it is an approximate solution. This verification runs in polynomial time. Computing *all* solutions is harder, this problem is *#P-hard*, because there exists no algorithm that runs in polynomial time to verify whether a guessed number of solutions is the right estimate. Hence the problem is intractable for growing dimension and increasing degrees. Consequently, the computations are restricted to a fixed dimension and it makes sense to measure the complexity of a problem in terms of the output size, that is the number of solutions.

On average, the cost of the homotopy continuation method depends linearly on the number of solutions, for a fixed dimension. This means for instance that it takes twice as long to solve a system with twice as many solutions. In [141], Shub and Smale proved that, on average, using the homotopy continuation method based on Bézout's theorem, it is possible to compute one solution of a polynomial system in polynomial time. Their proof is an application of Smale's α-theory, which renders universal constants on the domains of convergence of Newton's method, to continuation methods. This allows to estimate the number of steps needed to trace a solution path, in terms of the length of the path. On average means that the considered polynomial system is well-conditioned and has as many finite solutions as given by its total degree.

Bézout's theorem only uses the degrees to count the roots. The homotopy presented in the Section 1.3 links two extreme cases in one family: the most sparse with the most dense polynomial system. Applying the theorem of Bézout implies that *degree complexity* is used to model the difficulty of solving a polynomial system. This model works well when dealing with *dense polynomial systems*, i.e., when all monomials (or almost all) have a nonzero coefficient. However, when dealing with *sparse polynomial systems*, i.e., if only a few monomials have nonzero coefficients, then it is more appropriate to work with the notion of *geometric complexity*. One of the aims of this thesis is to provide practical evidence for the fact that the sparser a polynomial system is, the easier it can be solved.

1.5 Other methods and related work

In this section we briefly sketch three alternative methods for solving polynomial systems: interval methods, resultants and Gröbner bases. For each method we refer to a standard textbook and try to cite some recent research directions. At the end of this section we mention related work that was most influential to the development of the homotopy methods presented in this dissertation.

Interval methods (see [102]) can be used to isolate all roots inside a bounded box. By generalized bisection the box is subdivided after which inclusion-exclusion tests are applied to eliminate those boxes that contain no root. The algorithm terminates with a set of boxes that each isolate one root. We refer to [74], [75], [173] and [172] for a description of methods and publicly available software packages. These methods naturally restrict to real roots and can be applied as well to general systems of equations. The computation of noninteresting solutions outside a bounded domain can be avoided. When also the complex roots are desired, the amount of arithmetical work doubles. The main drawback is that the number of boxes that need to be investigated increases exponentially as the dimension increases. See [34], [132] and [155] for recent developments.

Resultants (see [32]) play an important role in elimination theory. They are based on conditions on the coefficients of an overconstrained system to determine whether there exists a solution. This concept enables to solve a polynomial system by eliminating the unknowns. The main drawback is that they introduce spurious solutions. For large solution sets, the bad condition of the problem one has to deal with after elimination might be insurmountable. In spite of all these disadvantages, resultants have been applied successfully in providing a real time algorithm for solving an inverse kinematics problem, see [96]. An interesting recent development concerns sparse resultants, see [43].

The Buchberger algorithm (see [32]) for constructing a Gröbner basis for the ideal generated by the polynomials in the system can be viewed as a generalization of three basic algorithms: Gaussian elimination, resultants and Euclid's algorithm for the construction of the greatest common divisor. Gröbner bases have many other applications than solving polynomial systems. Implementations of Buchberger's algorithm are widely available in various computer algebra systems. Gröbner bases are not restricted to zero-dimensional ideals. Their main disadvantage is the high complexity of intermediate descriptions of the ideal, while constructing a Gröbner basis. The size of a Gröbner basis can be considerably large compared to the set of isolated solutions. Recent practical progress in constructing a Gröbner basis for zero-dimensional ideals has been described in [48]. We refer to [150] for the interesting connection with methods of discrete and combinatorial geometry.

Finally, we list influential related work on homotopy methods. Multi-homogeneous homotopies were introduced by Morgan and Sommese in [107]. They described in [111] coefficient-parameter polynomial continuation. In [175], Wampler provided an efficient algorithm for computing Bézout numbers. Li, Sauer and Yorke introduced in [86] random product homotopies and in [87] they developed the cheater's homotopy. In [90], Li and Wang proposed conditions on multi-homogeneous homotopies. With their paper [25], Canny and Rojas drew our attention to Bernshtein's theorem [9]. In [72], Huber and Sturmfels outlined the fundamental principles of polyhedral homotopy methods. The elaboration of pruning methods to compute the mixed volume more efficiently was greatly stimulated by the practical results obtained by Emiris and Canny, see [44].

1.6 Thesis overview

The theme is to exploit the structure of a polynomial system to derive better root counts and to construct more performant homotopies. Figure 1.4 depicts the structure of the thesis. There are six chapters sandwiched between the introduction and the chapter that contains the description of our software.

We have developed three different homotopy methods to solve deficient polynomial systems more efficiently. In Figure 1.4, these three methods can be seen as three columns in the sandwich. The first method, called reduction, is treated in the second chapter and consists in applying methods of symbolic elimination. Multi-homogenization and the generalization to product homotopies, respectively described in the third and fourth chapter, provide generalizations of Bézout's theorem. The development of the third class of homotopy methods starts with the fifth chapter. Deficiency is often due to the sparse structure of the polynomial system and polyhedral homotopy methods yield more performant homotopies for solving sparse polynomial systems. The sixth chapter contains the description of an algorithm to compute the volume of a polytope. Although the development of this algorithm was motivated by the need for a more stable polyhedral continuation, the algorithm is of independent interest. The seventh chapter deals with the application of polyhedral homotopy methods to symmetric polynomial systems.

Much of the work that is presented in this dissertation has been published in the literature, though the material treated in those papers has been updated and considerably expanded.

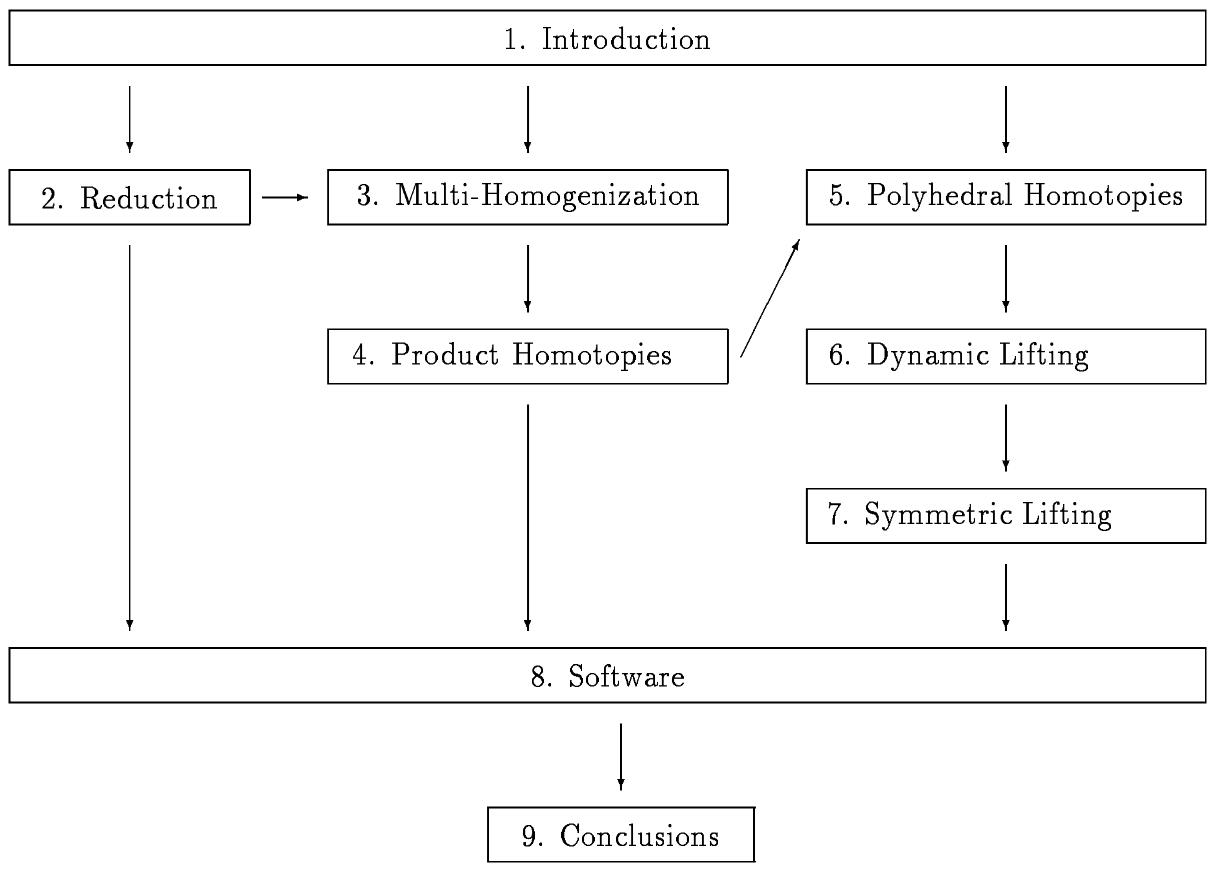


Figure 1.4: Plan of the thesis. The arrows suggest possible orders of reading.

Chapter 2

Reduction

Deficient polynomial systems, which seem to be typical for most applications, have fewer finite solutions than predicted by the total degree. If the start system based on the total degree is used in a homotopy to solve a deficient system, there are solution curves diverging to infinity, representing wasted computations. While the computational work is more or less proportional to the number of solution paths, this number can be very high in comparison to the actual number of finite solutions.

One single problem can have a lot of different algebraic formulations. To construct an optimal formulation requires both the expertise of the application field and a good knowledge of the solution method. This is beyond the scope of a general-purpose solver. Though it is possible to develop an automatic *rewriting method*. Such a method transforms a deficient system into a system with the same finite complex solutions, but with a lower total degree. The resulting system can be solved faster. This justifies to give *reduction* as name of this reformulation method.

In [104, Chapter 7], Morgan developed linear reduction as an automatic reformulation technique. He described in [109] the relationship between polynomial continuation and symbolic reduction methods. We propose to extend the idea of linear reduction to nonlinear reduction by using a key concept of symbolic computation methods (see [32]), i.e., the Subtraction-polynomial. By replacing the original polynomials by Subtraction-polynomials with a lower degree, the reduced system can be solved by a continuation method. Applying this combined solving procedure, the computation of solutions at infinity can in many cases be avoided. We have published these ideas in [161].

The outline of this chapter is as follows. In the following section we recall the concept of linear reduction. The basic idea of our method will be introduced in Section 2.2. Section 2.3 describes several replacing criteria. The nonlinear reduction method is elaborated in detail in Section 2.4. Until then, everything is described as if intended to use symbolic computations, whereas in Section 2.5, the numerical properties of the reduction algorithm will be investigated. Applications of the reduction algorithm are presented in Section 2.6. The last section contains our conclusions and lists some future research directions.

2.1 Linear reduction

In [104, Chapter 7], *linear reduction* has been presented. The coefficients of the system $F(\mathbf{x}) = \mathbf{0}$ are put in a matrix M_F, the so-called *coefficient matrix*. By row-reduction we bring M_F to a row echelon form. The routine GEL in [104] tries to make the coefficient matrix M_F as sparse as possible, but for reducing d, row-reduction is enough.

Example 2.1.1 Consider again the system $F(\mathbf{x}) = \mathbf{0}$ which arose as the intersection of two circles in the plane, presented in Example 1.1.6. The linear reduction on $F(\mathbf{x}) = \mathbf{0}$

$$F(\mathbf{x}) = \begin{cases} x_1^2 + x_2^2 \quad\quad\quad - 25 = 0 \\ x_1^2 + x_2^2 - 16x_1 + 39 = 0 \end{cases} \longrightarrow \quad F'(\mathbf{x}) = \begin{cases} x_1^2 + x_2^2 \quad\quad - 25 = 0 \\ \quad\quad\quad -16x_1 + 64 = 0 \end{cases} \tag{2.1}$$

is implemented by making an appropriate linear combination of the rows of the coefficient matrix M_F:

$$M_F = \begin{bmatrix} 1 & 1 & 0 & -25 \\ 1 & 1 & -16 & 39 \end{bmatrix} \quad \longrightarrow \quad M_{F'} = \begin{bmatrix} 1 & 1 & 0 & -25 \\ 0 & 0 & -16 & 64 \end{bmatrix}. \tag{2.2}$$

The reduced system $F'(\mathbf{x}) = \mathbf{0}$ has a total degree of two, instead of four. This saves at least half of the computational work during the continuation process. Note that the total degree of $F'(\mathbf{x}) = \mathbf{0}$ is now an exact root count for this problem. ∎

If the reduced system contains a trivial equation like '$0 = 0$', then it is probable that there are an infinite number of solutions. Similarly, if an equation like '$1 = 0$' remains after reduction, then there are no finite solutions to compute. Row-reduction provides a simple and powerful reduction algorithm, yielding a significant improvement of the performance of the homotopy, achieved at a little cost.

2.2 Nonlinear reduction

By applying S-polynomials, we can eliminate those terms with the highest degree and consequently reduce the degree of a polynomial. For this purpose, the graded (or graduated) lexicographic ordering (see [32]) is used to order the monomials.

Definition 2.2.1 The *lexicographic ordering* $<_L$ is defined as follows:

$$\mathbf{x}^{\mathbf{d}_1} <_L \mathbf{x}^{\mathbf{d}_2} \quad \Leftrightarrow \quad \exists j : \forall i < j : d_{1i} = d_{2i} \ \text{ and } \ d_{1j} < d_{2j}. \tag{2.3}$$

Definition 2.2.2 The *graded lexicographic ordering* $<_{GL}$ is defined as follows:

$$\mathbf{x}^{\mathbf{d}_1} <_{GL} \mathbf{x}^{\mathbf{d}_2} \quad \Leftrightarrow \quad \sum_{i=1}^n d_{1i} < \sum_{i=1}^n d_{2i} \ \text{ or } \ \left(\sum_{i=1}^n d_{1i} = \sum_{i=1}^n d_{2i} \ \text{ and } \ \mathbf{x}^{\mathbf{d}_1} <_L \mathbf{x}^{\mathbf{d}_2} \right). \tag{2.4}$$

In the sequel, we assume the monomials in a polynomial to be ordered in descending order according to the graded lexicographic ordering. This allows us to establish Notation 2.2.3.

Notation 2.2.3 $Lt(f)$ denotes the leading term of the polynomial f. The coefficient of $Lt(f)$ is denoted by $Lc(f)$. The least common multiple of the polynomials f_1 and f_2 is denoted by $lcm(f_1, f_2)$.

A key tool in *rewriting rules* used in elimination (see [21], [32]) is defined next.

Definition 2.2.4 The *Subtraction-polynomial* or *S-polynomial* $\mathcal{S}(f_1, f_2)$ of two polynomials f_1 and f_2 is given by

$$\mathcal{S}(f_1, f_2) = Lc(f_2)\frac{lcm(Lt(f_1), Lt(f_2))}{Lt(f_1)}f_1 - Lc(f_1)\frac{lcm(Lt(f_1), Lt(f_2))}{Lt(f_2)}f_2. \tag{2.5}$$

The following example introduces *nonlinear reduction.* Note that the reduction algorithm never adds a polynomial to the system, but always tries to replace a polynomial by an S-polynomial.

Example 2.2.5 Consider the system $F(\mathbf{x}) = \mathbf{0}$, presented in [2]. The total degree of this system equals nine, but there are only two finite solutions. The following reduction of $F(\mathbf{x}) = \mathbf{0}$

$$F(\mathbf{x}) = \begin{cases} x_1^3 - x_1 = 0 \\ x_1^2 x_2 + 1 = 0 \end{cases} \longrightarrow \quad F'(\mathbf{x}) = \begin{cases} -x_1x_2 - x_1 = 0 \\ x_1^2 x_2 + 1 = 0 \end{cases} \tag{2.6}$$

is obtained by replacing f_1 by the S-polynomial $\mathcal{S}(f_1, f_2)$, computed as follows:

$$\mathcal{S}(f_1, f_2) = x_2 f_1 - x_1 f_2 = -x_1 x_2 - x_1. \tag{2.7}$$

The total degree of the reduced system $F'(\mathbf{x}) = \mathbf{0}$ equals six. ■

In Section 2.4.2, this system will be reduced further to a system with a total degree of two, which is exactly the number of finite solutions.

2.3 The replacing criterion

One may wonder whether a polynomial can be replaced so easily by an S-polynomial. Indeed there must be some conditions, as illustrated by the following example.

Example 2.3.1 (Example 2.2.5 continued.) Consider the system F, displayed in (2.6). When f_2 is replaced by $\mathcal{S}(f_1, f_2)$, we obtain the following system

$$F'(\mathbf{x}) = \begin{cases} x_1^3 - x_1 = 0 \\ -x_1x_2 - x_1 = 0 \end{cases} \tag{2.8}$$

that has an infinite number of solutions: $\mathbf{x} = (0, \lambda),\ \lambda \in \mathbb{C}$. ■

A condition under which a polynomial can be replaced by an S-polynomial without changing the finite solutions is called a *replacement criterion.* Without loss of generality, we will formulate criteria under which f_1 may be replaced by an S-polynomial and denote the reduced system by $F'(\mathbf{x}) = \mathbf{0}$.

2.3.1 A theoretical criterion

Definition 2.3.2 Two systems of polynomial equations $F(\mathbf{x}) = \mathbf{0}$ and $F'(\mathbf{x}) = \mathbf{0}$ are *equivalent* if and only if they have the same finite solutions with the same multiplicities.

The question is when the reduced system and the original system are equivalent. This problem can be restated into an algebraic problem using ideals.

Definition 2.3.3 Let $\{ f_1, f_2, \ldots, f_r \}$ be a set of multivariate polynomials $f_i \in \mathbb{C}[\mathbf{x}]$, then

$$< f_1, f_2, \ldots, f_r > = \{ \sum_{i=1}^{r} f_i' f_i \mid f_i' \in \mathbb{C}[\mathbf{x}] \} \tag{2.9}$$

is defined as *the ideal generated by the polynomials* $f_1, f_2, \ldots, f_r$.

Theorem 2.3.4 (see [32]) states that any two systems generating the same ideal, have the same sets of solutions.

Theorem 2.3.4 $< F > = < F' > \Leftrightarrow F$ *and* F' *are equivalent.*

Consequently, Corollary 2.3.5 provides a theoretical criterion.

Corollary 2.3.5 *If* $f_1 \in < \mathcal{S}(f_1, f_2), f_2 >$, *then* F *and* F' *are equivalent.*

Example 2.3.6 (Example 2.2.5 continued.) Returning to the reduction in (2.6) of the preceding section, note that $f_1 \in < \mathcal{S}(f_1, f_2), f_2 >$ because $x_1^3 - x_1 = -x_1^2(-x_1x_2 - x_1) - x_1(x_1^2x_2 + 1)$. This justifies the replacement of f_1, whereas the system F' displayed in (2.8), see Example 2.3.1, obtained by replacing f_2 is not equivalent to F because $f_2 \notin < f_1, \mathcal{S}(f_1, f_2) >$. ■

2.3.2 An efficient criterion

Corollary 2.3.5 provides a reliable criterion, but how can it be verified in an efficient manner? The following proposition is an application of a criterion presented in [54]:

Proposition 2.3.7 *If* $lcm(Lt(f_1), Lt(f_2)) = Lt(f_1)$, *then* $< F > = < F' >$.

Proof. If $lcm(Lt(f_1), Lt(f_2)) = Lt(f_1)$, then

$$\mathcal{S}(f_1, f_2) = Lc(f_2)f_1 - Lc(f_1)\frac{Lt(f_1)}{Lt(f_2)}f_2. \tag{2.10}$$

Then $f_1 \in < \mathcal{S}(f_1, f_2), f_2 >$ and Corollary 2.3.5 can be applied. □

This criterion is called the *lcm-criterion* in [32]. It is simple to implement and efficient to evaluate. Unfortunately, it is too restrictive: there are more replacements allowed than indicated, as illustrated in the following example.

Example 2.3.8 (Example 2.2.5 continued.) Consider again the reduction displayed in (2.6). In this case,

$$f_1 \in < \mathcal{S}(f_1, f_2), f_2 > \quad \text{and} \quad Lt(f_1) \neq lcm(Lt(f_1), Lt(f_2)). \tag{2.11}$$

So the replacement is theoretically allowed, but prohibited by the criterion in Proposition 2.3.7. ■

This example motivates the search for a less restrictive criterion.

2.3.3 Reduction with R-polynomials

Definition 2.3.9 The *Residue-polynomial* or *R-polynomial* $\mathcal{R}(f_1, f_2)$ of the polynomials f_1 and f_2 is defined by

$$\mathcal{R}(f_1, f_2) = f_1 - Lc(f_1)\frac{Lt(f_1)}{Rt(f_2)}f_2, \tag{2.12}$$

where $Rt(f_2)$ is *the reducing term of* f_2. $Rt(f_2)$ is defined as the largest term in the polynomial f_2 so that $lcm(Lt(f_1), Rt(f_2)) = Lt(f_1)$. If $Rt(f_2) = 0$, then $\mathcal{R}(f_1, f_2) = f_1$.

Observe that $\mathcal{R}(f_1, f_2)$ eliminates $Lt(f_1)$, whenever $Rt(f_2) \neq 0$.

Example 2.3.10 (Example 2.2.5 continued.) Consider the polynomials f_1 and f_2 of the system F, displayed in (2.6). The S-polynomial $\mathcal{S}(f_1, f_2) = -x_1x_2 - x_1$, as calculated in (2.7). The reducing term $Rt(\mathcal{S}(f_1, f_2)) = -x_1 \neq 0$, so by reducing f_1 with $\mathcal{S}(f_1, f_2)$,

$$\mathcal{R}(f_1, \mathcal{S}(f_1, f_2)) = x_1^3 - x_1 + x_1^2(-x_1x_2 - x_1) = -x_1 - x_1^3x_2 \tag{2.13}$$

we can eliminate the leading term x_1^3 of f_1. ■

Theorem 2.3.11 proposes a replacement criterion by subsequently reducing f_1 w.r.t. f_2 and $\mathcal{S}(f_1, f_2)$, by the application of R-polynomials.

Theorem 2.3.11 *If* $\mathcal{R}(\mathcal{R}(f_1, f_2), \mathcal{S}(f_1, f_2)) = 0$ *or* $\mathcal{R}(\mathcal{R}(f_1, \mathcal{S}(f_1, f_2)), f_2) = 0$, *then* $f_1 \in <\mathcal{S}(f_1, f_2), f_2>$.

Proof. If $\mathcal{R}(\mathcal{R}(f_1, f_2), \mathcal{S}(f_1, f_2)) = \mathcal{R}(f_1, f_2) - c_{\mathbf{d}}\mathbf{x}^{\mathbf{d}} = 0$, then $f_1 = c_{\mathbf{d}_1}\mathbf{x}^{\mathbf{d}_1}f_2 + c_{\mathbf{d}_2}\mathbf{x}^{\mathbf{d}_2}\mathcal{S}(f_1, f_2)$, and thus $f_1 \in <\mathcal{S}(f_1, f_2), f_2>$. Analogously, this can be proved if the second part of the criterion holds. □

Note that the reverse of Theorem 2.3.11 is not always true.

Theorem 2.3.12 *If* $lcm(Lt(f_1), Lt(f_2)) = Lt(f_1)$, *then* $\mathcal{R}(\mathcal{R}(f_1, f_2), \mathcal{S}(f_1, f_2)) = 0$.

Proof. If the efficient criterion of Proposition 2.3.7 holds, then

$$\mathcal{R}(\mathcal{R}(f_1, f_2), \mathcal{S}(f_1, f_2)) = \left(f_1 - \frac{Lc(f_1)}{Lc(f_2)}\frac{Lt(f_1)}{Lt(f_2)}f_2\right) - \frac{1}{Lc(f_2)}\mathcal{S}(f_1, f_2) = 0, \tag{2.14}$$

and the proof is complete. □

Theorem 2.3.12 tells us that the first part of the criterion of Theorem 2.3.11 is a generalization of the efficient criterion of Proposition 2.3.7. The second part of the criterion of Theorem 2.3.11 allows even more possibilities for reduction, as illustrated by the following example.

Example 2.3.13 (Example 2.3.10 continued.) For instance, for the system of Example 2.2.5, we have

$$\mathcal{R}(\mathcal{R}(f_1, \mathcal{S}(f_1, f_2)), f_2) = \mathcal{R}(f_1 + x_1^2\mathcal{S}(f_1, f_2), f_2) = (f_1 + x_1^2\mathcal{S}(f_1, f_2)) + x_1f_2 = 0, \tag{2.15}$$

which proves that $f_1 \in <f_2, \mathcal{S}(f_1, f_2)>$. ■

Theorem 2.3.11 provides an efficient algorithm to verify the condition of Corollary 2.3.5. Evaluating this criterion requires more work than evaluating the efficient criterion of Proposition 2.3.7, but the difference in computational time is negligible in comparison to the calculation of a solution curve diverging to infinity. Nevertheless, the efficient criterion remains useful as a kind of preprocessing to the criterion of Theorem 2.3.11. Only if the efficient criterion fails, we have to evaluate the criterion based on R-polynomials.

The following theorem deals with the case when $\deg(\mathcal{S}(f_1, f_2)) = 0$.

Theorem 2.3.14 *Consider the tuple $F = (f_1, f_2)$ of two polynomials in two unknowns.*

1. *If $\mathcal{S}(f_1, f_2) \equiv 0$ and f_1 and f_2 have degrees higher than one in each unknown, then the system $F(\mathbf{x}) = \mathbf{0}$ has an infinite number of solutions.*

2. *If $\mathcal{S}(f_1, f_2) \equiv c,\ c \in\ \mathbb{C}_0$, then the system $F(\mathbf{x}) = \mathbf{0}$ has no finite solutions.*

Proof. The proof equally consists of two parts.

1. By assumption, $\mathcal{S}(f_1, f_2) = c_{\mathbf{d}_1} \mathbf{x}^{\mathbf{d}_1} f_1 - c_{\mathbf{d}_2} \mathbf{x}^{\mathbf{d}_2} f_2 \equiv 0$.
   If $d_{11} \leq d_{21}$ and $d_{12} \leq d_{22}$, then f_1 can be written as the product of a monomial and f_2. Thus f_2 divides f_1. Similarly, if $d_{11} \geq d_{21}$ and $d_{12} \geq d_{22}$, then f_1 divides f_2. In these cases, there are an infinite number of solutions.
   If $d_{11} < d_{21}$ and $d_{12} > d_{22}$, then it follows from $\mathcal{S}(f_1, f_2) \equiv 0$ that
   $$c_{\mathbf{d}_1} x_2^{d_2'} f_1 - c_{\mathbf{d}_2} x_1^{d_1'} f_2 \equiv 0, \quad d_1' = d_{21} - d_{11} \text{ and } d_2' = d_{12} - d_{22}. \tag{2.16}$$
   Now let $f_1 = f_1' f'$ and $f_2 = f_2' f'$, with f' the greatest common factor of the polynomials f_1 and f_2. Because of (2.16), $f_1' = c_{\mathbf{d}_2} x_1^{d_1'}$ and $f_2' = c_{\mathbf{d}_1} x_2^{d_2'}$. The polynomial f' must have degrees higher than one in each unknown, because of the assumption that f_1 and f_2 have degrees higher than one in each unknown. So the system $F(\mathbf{x}) = \mathbf{0}$ has an infinite number of solutions. Analogously, we can prove the same for the case $d_{11} > d_{21}$ and $d_{12} < d_{22}$.

2. Because $\mathcal{S}(f_1, f_2) \in\ < f_1, f_2 >$, all common zeros of f_1 and f_2 are zeros of $\mathcal{S}(f_1, f_2)$. Now $\mathcal{S}(f_1, f_2) \equiv c \neq 0$, thus f_1 and f_2 have no common zero. ∎

The generalization of the second statement to more than two polynomial equations is straightforward. In this case, the system is said to be *inconsistent*. For more equations, the first statement does not hold anymore.

2.4 Description of the method

The aim of this section is construct an algorithm based on the concepts developed in the previous sections. The method is visualized in Figure 2.1.

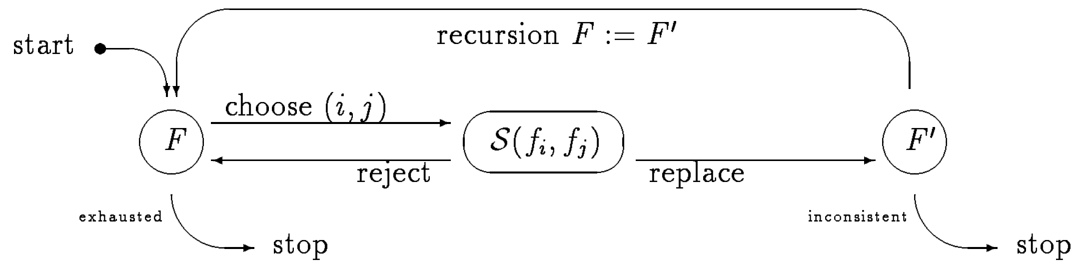


Figure 2.1: Object state diagram of the nonlinear reduction algorithm. There are three objects: the original system, the reduced system and the S-polynomial connecting them. The actions are represented by the arrows.

This section is further organized as follows. A detailed description of the algorithms is given in Section 2.4.1. In Section 2.4.2, the method is illustrated with an elaborated example. By controlling the growth of the enumeration tree (as explained in Section 2.4.3), the algorithm can be proven to run in polynomial time. This proof appears in Section 2.4.4.

2.4.1 The algorithms

Algorithm 2.4.1 provides an implementation of the replacement criteria, developed in Section 2.3. The criterion of Proposition 2.3.7 serves as a first check. Only when this simple criterion fails, Theorem 2.3.11 is applied. The second part of the condition in Theorem 2.3.11 is only applied when the first part gives a negative result.

Algorithm 2.4.1 Check the replacement criteria.

Input: f_1, f_2, $\mathcal{S}$.	$\mathcal{S} = \mathcal{S}(f_1, f_2)$
Output: $o \in \{$ ok0, ok1, ok2, $\neg$ok $\}$.	$\neg$ok*:* $\mathcal{S}$ *may not replace* f_1
	ok# indicates granting criterion
if $lcm(Lt(f_1), Lt(f_2)) = Lt(f_1)$	*check Proposition* 2.3.7
then return ok0;	$\mathcal{S}$ *can replace* f_1
elsif $\mathcal{R}(\mathcal{R}(f_1, \mathcal{S}), f_2) = 0$	*check Theorem* 2.3.11
then return ok1;	*first part satisfied*
elsif $\mathcal{R}(\mathcal{R}(f_1, f_2), \mathcal{S}) = 0$	*check second part*
then return ok2;	$\mathcal{S}$ *can replace* f_1
else return $\neg$ok;	$\mathcal{S}$ *may not replace* f_1
end if.	

The general outline of nonlinear reduction is sketched in Algorithm 2.4.2. It is an enumerative backtracking algorithm. By exhaustive enumeration, S-polynomials are computed of every pair of polynomials. Only those S-polynomials $\mathcal{S}(f_i, f_j)$ with a lower degree than $\deg(f_i)$ or $\deg(f_j)$ will be examined as candidates to replace f_i or f_j. The backtracking occurs when several recursive calls are admitted at the same level, trying different replacements.

Algorithm 2.4.2 Nonlinear reduction by enumerative S-polynomial computation.

```
Input:  F.                                                  tuple of polynomials
Output: F'.                                             reduced equivalent system

Sort(F);                                          so that deg(f_i) ≥ deg(f_j), ∀i > j
for i in 1,2,...,n loop                                              enumeration
  for j in i+1,...,n loop
    S := S(f_i, f_j);                                          compute S-polynomial
    if S ∈ ℂ_0                                                  inconsistent system
     then F' := F;  f'_i := S;  return;                         terminate reduction
    end if;
    if S = 0                                                superfluous polynomial
     then F' := F;  f'_i := 0;                             eliminating f_i yields F'
     elsif deg(S) ≤ deg(f_j) < deg(f_i)                    f_i is candidate to replace
          or deg(f_j) < deg(S) ≤ deg(f_i)
        then if Check(f_i, f_j, S) ≠ ¬ok                        apply Algorithm 2.4.1
               then F' := F;  f'_i := S;                     replacing f_i yields F'
                    Reduce(F', F'');  F' := F'';                       recursive call
             end if;
        elsif deg(S) ≤ deg(f_i) = deg(f_j)              f_i and f_j are both candidate
           then if Check(f_i, f_j, S) ≠ ¬ok                     apply Algorithm 2.4.1
                 then F' := F;  f'_i := S;                   replacing f_i yields F'
                      Reduce(F', F''^a);                              recursive call
                 else  F''^a := F;                                      no reduction
                end if;
                if Check(f_j, f_i, S) ≠ ¬ok                     apply Algorithm 2.4.1
                 then F' := F;  f'_j := S;                   replacing f_j yields F'
                      Reduce(F', F''^b);                              recursive call
                 else  F''^b := F;                                      no reduction
                end if;
                F' ∈ {F''^a, F''^b} : D_F' minimal;           compare total degrees
    end if;
  end loop;
end loop.
```

As long as the degree of the computed S-polynomial yields a reduction to the total degree of the system, the algorithm continues. The algorithm naturally stops when all S-polynomials have a higher degree than the degrees of their polynomials. In addition to this, when $\mathcal{S} \in \mathbb{C}_0$, the system is inconsistent (see Theorem 2.3.14) and the enumeration also stops.

Note that the algorithm assumes an ordering among the polynomials and starts with those polynomials with highest degree. Indeed, it makes no sense that the algorithm keeps fiddling around with low degree, possibly linear, polynomials. By considering the highest degree polynomials first, we can hope for the most considerable reductions.

Algorithm 2.4.2 presents the basic version of nonlinear reduction. Its exhaustive nature needs to be controlled in order to be useful in practice. Before tuning this basic version, let us first examine its behavior by elaborating a small example.

2.4.2 An elaborated example

Recall the system F presented in Example 2.2.5. In Figure 2.2, the flow of the algorithm applied to this example is presented.

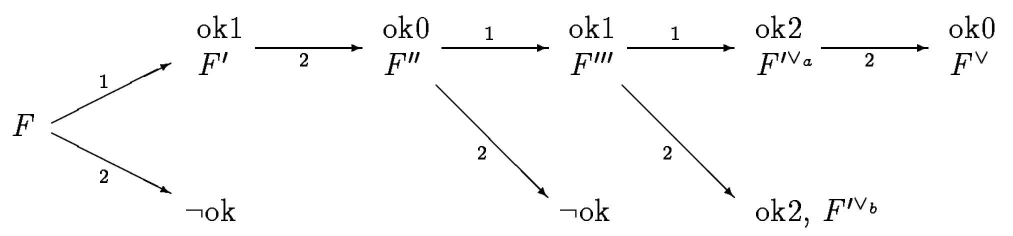


Figure 2.2: The flow of Algorithm 2.4.2 applied to an example. The nodes in the tree give the output of Algorithm 2.4.1 and, in case of replacement, the symbol of the reduced system. The numbers associated with the arrows indicate which polynomials are replaced.

There are two chains of reductions, respectively leading to F^{V} and $F^{\prime V_b}$. The first chain of reductions, $F \to F' \to F'' \to F''' \to F^{\prime V_a} \to F^{V}$, is given in (2.17) and (2.18). The differences among the systems reveal the S-polynomials.

$$\left\{ \begin{array}{c} x_1^3 - x_1 = 0 \\ x_1^2 x_2 + 1 = 0 \end{array} \right. \to \left\{ \begin{array}{c} -x_1 x_2 - x_1 = 0 \\ x_1^2 x_2 + 1 = 0 \end{array} \right. \to \left\{ \begin{array}{c} -x_1 x_2 - x_1 = 0 \\ -x_1^2 + 1 = 0 \end{array} \right. \tag{2.17}$$

$$\to \left\{ \begin{array}{c} -x_1^2 - x_2 = 0 \\ -x_1^2 + 1 = 0 \end{array} \right. \to \left\{ \begin{array}{c} -x_2 - 1 = 0 \\ -x_1^2 + 1 = 0 \end{array} \right. \to \left\{ \begin{array}{c} -x_2 - 1 = 0 \\ -x_1^2 - x_2 = 0 \end{array} \right. \tag{2.18}$$

The second part of the alternative chain of reductions, namely $F''' \to F^{\prime V_b}$ is displayed in (2.19).

$$\left\{ \begin{array}{c} -x_1^2 - x_2 = 0 \\ -x_1^2 + 1 = 0 \end{array} \right. \to \left\{ \begin{array}{c} -x_1^2 - x_2 = 0 \\ -x_2 - 1 = 0 \end{array} \right. \tag{2.19}$$

The final result of the algorithm is the system F^{V}, because the algorithm elaborates the tree of all possibilities using the preorder traversal. In Figure 2.2, this means that the reductions on the above branch are computed first, before backtracking to the other possibilities.

2.4.3 Controlling the exhaustive enumeration

Let us consider the case when Algorithm 2.4.2 is applied to a dense 3-dimensional linear system F, with the aim of obtaining a triangulation. In (2.20), the result of the first branch of the enumeration tree is displayed.

$$F \rightarrow \left\{ \begin{array}{r} f_1'(x_2, x_3) = 0 \\ f_2(x_1, x_2, x_3) = 0 \\ f_3(x_1, x_2, x_3) = 0 \end{array} \right. \rightarrow \left\{ \begin{array}{r} f_1'(x_2, x_3) = 0 \\ f_2''(x_2, x_3) = 0 \\ f_3(x_1, x_2, x_3) = 0 \end{array} \right. \rightarrow \left\{ \begin{array}{r} f_1'''(x_3) = 0 \\ f_2''(x_2, x_3) = 0 \\ f_3(x_1, x_2, x_3) = 0 \end{array} \right. \quad (2.20)$$

Considering the whole tree, there are six nodes directly under the root, representing all possible replacements using the three linear S-polynomials. At the next level, there is only one linear S-polynomial, yielding two branches. Proceeding to the next and last level, there are again two possibilities. Thus the enumeration tree has 24 leaves. Each leave corresponds to a different triangulation of the original linear system. Regardless whether this complete information is desired, it tends to be intractable to compute, as n increases.

The simplest way to prevent the enumeration of the whole tree is to impose a limit on the number of times we allow the algorithm to replace a polynomial by an S-polynomial without a reduction of the total degree. To this end we count the number of replacements without reduction, made since the computation of the reduced system with the lowest total degree. Adding this limit as an additional stopping criterion to Algorithm 2.4.2 is straightforward. In case of a linear system, as considered above, it suffices to set this limit equal to n, in order to obtain exactly one triangulation. Complementary to this limit, we have to determine a limit on the number of S-polynomials we allow to compute.

As alternative techniques to prune the enumeration tree, several more sophisticated methods can be presented. One of them is a greedy way of enumeration, by reorganizing the recursive calls in Algorithm 2.4.2. This greedy version considers several reduced systems obtained at the same level and performs the recursive call first with that reduced system with the lowest total degree. The number of reductions allowed at one level and the number of recursive calls are other additional parameters to tune in this greedy version.

2.4.4 Cost analysis

The purpose of this section is to get an impression of the cost of the nonlinear reduction method, by measuring the amount of arithmetical work needed at several stages of the method. Therefore, we count the number of replacement steps in Algorithm 2.4.2 and estimate the cost of computing the S-polynomials and R-polynomials.

Example 2.4.3 Consider the elaborated example presented in Section 2.4.2 and in particular Figure 2.2. The method required the computation of five S-polynomials and twenty R-polynomials.

It turns out that a crucial parameter for the cost of computing an S-polynomial is the *length of the polynomials*, i.e., the number of terms.

Proposition 2.4.4 *Consider two n-variate polynomials f_1 and f_2 of lengths l_1 and l_2 respectively. Denote $l = l_1 + l_2$. It takes $O(ln)$ operations to compute the S-polynomial or R-polynomial of f_1 and f_2.*

Proof. Consider first the computation of the S-polynomial. It takes $O(n)$ operations to compute the factors needed to multiply f_1 and f_2 with. Performing these multiplications requires respectively $O(l_1 n)$ and $O(l_2 n)$ operations, which is of order $O(ln)$, $l = l_1 + l_2$. The final subtraction requires $O(l)$ operations. Analogously, the same cost estimate can be obtained for the computation of the R-polynomial. □

The greatest fluctuations in the cost for computing an S-polynomial is caused by the lengths of the polynomials. If both polynomials are sparse, e.g., there are only two terms in each polynomial, then the cost is $O(n)$. For dense multivariate polynomials, the length grows exponentially as the dimension n increases, with an exponential cost to compute the S-polynomials. Fortunately, the degrees of the polynomials processed by Algorithm 2.4.2 are bounded, because it never admits the replacement of a polynomial by a polynomial with higher degree. Typically, for sparse systems with moderate degrees, the length of the polynomials is $O(n^2)$, which yields the acceptable cost $O(n^3)$ for the computation of one S-polynomial.

Proposition 2.4.5 *Let D be the total degree of the system. Denote by k the number of times the nonlinear reduction algorithm allows a replacement that does not reduce D. Then the number of computations of S-polynomials is bounded by $D + k$, in the optimal case, and is bounded by $(D + k)O(n^2)$, in the worst case.*

Proof. Consider the case where $k = 0$. There can be no more than D recursive calls in the algorithm, because otherwise there would be a replacement step which yields a system with the same total degree. In the optimal case, this reduction is the result of the replacement of the first S-polynomial computed in the enumeration, whereas in the worst case, only the last S-polynomial yields a reduction of the total degree. In that case, we have computed exactly $\frac{n(n-1)}{2}$ S-polynomials.

For $k > 0$, the algorithm can investigate other possibilities and explore other branches. In the optimal case, it is the second S-polynomial that yields the additional possibilities, allowing k additional S-polynomial computations. In the worst case, only the last $\frac{n(n-1)}{2}$th S-polynomial computed in the enumeration yields a reduction. □

Note that in the cases where the upper bounds are reached, the system is equivalent to a linear system and that by reduction, the problem is solved. Combining the results of Proposition 2.4.4 and Proposition 2.4.5 yields the proof that the nonlinear reduction algorithm runs in polynomial time, when considering D as part of the input.

Theorem 2.4.6 *Let D denote the total degree of the system. Assume the length of the polynomials is $O(l)$. Denote by k the number of times the nonlinear reduction algorithm allows a replacement that does not reduce D. Then number of arithmetic operations is bounded by $O((D + k)ln)$ in the optimal case and by $O((D + k)ln^3)$ in the worst case.*

Proof. The cost of computing an S-polynomial is $O(ln)$, by Proposition 2.4.4. The replacement criterion can require at most four computations of R-polynomials. So, the cost of a replacement step is $O(ln)$. Application of Proposition 2.4.5 finishes the proof. □

Note that the upper bounds for the cost correspond to the upper bounds for the cost of solving a polynomial system, using the start system based on the total degree, see [141]. However, when the reduction algorithm reaches this upper bound, the system is then also solved.

2.5 Numerical stability of the reduction process

2.5.1 How stable is reduction?

The purpose of the following is to show that the theoretical properties of the preceding section further hold, in spite of numerical calculations. This means that no finite solutions can be eliminated when highest degree terms are eliminated. Therefore, all attention is directed to the case where a coefficient of a highest degree term becomes zero, w.r.t. the *machine precision*, that is the smallest positive nonzero real number that can be represented exactly as a floating point number.

With a modification of the definition of S-polynomials, the moduli of the coefficients remain of the same magnitude and the rounding errors on the highest degree terms are of a magnitude ϵ equal to the machine precision ϵ_{mach}, which we denote by $\epsilon\ is\ O(\epsilon_{mach})$.

Definition 2.5.1 Given two polynomials f_1 and f_2, their *stable Subtraction-polynomial* or *stable S-polynomial* $\mathcal{S}'(f_1, f_2)$ is defined as follows. If $|Lc(f_1)| \le |Lc(f_2)|$, then

$$\mathcal{S}'(f_1, f_2) = \frac{lcm(Lt(f_1), Lt(f_2))}{Lt(f_1)} f_1 - \frac{Lc(f_1)}{Lc(f_2)} \frac{lcm(Lt(f_1), Lt(f_2))}{Lt(f_2)} f_2, \tag{2.21}$$

else

$$\mathcal{S}'(f_1, f_2) = \frac{Lc(f_2)}{Lc(f_1)} \frac{lcm(Lt(f_1), Lt(f_2))}{Lt(f_1)} f_1 - \frac{lcm(Lt(f_1), Lt(f_2))}{Lt(f_2)} f_2. \tag{2.22}$$

In the implementation of Algorithm 2.4.2, stable S-polynomials have to be used. As an alternative, the polynomial equations in the system can be scaled so that their leading coefficient is always equal to one.

2.5.2 A theorem about stability

We will now consider all systems in a neighborhood of the reduced system. The aim is to show that a perturbation of the coefficients of the highest degree monomials of magnitude ϵ_{mach} will not introduce any solutions we can consider as finite.

Let $H(\mathbf{x}, t)$ be a family of polynomial systems:

$$H : \mathbb{C}^n \times \mathbb{C} \rightarrow \mathbb{C}^n : (\mathbf{x}, t) \mapsto H(\mathbf{x}, t). \tag{2.23}$$

We assume that there exists a $t \in \mathbb{C}$ so that the system $H(\mathbf{x}, t) = \mathbf{0}$ in $\mathbf{x}$ admits only regular solutions. In particular, for every fixed $t^* \in \mathbb{C}$, there exists an $\epsilon > 0$ so that for all $t \in B(t^*, \epsilon) = \{ t \in \mathbb{C} \mid 0 < |t - t^*| < \epsilon \}$, the system $H(\mathbf{x}, t) = \mathbf{0}$ has only regular solutions. Let K be a connected component of the set $H_B^{-1}(\mathbf{0}) = \{ (\mathbf{x}, t) \mid H(\mathbf{x}, t) = \mathbf{0},\ t \in B(t^*, \epsilon) \}$.

Definition 2.5.2 The *cycle number* m of the connected component $K_t = \{ \mathbf{x} \mid (\mathbf{x}, t) \in K \}$ of the set $H_B^{-1}(\mathbf{0})$, for $t \in B(t^*, \epsilon)$, is defined as the number of elements in K_t.

Lemma 2.5.3 *Every bounded connected component K_t of the set $H_B^{-1}(\mathbf{0})$, for $t \in B(t^*, \epsilon)$, can be written as:*

$$\left\{ \begin{array}{l} t(\lambda) = t^* + \lambda^m \\ \mathbf{x}(\lambda) = \mathbf{x}^* + \mathbf{f}(\lambda), \quad \mathbf{f}(0) = \mathbf{0} \end{array} \right. \tag{2.24}$$

for a certain $\lambda \in \mathbb{C}$, where m equals the cycle number of K_t, and with $\mathbf{f}$ an analytic function in $B(0, \epsilon^{1/m})$.

Proof. Let φ denote the map $\varphi : \mathbb{C} \to \mathbb{C} : \lambda \mapsto t^* + \lambda^m$. Fix $(\mathbf{x}^{(0)}, t^{(0)}) \in K$ and choose $\lambda^{(0)} \in B(0, \epsilon^{1/m})$ so that $t^{(0)} = \varphi(\lambda^{(0)})$. For every $\lambda^{(1)} \in B(0, \epsilon^{1/m})$, there exist curves $\gamma : [0,1] \to B(0, \epsilon^{1/m})$ so that $\gamma(0) = \lambda^{(0)}$ and $\gamma(1) = \lambda^{(1)}$. For every γ, $\varphi \circ \gamma$ is a curve in $B(t^*, \epsilon)$. By the Implicit Function Theorem, K can locally be analytically parameterized by t. Hence, by analytic continuation (see [23]), with every $\varphi \circ \gamma$ and thus with every γ, corresponds a unique curve $\mathbf{g} : [0,1] \to \mathbb{C}^n$ so that $\mathbf{g}(0) = \mathbf{x}^{(0)}$ and $(\mathbf{g}, \varphi(\gamma(\theta))) \in K$ for all $\theta \in [0,1]$.

We can define $\mathbf{h}(\lambda^{(1)}) = \mathbf{g}(1)$, if $\mathbf{g}$ is independent of the curve γ connecting $\lambda^{(0)}$ with $\lambda^{(1)}$. Hence, it has to be proven that with every closed curve γ originating and terminating at $\lambda^{(0)}$, corresponds a closed curve $\mathbf{g}$ in $\mathbb{C}^n$. By the Monodromy Theorem (see [23]), the problem resumes to proving that the curve, corresponding to $\gamma(\theta) = \lambda^{(0)} e^{2\pi\theta i}$ satisfies $\mathbf{g}(1) = \mathbf{x}^{(0)}$.

Let $\mathbf{x}^{(k)} = \mathbf{g}(\frac{k}{m})$, $k \in \mathbb{Z}$, then $\mathbf{x}^{(k)} \in K_{t^{(0)}}$ and if the relation $\mathbf{x}^{(k)} = \mathbf{x}^{(k+j)}$ holds for a particular value of $k \in \mathbb{Z}$, then it also holds for all $k \in \mathbb{Z}$. As K is connected, every $(\mathbf{x}, t^{(0)}) \in K$ is connected to $(\mathbf{x}^{(0)}, t^{(0)})$ by a curve ρ in K, and by the Monodromy Theorem ρ can be chosen to be of the form

$$\rho \;:\; [0,1] \to K \;:\; \theta \mapsto \left(\mathbf{g}\left(\frac{l\theta}{m}\right), \varphi\left(\gamma\left(\frac{l\theta}{m}\right)\right)\right) = t^* + (t^{(0)} - t^*) e^{2\pi l \theta i} \tag{2.25}$$

where $l \in \mathbb{Z}$. Because $K_{t^{(0)}}$ has m elements, j equals m and the sequence $\{\mathbf{x}^{(k)}\}_{k \in \mathbb{Z}}$ is

$$\ldots, \mathbf{x}^{(m-1)}, \mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m-1)}, \mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m-1)}, \mathbf{x}^{(0)}, \ldots \tag{2.26}$$

where $\{\, \mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m-1)} \,\} = K_{t^{(0)}}$. In particular $\mathbf{g}(1) = \mathbf{g}(0)$ and the function $\mathbf{h}(\lambda)$ is thus well defined in $B(0, \epsilon^{1/m})$. Since $\mathbf{h}$ is locally the composition of two analytic maps, $\mathbf{h}$ is analytic.

Clearly, as K is connected, every $(\mathbf{x}, t) \in \mathbb{C}$ is of the form

$$\left\{ \begin{array}{l} t = \varphi(\lambda) = t^* + \lambda^m \\ \mathbf{x} = \mathbf{h}(\lambda) \end{array} \right. \tag{2.27}$$

for a certain $\lambda \in \mathbb{C}$. Because of the assumption that K is bounded, only removable singularities for $\mathbf{x}^*$ can occur, so the limit of $\mathbf{h}$ for $\lambda \to 0$ exists and equals $\mathbf{x}^*$.

Finally, let $\mathbf{f}(\lambda) = \mathbf{h}(\lambda) - \mathbf{h}(0)$ and the proof is complete. □

Let F' denote the exactly reduced system, without discarding the highest degree terms with a coefficient ϵ of order $O(\epsilon_{mach})$. Let $\widetilde{F}$ be the numerically reduced system, where we omitted all monomials with coefficient of order ϵ_{mach}. So we have $F' = \widetilde{F} + \epsilon E$, where

E is a polynomial system, having coefficients of the same magnitude as the coefficients of $\widetilde{F}$. The systems F' and $\widetilde{F}$ can be embedded in the homotopy $H(\mathbf{x},t) = \widetilde{F} + tE$. As the continuation parameter t goes from ϵ continuously to zero, each finite solution $\mathbf{y}^*$ of the system F' varies continuously to a solution $\mathbf{x}^*$ of the system $\widetilde{F}$ or diverges to infinity.

Theorem 2.5.4 *For each solution $\mathbf{y}^*$ of the system F', one of the following holds:*

1. *$\exists \mathbf{x}^*$, a finite solution of $\widetilde{F}$: $||\mathbf{y}^* - \mathbf{x}^*||$ is $O(\epsilon^{1/m})$,*
2. *$\exists \mathbf{x}^*$, a solution at infinity of $\widetilde{F}$: $\exists l:$ $\mathbf{y}_l^*$ is $O(\epsilon^{-1/m})$,*

where m is the cycle number of the connected component containing the solution $(\mathbf{x}^*, 0)$ of $H(\mathbf{x},t) = \mathbf{0}$.

Proof. The proof equally consists of two parts.

1. The path starting at $\mathbf{y}^*$ converges to a solution $\mathbf{x}^*$.

   By Lemma 2.5.3, each solution $(\mathbf{x},t)$ of $H(\mathbf{x},t) = \mathbf{0}$ can be parameterized, in a neighbourhood of $(\mathbf{x}^*, 0)$ by λ in the following way:

   $$\left\{ \begin{array}{l} t(\lambda) = 0 + \lambda^m \\ \mathbf{x}(\lambda) = \mathbf{x}^* + \mathbf{f}(\lambda), \quad \mathbf{f}(0) = \mathbf{0} \end{array} \right. \tag{2.28}$$

   where m is the cycle number of the connected component containing $(\mathbf{x}^*, 0)$. Expanding $\mathbf{f}(\lambda)$, for $\lambda \to 0$, we can write

   $$\mathbf{x}(\lambda) = \mathbf{x}^* + \lambda \mathbf{f}'(0) + O(\lambda^2)\mathbf{u}, \quad \text{with } ||\mathbf{u}|| = 1. \tag{2.29}$$

   Let t be equal to ϵ, then the solution $\mathbf{y}^*$ corresponds to the solution $(\mathbf{y}^*, \epsilon)$ of $H(\mathbf{x},t) = \mathbf{0}$ and we have that

   $$\mathbf{y}^* - \mathbf{x}^* = O(\epsilon^{1/m})\mathbf{v}, \text{ with } ||\mathbf{v}|| \; = \; 1 \tag{2.30}$$

   which completes the first part of the proof.

2. The path starting at $\mathbf{y}^*$ diverges to infinity.

   This means that the solution path starting at $\mathbf{y}^*$ ends in a solution at infinity $\mathbf{x}^*$. By the standard projective representation (using x_0 as homogenization coordinate), the solution at infinity can be written as $\mathbf{x}^* = (0, x_1^*, x_2^*, \ldots, x_n^*)$. Because Lemma 2.5.3 holds only for a bounded component K, the projective representation w.r.t. x_1 will be used to represent $\mathbf{x}^*$, assuming that $x_1^* \neq 0$. The representations for $\mathbf{x}^*$ and $\mathbf{y}^*$ are the following:

   $$\mathbf{x}^* = \left(0, 1, \frac{x_2^*}{x_1^*}, \ldots, \frac{x_n^*}{x_1^*}\right) \quad \text{and} \quad \mathbf{y}^* = \left(\frac{1}{y_1^*}, 1, \frac{y_2^*}{y_1^*}, \ldots, \frac{y_n^*}{y_1^*}\right) \tag{2.31}$$

   The first part of the proof can now be applied, because with this representation all solutions are finite. So, it follows that

   $$x_1(\lambda) = \frac{1}{x_0^* + f_0(\lambda)} = \frac{1}{f_0(\lambda)} \approx \frac{1}{r\lambda^k}, \quad \text{with } r \neq 0 \text{ and some } k \geq 1. \tag{2.32}$$

   Let t be equal to ϵ, then we have that y_1^* is $O(\epsilon^{-1/m})$. □

Because for most applications m equals 1 for every solution, all theoretical properties of Section 2.3 hold, when the reduction algorithm does all its calculations numerically.

2.6 Applications

2.6.1 The polynomial systems

Our method has been applied to several deficient polynomial systems which appeared in the literature [21, 90, 103, 151, 180]. This choice of the examples exhibits the variety of features available in the reduction algorithm.

An overconstrained system

The first system, taken from [21], cannot be solved directly by a continuation method, because there are more equations than unknowns. We could take the first two equations to solve it, which yields a system of total degree equal to twelve. However, it is better to apply the reduction algorithm first.

$$F(\mathbf{x}) = \left\{ \begin{array}{r} 3x_1^2x_2 + 9x_1^2 + 2x_1x_2 + 5x_1 + x_2 - 3 = 0 \\ 2x_1^3x_2 + 6x_1^3 - 2x_1^2 - x_1x_2 - 3x_1 - x_2 + 3 = 0 \\ x_1^3x_2 + 3x_1^3 + x_1^2x_2 + 2x_1^2 = 0 \end{array} \right. \tag{2.33}$$

During the reduction one equation of the system can be eliminated, because an S-polynomial becomes zero. This yields a reduced system with total degree equal to four. There are three real solutions.

An interpolating quadrature formula over a grid

For the construction of an n-point interpolating quadrature formula for integrating a function defined on a grid, the polynomial system $F(\mathbf{x}) = \mathbf{0}$, with jth equation displayed in (2.34), has to be solved.

$$f(\mathbf{w}) = \sum_{k=1}^{n} w_k \left(\frac{k}{2} - \tau \right)^j - m_j = 0, \quad j = 0, 1, \ldots, n \tag{2.34}$$

The constants m_j in (2.34) are given numbers. The unknowns of the system $F(\mathbf{w}) = \mathbf{0}$ are the weights $\mathbf{w} = (w_0, w_1, \ldots, w_n)$ of the quadrature formula. This problem occurred in the context of wavelet functions, see [151] for special purpose (and hence more efficient) methods for solving this problem. We solved this problem with $n = 4$, which renders a polynomial system whose total degree equals 120. The system has four complex conjugate solutions and one real solution. Applying the reduction algorithm, only eight solution curves remain to be traced.

Optimizing the Wood function

This system is derived from optimizing the Wood function [103]:

$$F(\mathbf{x}) = \left\{ \begin{array}{r} 200x_1^3 - 200x_1x_2 + x_1 - 1 = 0 \\ -100x_1^2 + 110.1x_2 + 9.9x_4 - 20 = 0 \\ 180x_3^3 - 180x_3x_4 + x_3 - 1 = 0 \\ -90x_3^2 + 9.9x_2 + 100.1x_4 - 20 = 0 \end{array} \right. . \tag{2.35}$$

The total degree of this system equals 36, whereas there are three real solution and six complex conjugate solutions. Applying the reduction algorithm, the system is transformed into an equivalent system with a total degree equal to 16.

A lumped-parameter chemically reacting system

The fourth example is the mathematical model of a lumped-parameter chemically reacting system, see [90]:

$$F(\mathbf{x}) = \begin{cases} c_1x_1x_3 + c_1x_1x_4 - (c_1+1)x_1 + c_2x_3 + c_6 = 0 \\ c_3x_2x_3 + c_3x_2x_4 - (c_3+1)x_2 + c_4x_4 + c_7 = 0 \\ -c_1x_1x_3 - c_1x_1x_4 - c_5x_3x_4 + c_1x_1 - c_2x_3 = 0 \\ -c_2x_2x_3 - c_3x_2x_4 - c_5x_3x_4 + c_3x_2 - c_4x_4 = 0 \end{cases} . \tag{2.36}$$

After linear and nonlinear reduction, the total degree of the reduced system equals four. So we can compute all finite solutions without any wasted computations.

A electrochemistry example: problem 601

The last example is a system derived from an application in the field of electrochemistry. It is known as problem 601 in [180].

$$F(\mathbf{x}) = \begin{cases} c_1x_2^6 + c_2x_2^5 + c_3x_2^4 + c_4x_1^2x_3 + c_5x_2^3 + c_6x_2^2 + c_7x_2 + c_8 = 0 \\ c_9x_2^5 + c_{10}x_2^4 + c_{11}x_1^2x_2 + c_{12}x_1^2x_3 + c_{13}x_2^3 \\ \quad +c_{14}x_1x_2 + c_{15}x_2^2 + c_{16}x_2 + c_{17} = 0 \\ c_{18}x_1^2 + c_{19}x_1x_3 + c_{20}x_2 + c_{21} = 0 \end{cases} \tag{2.37}$$

The total degree of F equals sixty, while there are only fourteen finite solution. Before reducing this system, its coefficients must be scaled. The scaling procedure, presented in [104, Chapter 5], for scaling equations and unknown proves to be very useful for this type of application. After application of reduction on the scaled system, the total degree equals forty.

2.6.2 Performance

In Table 2.1, a summary of the performance of the reduction algorithm is given. It is divided in three parts. For each system, the first column indicates whether linear reduction was useful or not (a 'yes' or a 'no' respectively in the table). The second column lists the number of replacements with an equal total degree (#eq), given as input to the algorithm in order to control the growth of the enumeration tree. The next part of the table then displays some output statistics, such as the number of computed S-polynomials (#S) and the number of computed R-polynomials (#R). This last number measures the performance of the simple criterion. In the third part of Table 2.1, the total degrees D_F and $D_{F'}$ of the original and reduced system are given, compared with the actual number of finite solutions, denoted by $N(F)$.

The choice of the examples illustrates the effectiveness of the second replacement criterion and indicates the usefulness of linear reduction as preprocessor facility. From

system	description	lin	#eq	#S	#R	D_F	$D_{F'}$	$N(F)$
(2.33)	overconstrained	yes	5	22	16	12	4	3
(2.34)	quadrature over grid	no	8	150	0	120	8	5
(2.35)	optimizing Wood	no	1	18	0	36	16	9
(2.36)	lumped-parameter	yes	3	24	8	16	4	4
(2.37)	electrochemistry	no	0	9	0	60	40	15

Table 2.1: Performance of reduction: the importance of linear reduction, the amount of operations and the total degrees versus the actual number of finite solutions.

Table 2.1, we see that the greatest amount of work yielded the biggest reduction. Analogously, the same holds for systems where the smallest reduction could be obtained. So, the more work the reduction algorithm does, the better the outcome.

Compared to the cost of continuation, the reduction is very effective. The length of the S-polynomials is bounded in all cases by $O(n^2)$, so the cost of computing an S-polynomial is $O(n^3)$, which is equivalent to the cost of solving a linear system. In practice, about one hundred linear systems have to be solved when numerically tracing a solution path. In Table 2.1, we see that the cost of the reduction algorithm never exceeds the cost of computing all linear systems during the following of one or two solution paths. So, applying the reduction algorithm is a big deal.

2.7 Conclusions and future research

Applying concepts of symbolic elimination methods enabled us to develop an algorithm for eliminating solutions at infinity, so that very often divergent solution curves can be avoided. The reduction algorithm provides an efficient preprocessor for solving polynomial systems by homotopy continuation methods.

Despite its usefulness, the current version of the nonlinear reduction algorithm is not a very sophisticated one. For instance, instead of plainly enumerating all possible choices of the polynomials in the system for computing S-polynomials, we should be able to predict a priori which S-polynomials will yield a reduction, before actually computing all of them. Another topic for further research could be the development of a reduction method with an eye to other root counts than the plain total degree, such as the ones presented in the forthcoming chapters.

S-polynomials are a key tool in the construction of a Gröbner basis, which can be used as an elimination method for solving polynomial systems, see e.g. [21, 32]. Compared to the Buchberger algorithm for constructing Gröbner bases, the nonlinear reduction method makes controlled use of S-polynomials, in the following three ways. First, the length of the S-polynomials is bounded, because we never consider any S-polynomial with degrees higher than those of its input polynomials. Second, the number of polynomials in the system never increases, because either the S-polynomial is used to replace a polynomial in the system, or is discarded. Third, because stable S-polynomials are used and the number of replacements is limited, the numerical stability of the solver is equally controlled, whereas elimination methods are forced to work with a computationally expensive symbolic representation of the numbers to obtain meaningful results.

As a conclusion, we can say that we switch to homotopy continuation methods before the elimination process runs out of hand. It would be an interesting project to investigate where the break-even point lies between methods aimed to solve systems with random coefficients, such as homotopy continuation, and solvers which count on systems with specially chosen coefficients to run efficiently.

Chapter 3

Multi-Homogenization

This chapter deals with the construction of homotopies based on the application of Bézout's theorem to multi-homogeneous polynomial systems.

Multi-homogenization allows to exploit the structure of the deficient system by constructing a partition of the set of unknowns. For many applications, this partition is obvious, as we will briefly mention in the following examples. Solvers for eigenvalue problems consider the eigenvalue separately from the components of the eigenvector. The partition for interpolating integration formulae takes the knots apart from the weights. Inverse kinematics of robot arms invite a grouping of sines and cosines of the corresponding angles between the links. This approach leads for many applications to a significantly sharper root count than the total degree.

The use of multi-homogeneous start systems in homotopies has been proposed by Morgan and Sommese in [106, 107]. In [177], Wampler, Morgan and Sommese indicated how to construct a multi-homogeneous start system. Deficient multi-homogeneous polynomial systems have been investigated by Li and Wang, see [90]. The problem of computing Bézout numbers and finding the optimal multi-homogeneous structure has been treated in [175] by Wampler and in [176], he proposed efficient multi-homogeneous start systems.

In the sequel we discuss the following three basic problems: the construction of the start system, the computation of the Bézout number and the search for an optimal multi-homogeneous structure. We addressed in [158] the last problem and proposed to construct multi-homogeneous start systems by interpolation, a result that was published in [160]. The more efficient way we presented in [167] for computing multi-homogeneous Bézout numbers was called 'the basic algorithm' by Wampler in [175].

The outline of this chapter is as follows. First we give the statement of Bézout's theorem in multi-projective space. The second section deals with the construction of multi-homogeneous start systems constructed by interpolation. An algorithm for the construction of multi-homogeneous linear product systems is presented in the third section. In the fourth section the problem of computing minimal Bézout numbers is discussed. Applications are given in the fifth section. Finally, we state our conclusions and suggest directions for future research.

3.1 Bézout's theorem in multi-projective space

This section is concerned with the definitions and the statement of Bézout's theorem in multi-projective space. We refer to [140] for a more general theoretical description of the material presented here. First we define multi-projective space and multi-homogeneous systems. Then we state Bézout's theorem in multi-projective space, which yields an upper bound for the number of isolated solutions in $\mathbb{C}^n$.

3.1.1 Multi-projective space

Definition 3.1.1 We introduce *the multi-projective space* as the Cartesian product of m projective spaces. In particular, for $\mathbf{k} \in \mathbb{N}^m$ with $n = \sum_{j=1}^m k_j$, *the m-projective space* $\mathbb{P}^{(n,m)}$ is defined by

$$\mathbb{P}^{(n,m)} = \mathbb{P}^{k_1} \times \mathbb{P}^{k_2} \times \cdots \times \mathbb{P}^{k_m}. \tag{3.1}$$

We can view $\mathbb{P}^{(n,m)}$ as $\mathbb{C}^n$ extended with m hyperplanes at infinity, each characterized by $z_{j0} = 0$, for $j = 1, 2, \ldots, m$. Analogously to the ordinary 1-projective space, we can prove the following proposition.

Proposition 3.1.2 $\mathbb{P}^{(n,m)} \setminus \{z_{j0} = 0 \mid j = 1, 2, \ldots, m\}$ *is isomorphic to* $\mathbb{C}^n$ *by the map*

$$\begin{aligned} \phi \ : \ & \mathbb{P}^{(n,m)} \setminus \{z_{j0} = 0 \mid j = 1, 2, \ldots, m\} \to \mathbb{C}^n \\ & \mathbf{z} = (z_{10}, z_{11}, \ldots, z_{1k_1}, z_{20}, z_{21}, \ldots, z_{2k_2}, \ldots, z_{m0}, z_{m1}, \ldots, z_{mk_m}) \\ & \quad \mapsto \phi(\mathbf{z}) = \left(\tfrac{z_{11}}{z_{10}}, \ldots, \tfrac{z_{1k_1}}{z_{10}}, \tfrac{z_{21}}{z_{20}}, \ldots, \tfrac{z_{2k_2}}{z_{20}}, \ldots, \tfrac{z_{m1}}{z_{m0}}, \ldots, \tfrac{z_{mk_m}}{z_{m0}}\right). \end{aligned} \tag{3.2}$$

Denote by $X = \{x_1, x_2, \ldots, x_n\}$ the set of unknowns of a multivariate polynomial.

Definition 3.1.3 Consider $\mathbf{k} \in \mathbb{N}^m$, with $\sum_{j=1}^m k_j = n$. An *m-partition* Z *of* X is a partition $Z = \{Z_1, Z_2, \ldots, Z_m\}$ of X with $\#Z_j = k_j$. An *m-homogeneous partition* $\widetilde{Z}$ *of* X, $\widetilde{Z} = \{\widetilde{Z}_1, \widetilde{Z}_2, \ldots, \widetilde{Z}_m\}$ is derived from an m-partition Z of X as follows $\widetilde{Z}_j := Z_j \cup \{z_{j0}\}$, for $j = 1, 2, \ldots, m$.

An m-projective space $\mathbb{P}^{(n,m)}$ is uniquely defined by a given m-homogeneous partition $\widetilde{Z}$ of a set X of n unknowns.

Definition 3.1.4 Let $Y \subseteq X$ and consider a monomial $\mathbf{x}^{\mathbf{d}}$. Define *the degree of* $\mathbf{x}^{\mathbf{d}}$ *in* Y by $\deg(\mathbf{x}^{\mathbf{d}}, Y) = \sum_{x_i \in Y} d_i$. Consider $f = \sum_{k=0}^{l} c_{\mathbf{d}_k} \mathbf{x}^{\mathbf{d}_k}$. Then *the degree of* f *in the set* Y is defined by $\deg(f, Y) = \max\{ \deg(\mathbf{x}_k^{\mathbf{d}}, Y) \mid c_{\mathbf{d}_k} \neq 0 \}$. Naturally, $\deg(f, X) = \deg(f)$.

Now we can define *multi-homogeneous polynomials* and *multi-homogeneous systems.*

Definition 3.1.5 A polynomial f is *m-homogeneous w.r.t.* Z if for every $Z_j \in Z$ and for every term $c_{\mathbf{d}}\mathbf{x}^{\mathbf{d}}$ of f, $\deg(\mathbf{x}^{\mathbf{d}}, Z_j) = \deg(f, Z_j)$ or equivalently, if $f(\ldots, cz_{j0}, cz_{j1}, \ldots, cz_{jk_j}, \ldots) = c^d f(\mathbf{z})$, with $d = \deg(f, Z_j)$, for $j = 1, 2, \ldots, m$. A polynomial system given by the tuple $F = (f_1, f_2, \ldots, f_n)$ is *m-homogeneous w.r.t.* Z if all its polynomials f_i are m-homogeneous w.r.t. Z, for $i = 1, 2, \ldots, n$.

By a *multi-projective transformation*, we can make a given polynomial multi-homogeneous.

Definition 3.1.6 Consider a polynomial f and an m-homogeneous partition $\widetilde{Z}$. We define *the m-projective transformation $\widetilde{f}$ of f w.r.t. $\widetilde{Z}$* by $\widetilde{f}(\mathbf{z}) = z_{10}^{d_1} z_{20}^{d_2} \cdots z_{m0}^{d_m} f(\phi(\mathbf{z}))$, with $d_j = \deg(f, Z_j)$, for $j = 1, 2, \ldots, m$. Let $F = (f_1, f_2, \ldots, f_n)$ denote a tuple of polynomials, then *the m-projective transformation $\widetilde{F}$ of F w.r.t. $\widetilde{Z}$* is defined by $\widetilde{F} = (\widetilde{f}_1, \widetilde{f}_2, \ldots, \widetilde{f}_n)$, with $\widetilde{f}_i$ the m-projective transformation of f_i, for $i = 1, 2, \ldots, n$.

Example 3.1.7 Consider the following polynomial system [107]:

$$F(\mathbf{x}) = \left\{ \begin{array}{c} x_1^2 - 1 = 0 \\ x_1 x_2 - 1 = 0 \end{array} \right. . \tag{3.3}$$

In order to 2-homogenize the system, consider the partition $Z = \{Z_1, Z_2\}$ of the set $\{x_1, x_2\}$ defined by $Z_1 = \{x_1\}$ and $Z_2 = \{x_2\}$. We rename the unknowns: $z_{11} = x_1$ and $z_{21} = x_2$. Adding to each set Z_j a homogeneous unknown z_{j0}, denote $\widetilde{Z}_j = Z_j \cup \{z_{j0}\}$. Thus $\widetilde{Z}_1 = \{z_{10}, z_{11}\}$ and $\widetilde{Z}_2 = \{z_{20}, z_{21}\}$. Now we can 2-homogenize the system:

$$F(\phi(\mathbf{z})) = \left\{ \begin{array}{c} \left(\frac{z_{11}}{z_{10}}\right)^2 - 1 = 0 \\ \left(\frac{z_{11}}{z_{10}}\right)\left(\frac{z_{21}}{z_{20}}\right) - 1 = 0 \end{array} \right. \quad \Rightarrow \quad \widetilde{F}(\mathbf{z}) = \left\{ \begin{array}{c} z_{11}^2 - z_{10}^2 = 0 \\ z_{11} z_{21} - z_{10} z_{20} = 0 \end{array} \right. . \tag{3.4}$$

In order to obtain a system with as many equations as unknowns, two linear equations can be added to $\widetilde{F}$. The coefficients of these added linear equations should be taken at random. ■

3.1.2 Bézout's theorem

Definition 3.1.8 Consider a tuple $F = (f_1, f_2, \ldots, f_n)$ of n-variate polynomials in X and an m-partition Z of X. Then *the degree matrix of F w.r.t. Z* is defined by the matrix $M_D \in \mathbb{N}^{n \times m}$, with its (i,j)th entry given by $\deg(f_i, Z_j)$, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$. We denote $M_D = \deg(F, Z)$.

An m-partition Z and a degree matrix M_D determine together an *m-homogeneous structure of F*. Denote for a given m-partition Z and a degree vector $\mathbf{d} \in \mathbb{N}^m$ the class of all polynomials with that m-homogeneous structure by $\mathbb{C}^{(Z,\mathbf{d})}[\mathbf{x}] = \{\, f \in \mathbb{C}[\mathbf{x}] \mid \deg(f, Z) = \mathbf{d}\}$. Each multi-homogeneous structure uniquely defines a *multi-homogeneous Bézout number.*

Definition 3.1.9 Let $M_D \in \mathbb{N}^{n \times m}$ be the degree matrix of the system $F(\mathbf{x}) = \mathbf{0}$ w.r.t. an m-partition Z. Recall that the vector $\mathbf{k} \in \mathbb{N}^m$ determines the cardinalities of the sets in Z. The polynomial $f_{M_D}(\mathbf{x}) = \prod_{i=1}^n \sum_{j=1}^m d_{ij} x_j$, with d_{ij} the (i,j)th entry in the matrix M_D, is *the degree polynomial associated with M_D*. We define B as *the m-homogeneous Bézout number of F w.r.t. Z* which equals the coefficient of $\mathbf{x}^{\mathbf{k}}$ in the polynomial $f_{M_D}(\mathbf{x})$.

In [140], we can find a proof for the following theorem.

Theorem 3.1.10 (Bézout in $\mathbb{P}^{(n,m)}$) *If a polynomial system F has a finite number of solutions, then there are in $\mathbb{P}^{(n,m)}$ exactly B isolated solutions, counted with multiplicities.*

Example 3.1.11 (Example 3.1.7 continued.) Consider the polynomial system $F(\mathbf{x}) = \mathbf{0}$ given in (3.3). To compute the 2-homogeneous Bézout number, we associate with each set $\widetilde{Z}_k$ the unknown α_k and construct the degree polynomial f_{M_D} with as coefficients the corresponding degrees d_{ij}'s taken from the matrix M_D:

$$M_D = \begin{bmatrix} 2 & 0 \\ 1 & 1 \end{bmatrix}, \quad f_{M_D}(\alpha_1, \alpha_2) = (2\alpha_1 + 0\alpha_2)(1\alpha_1 + 1\alpha_2) = 2\alpha_1^2 + 2\alpha_1\alpha_2. \tag{3.5}$$

The 2-homogeneous Bézout number B is given by the coefficient of the term $\alpha_1^1\alpha_2^1$ as 1 and 1 are the cardinalities of Z_1 and Z_2 in the partition of $\{x_1, x_2\}$. For this system B provides an exact root count for F, whereas the total degree D of F is twice as high. ∎

In [107], Morgan and Sommese proved the following theorem.

Theorem 3.1.12 *Consider a polynomial system $F(\mathbf{x}) = \mathbf{0}$ and an m-homogeneous structure given by the m-partition Z and the corresponding degree matrix M_D. Let $F^{(0)}(\mathbf{x}) = \mathbf{0}$ denote a system with* $\deg(F^{(0)}, Z) = M_D$ *and with exactly B regular solutions in $\mathbb{C}^n$. Then the usual convex-linear homotopy is a good homotopy for obtaining all isolated solutions of $F(\mathbf{x}) = \mathbf{0}$.*

As pointed out by Li and Wang in [90], it is crucial that the start system $F^{(0)}(\mathbf{x}) = \mathbf{0}$ is not deficient w.r.t. the Bézout number B determined by the particular m-homogeneous structure of $F(\mathbf{x}) = \mathbf{0}$.

For many applications $B \ll D$, so that using a start system based on a multi-homogeneous structure significantly improves the performance of the homotopy.

3.2 Interpolating start systems

The first method we present for constructing start systems w.r.t. a given m-homogeneous structure is based on interpolation. This yields an extended Newton homotopy, which allows to obtain all isolated solutions of the target system, provided there are B independent roots. The interpolation requires the solution of as many linear systems as the number of independent roots times the number of polynomials.

3.2.1 An extended Newton homotopy

A standard homotopy, frequently applied to enlarge the domain of convergence of Newton's method, is the so-called Newton homotopy.

Definition 3.2.1 Consider a polynomial system $F(\mathbf{x}) = \mathbf{0}$ and $\mathbf{x}^{(0)} \in \mathbb{C}^n$. Then

$$H(\mathbf{x}, t) = F(\mathbf{x}) - F(\mathbf{x}^{(0)})t = t(F(\mathbf{x}) - F(\mathbf{x}^{(0)})) + (1-t)F(\mathbf{x}) \tag{3.6}$$

is called the *Newton homotopy* for computing a solution of $F(\mathbf{x}) = \mathbf{0}$.

Note that with the Newton homotopy, the continuation parameter t varies from 1 to 0, defining a solution path starting at the given point $\mathbf{x}^{(0)}$. Despite its enlarged domain of convergence, we cannot guarantee that the path converges to a solution of $F(\mathbf{x}) = \mathbf{0}$, even when the target system has a solution.

We can view the vector $\mathbf{x}^{(0)}$ as an interpolating vector for the tuple of polynomials in F. The idea is to construct a tuple $F^{(0)}$ of multivariate polynomials by interpolation. If the interpolating vectors are regular solutions of the system defined by that tuple, then we have an *interpolating start system* and subsequently an *interpolating homotopy*. The Newton homotopy is the simplest example of an interpolating homotopy, with $F^{(0)}(\mathbf{x}) = F(\mathbf{x}) - F(\mathbf{x}^{(0)}) = \mathbf{0}$ as interpolating start system.

The approach we elaborate in this section extends the use of the Newton homotopy, in the sense that we choose B points to start with. This gives rise to an *extended Newton homotopy*. By Theorem 3.1.12, we are certain to obtain all the isolated solutions of the target system. The interpolation method allows to take advantage of the fact that we might already know some of the roots and that we are interested in the remaining solutions of the system. By interpolating through these known roots, we can avoid recalculating them. Unfortunately, the number of roots that can be chosen independently for a given structure is limited, as will be discussed next.

3.2.2 Independent roots of polynomial system

In [61, page 371], we find the following theorem of special interest.

Theorem 3.2.2 (Cayley-Bacherach) *If C_1 and C_2 are curves in $\mathbb{P}^2$ of respective degrees d_1 and d_2 meeting at $d_1 d_2$ distinct points, then any curve E of degree $d_1 + d_2 - 3$ that passes through all but one point of $C_1 \cap C_2$ necessarily passes through that remaining point also.*

Example 3.2.3 Consider a specific example of this theorem, for $d_1 = 3 = d_2$, i.e., the situation of two cubics in the plane. By applying Theorem 3.2.2, we can state that any cubic passing through eight of nine intersection points must also contain the remaining one. This implies that we cannot choose at random nine points in the plane and then claim that these points are the regular solutions of a system of two cubics. ∎

For the problem of Example 3.2.3, there are eight *independent roots*, the ninth one is dependent from the other ones. We are only interested in independent regular solutions, which we simply call independent in the sequel.

Another important issue is the representation of the interpolating polynomial, in particular the number of monomials we allow to have a nonzero coefficient. Consider for instance the univariate case. If we use $x^d - c$ as interpolating polynomial, then we can choose only one root at random, leading to a fixed value of c and the determination of all other roots. If we take a dense interpolating polynomial instead, then we can choose independently d numbers to be the roots. So, the more monomials, the larger the number of independent roots. Therefore, we propose to consider mainly dense interpolating polynomials to represent the polynomials in $\mathbb{C}^{(Z,\mathbf{d})}[\mathbf{x}]$.

Definition 3.2.4 The polynomial $f(\mathbf{x}) = \sum_{k=0}^{l} c_{\mathbf{d}_k}\mathbf{x}^{\mathbf{d}_k} \in \mathbb{C}^{(Z,\mathbf{d})}[\mathbf{x}]$ is a dense representation w.r.t. Z if $\deg(f, Z) \neq \deg(f + c_{\mathbf{d}'}\mathbf{x}^{\mathbf{d}'}, Z)$ for every $c_{\mathbf{d}'}\mathbf{x}^{\mathbf{d}'}$ that causes $f + c_{\mathbf{d}'}\mathbf{x}^{\mathbf{d}'}$ to have $l+2$ terms.

The first stage of the interpolation method consists in creating a dense representation for a given m-homogeneous structure, by adding all candidate monomials $\mathbf{x}^{\mathbf{d}}$ for which $\deg(\mathbf{x}^{\mathbf{d}}, Z_j) \leq \deg(f_i, Z_j)$ to f_i, for all $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$. We denote the obtained dense representation of F by F^*.

Example 3.2.5 (Example 3.1.11 continued.) Consider the polynomial system F given in (3.3), with a 2-homogeneous structure presented in Example 3.1.11. We can enlarge F to obtain the following system

$$F^*(\mathbf{x}) = \left\{ \begin{array}{c} x_1^2 + x_1 - 1 = 0 \\ x_1x_2 + x_1 + x_2 - 1 = 0 \end{array} \right. . \tag{3.7}$$

The term x_2 could not be added to the first equation of F, because that would have violated the 2-homogeneous structure. ∎

3.2.3 Constructing the interpolating polynomials

Proposition 3.2.6 *Consider $f(\mathbf{x}) = \sum_{k=0}^{l} c_{\mathbf{d}_k}\mathbf{x}^{\mathbf{d}_k}$ and denote by $C_f = \{\ \mathbf{x} \mid f(\mathbf{x}) = 0\ \}$ the curve defined by f. Except for a choice of vectors belonging to a set of measure zero, C_f is uniquely determined by any choice of l interpolating vectors $\mathbf{w}^{(i)} \in \mathbb{C}_0^n$ satisfying $f(\mathbf{w}^{(i)}) = 0$, for $i = 1, 2, \ldots, l$.*

Proof. Without loss of generality, assume $\mathbf{d}_0 = \mathbf{0}$. The conditions $f(\mathbf{w}^{(i)}) = 0$ determine l linear equations on the coefficients of f, yielding a linear system, denoted by $A\mathbf{c} = \mathbf{b}$, with $A \in \mathbb{C}^{l\times l}$, $\mathbf{b} \in \mathbb{C}^l$. Take at random $c_{\mathbf{d}_0} \in \mathbb{C}_0$ and define $b_i = -c_{\mathbf{d}_0}$, $a_{ij} = \mathbf{w}^{(i)\mathbf{d}_j}$. The solution of $A\mathbf{c} = \mathbf{b}$ then renders the coefficients of f: $c_k = c_{\mathbf{d}_k}$, $k = 1, 2, \ldots, l$.

If $\det(A) \neq 0$, then the coefficients $c_{\mathbf{d}_1}$, $c_{\mathbf{d}_2}$, ..., $c_{\mathbf{d}_l}$ are uniquely determined upon the choice of $c_{\mathbf{d}_0}$. This implies that the equation $f(\mathbf{x}) = 0$ is uniquely determined and hence also the curve C_f. The exceptional situation, represented by $\det(A) = 0$, determines a set of dimension not higher than $N-1$ in $\mathbb{C}_0^N$, $N = n^l$, that is a set of measure zero in $\mathbb{C}_0^N$, with $\mathbb{C}_0^N$ the set of all possible choices for the interpolating vectors $\mathbf{w}^{(i)}$, where $i = 1, 2, \ldots, l$. □

One necessary condition to have only regular roots is that the interpolating polynomials are linearly independent from each other. This can be achieved by fixing a priori some coefficients such that the coefficient matrix always has rank n. In order to achieve this, we represent the tuple of interpolating polynomials by considering polynomials that each have at least one monomial that does not belong to any other polynomial of the tuple.

Definition 3.2.7 Consider $F = (f_1, f_2, \ldots, f_n)$ with $f_i(\mathbf{x}) = \sum_{k=0}^{l_i} c_{\mathbf{d}_k^{(i)}}\mathbf{x}^{\mathbf{d}_k^{(i)}}$ and an m-homogeneous structure given by the m-partition Z and the degree matrix M_D. F is an *independent representation w.r.t. Z and M_D* if

$$\deg(F, Z) = M_D \quad \text{and} \quad \forall i, \exists k_1 : \ \mathbf{d}_{k_1}^{(i)} \neq \mathbf{d}_{k_2}^{(j)}, \ \forall j \neq i, k_2 = 0, 1, \ldots, l_j. \tag{3.8}$$

F is an *independent dense representation w.r.t. Z and M_D* if the addition of any $c_{\mathbf{d}}\mathbf{x}^{\mathbf{d}}$ to f_i that causes f_i to have $l_i + 2$ terms violates (3.8).

Example 3.2.8 $f_i(\mathbf{x}) = x_i + 1$ is the ith polynomial in an independent dense representation of a linear system, with $Z = \{\{x_1, x_2, \ldots, x_n\}\}$. ■

Theorem 3.2.9 shows how to count the number of independent roots of a polynomial system.

Theorem 3.2.9 *Consider $F = (f_1, f_2, \ldots, f_n)$, with $f_i(\mathbf{x}) = \sum_{k=0}^{l_i} c_{\mathbf{d}_k^{(i)}} \mathbf{x}^{\mathbf{d}_k^{(i)}}$, for $i = 1, 2, \ldots, n$. Suppose F is an independent dense representation w.r.t. an m-homogeneous structure. Let $l = \min(\min_{i=1}^{n} l_i, B)$. Except for a set of measure zero, by any choice of l interpolating vectors $\mathbf{w}^{(j)} \in \mathbb{C}_0^n$ satisfying $F(\mathbf{w}^{(j)}) = \mathbf{0}$, for $j = 1, 2, \ldots, l$, F has the vectors $\mathbf{w}^{(j)}$ as regular solutions.*

Proof. Apply Proposition 3.2.6 to determine the coefficients of the polynomials f_i such that $f_i(\mathbf{w}^{(j)}) = 0$. Hereby, $l_i - l + 1$ coefficients can be chosen at random in $\mathbb{C}_0$. Except for a choice of these coefficients belonging to a set of measure zero and due to the independent dense representation of F, we have that $\exists k$: $\deg(\gcd(f_i, f_j), \{x_k\}) = 0$, $\forall j \neq i$, with $\deg(f_i, \{x_k\}) \neq 0 \neq \deg(f_j, \{x_k\})$. So we can eliminate the unknowns one after the other without encountering a common factor among all polynomials at the end of the elimination.

All the l solutions $\mathbf{w}^{(j)}$ are regular solutions if $\det(J_F(\mathbf{w}^{(j)})) \neq 0$, for all $j = 1, 2, \ldots, l$. The exceptional case is represented by the equations $\det(J_F(\mathbf{w}^{(j)})) = 0$, which are equations in the coefficients of the system F. These coefficients are either chosen at random or are solutions of linear systems, whose coefficients are determined by the choice of the interpolation vectors. So each $\det(J_F(\mathbf{w}^{(j)})) = 0$ also determines a set of measure zero in the space of the interpolation vectors. □

Example 3.2.10 (Example 3.2.5 continued.) We start by considering the independent dense representation F^*, given in (3.7). There are two independent roots. Without loss of an independent root, the coefficient of x_1 in the second equation can be considered as zero. We choose two vectors $\mathbf{w}^{(1)} = (-1, 1)$ and $\mathbf{w}^{(2)} = (2, -1)$. By interpolation, we determine the coefficients of the polynomials in the tuple $F^{(0)}$ so that the chosen vectors are solutions of the system $F^{(0)}(\mathbf{x}) = \mathbf{0}$.

$$F^{(0)}(\mathbf{x}) = \left\{ \begin{array}{c} c_{12}x_1^2 + c_{11}x_1 + c_{10} = 0 \\ c_{22}x_1x_2 + c_{21}x_2 + c_{20} = 0 \end{array} \right. \tag{3.9}$$

The initial terms of the system differ from each other, so it can cause no problem to give c_{12} and c_{22} the value 1. To determine the other coefficients, we demand that the chosen vectors satisfy each equation. For each vector, we have to solve a linear system.

1. For the first equation we obtain the system:

$$\left\{ \begin{array}{c} 1 + c_{11}(-1) + c_{10} = 0 \\ 4 + c_{11}2 + c_{10} = 0 \end{array} \right. . \tag{3.10}$$

   Hence it follows that $c_{10} = -2$ and $c_{11} = -1$.

2. Analogously, we find for the second equation that $c_{20} = \frac{3}{2}$ and $c_{21} = -\frac{1}{2}$.

By Theorem 3.2.9, the two solutions of $F^{(0)}(\mathbf{x}) = \mathbf{0}$ are regular. Hence $F^{(0)}$ yields a good homotopy for solving the original system defined by F. ■

As a last comment on interpolating homotopies, we indicate how to deal with scaling equations, such as $x_1^2 + x_2^2 - 1 = 0$, which frequently occur in many applications. By Theorem 3.2.9, a system F with $x_1^2 + x_2^2 - 1 = 0$ as one of its equations can have at most five independent roots, because we can consider the dense representation $x_1^2 + x_1x_2 + x_2^2 + x_1 + x_2 + 1$ for the polynomial used in the scaling equation. Suppose that all other polynomials of F have more than five monomials. In that case, we can construct a start system with more than five roots by generating the interpolating vectors as follows. For every random value of $x_1 = c_1$, $c_1 \in \mathbb{C}_0$, solve $x_2^2 = 1 - c_1^2$ to determine the value of x_2. This observation considerably enhances the applicability of interpolating homotopies. Note that for linear scaling equations, we should be aware of the additional linear dependencies in the interpolation matrices.

3.3 Linear-product start systems

In [167], the strong connection between computing the Bézout number B and the construction of the start system, at that time denoted by Q, was pointed out. This yielded the so-called BQ-Algorithm, independently discovered from the construction of multi-homogeneous start systems mentioned in [177] by Wampler, Morgan and Sommese. Our approach started the computation of B directly from the degree matrix, rather than by using the degree polynomial. The key idea is that the algorithm for computing B carries a strong resemblance with the algorithm for solving a linear-product m-homogeneous start system.

3.3.1 The permanent of a matrix

To compute B efficiently, it is necessary to avoid computing the whole polynomial f_{M_D}. Therefore we introduce permanents, see [22], [92] and [153]. We refer to [101] for a thorough treatise on permanents. For our type of applications we need to define a special generalization of permanents for nonsquare matrices.

Definition 3.3.1 Let $A \in \mathbb{N}^{n \times n}$, *the permanent of* A is defined by

$$\operatorname{per}(A) = \sum_{\mathbf{p}} \prod_{i=1}^{n} a_{p_i i}, \tag{3.11}$$

where $\mathbf{p} \in \mathbb{N}^n$ ranges over all permutations of $(1, 2, \ldots, n)$. Consider $\mathbf{k} \in \mathbb{N}^m$, with $n = \sum_{j=1}^{m} k_j$ and let $A \in \mathbb{N}^{n \times m}$. Then

$$\operatorname{per}(A, \mathbf{k}) = \sum_{\mathbf{p}} \prod_{j=1}^{m} \prod_{l=1}^{k_j} a_{p_{\sigma(j,l)} j}, \quad \sigma(j, l) = \sum_{i=1}^{j-1} k_i + l, \tag{3.12}$$

where $\mathbf{p}$ ranges over all $\mathbf{p} \in \mathbb{N}^{k_1} \times \mathbb{N}^{k_2} \times \cdots \times \mathbb{N}^{k_m}$ satisfying $p_i \neq p_j$, for each $i \neq j$, and where each slice of $\mathbf{p}$: $(p_{i1}, p_{i2}, \ldots, p_{ik_i}) \in \mathbb{N}^{k_i}$ represents a different choice of k_i elements of $\{1, 2, \ldots, n\}$, for $i = 1, 2, \ldots, n$. If $m = n$, then $\operatorname{per}(A, \mathbf{k}) = \operatorname{per}(A)$.

Note that the expansion formula for a permanent of a matrix is the same as for a determinant except that all terms in the sum (3.11) have positive sign, i.e., there is a permanence of signs.

Example 3.3.2 (Example 3.1.11 continued.) Consider the degree matrix M_D in (3.5). Its permanent $\mathrm{per}(M_D) = 2 \times 1 + 1 \times 0 = 2$. In this simple case $\mathbf{k} = (1,1)$ and $\mathrm{per}(M_D, \mathbf{k}) = \mathrm{per}(M_D)$. ∎

Example 3.3.3 To illustrate the generalized permanent, consider the following polynomial system $F(\mathbf{x}) = \mathbf{0}$ (taken from [167]) and the degree matrix M_D, associated with the 2-partition $Z = \{\{x_1\}, \{x_2, x_3\}\}$:

$$F(\mathbf{x}) = \begin{cases} x_1x_2 + x_3 + 1 = 0 \\ x_1x_3 + x_2 + 2 = 0 \\ x_2x_3 + x_1 + 3 = 0 \end{cases} \quad \text{and} \quad M_D = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}. \tag{3.13}$$

We have $\mathbf{k} = (1,2)$ and we compute the permanent as follows

$$\begin{array}{rcccccccccccccccccc} \mathrm{per}(M_D,\mathbf{k}) & = & 1 & \times & 1 & \times & 2 & + & 1 & \times & 1 & \times & 2 & + & 1 & \times & 1 & \times & 1 & = & 5. \\ \text{row} & & 1 & & 2 & & 3 & & 2 & & 1 & & 3 & & 3 & & 1 & & 2 & & \\ \text{column} & & 1 & & 2 & & 2 & & 1 & & 2 & & 2 & & 1 & & 2 & & 2 & & \end{array} \tag{3.14}$$

The labels underneath formula (3.14) indicate the rows and columns from the matrix M_D. ∎

Proposition 3.3.4 *Suppose $M_D \in \mathbb{N}^{n \times m}$ is the degree matrix of a polynomial system F for a given m-partition, with cardinalities $\mathbf{k} \in \mathbb{N}^m$, $n = \sum_{j=1}^m k_j$. Then the m-homogeneous Bézout number B equals* $\mathrm{per}(M_D, \mathbf{k})$.

Proof. B is defined as the coefficient of $x_1^{k_1} x_2^{k_2} \cdots x_m^{k_m}$ of the polynomial

$$\begin{aligned} f_{M_D}(\mathbf{x}) \;=\; & (d_{11}x_1 + d_{12}x_2 + \cdots + d_{1m}x_m)(d_{21}x_1 + d_{22}x_2 + \cdots + d_{2m}x_m) \\ & \cdots (d_{n1}x_1 + d_{n2}x_2 + \cdots + d_{nm}x_m), \end{aligned} \tag{3.15}$$

with d_{ij} the (i,j)th entry of M_D. To compute the requested coefficient, we need to take each x_j exactly k_j times, multiplying yields then the factor $\prod_{l=1}^{k_j} d_{j_l j} x_j^{k_j}$, where for each l, there is a different index j_l. In order to obtain the contribution of the coefficient to $x_j^{k_j}$, it suffices to enumerate all possible different choices of k_j elements of $\{1, 2, \ldots, n\}$ and to make the corresponding products. The enumeration runs over all $j = 1, 2, \ldots, m$, and by the product structure of f_{M_D}, only permutations arise. □

A straightforward way to compute B is to enumerate all possible combinations of choices of k_j elements of $\{1, 2, \ldots, n\}$, as done in Algorithm 3.3.5. We proposed this algorithm in [167]. It was later called 'the basic algorithm' by Wampler in [175] and subsequently refined.

Algorithm 3.3.5 Basic algorithm for computing permanents.

Input: $A \in \mathbb{N}^{n\times m}$, $\mathbf{k} \in \mathbb{N}^m$, $n = \sum_{j=1}^m k_j$. *matrix with cardinalities*
Output: $\mathrm{per}(A, \mathbf{k})$. *permanent of A*

$X := \{1, 2, \ldots, n\}$; *set of unknowns*
$\mathrm{per}(A, \mathbf{k}) := \mathrm{per}(A, \mathbf{k}, X, 1, 1, 0)$. *call for backtracking*

with $\Sigma_j := \mathrm{per}(A, \mathbf{k}, X^{(j)}, j, \Pi_{j-1}, \Sigma_{j-1})$: *auxiliary function*

Input: $A \in \mathbb{N}^{n\times m}$, $\mathbf{k} \in \mathbb{N}^m$, $n = \sum_{j=1}^m k_j$, *matrix with cardinalities*
$X^{(j)} \subset X$, $j \in \{1, 2, \ldots, m\}$, *set to choose out*
Π_{j-1}, Σ_{j-1}. *partial results*
Output: Σ_j. *result of level j*

$\Sigma_j := \Sigma_{j-1}$; *initialization*
for all $E^{(j)} \subseteq X^{(j)}$: $\#E^{(j)} = k_j$ loop *enumerate all choices*
$\Pi_j := \Pi_{j-1} \times \prod_{e\in E^{(j)}} a_{ej}$; *update product*
if $j = m$
then $\Sigma_j := \Sigma_j + \Pi_j$; *update sum*
else $X^{(j+1)} := X^{(j)} \setminus E^{(j)}$; *reduce X*
$\Sigma_j := \Sigma_j + \mathrm{per}(A, \mathbf{k}, X^{(j+1)}, j+1, \Pi_j, \Sigma_j)$; *go to next level*
end if;
end loop.

Although Algorithm 3.3.5 reflects the mathematical definition of $\mathrm{per}(A, \mathbf{k})$ and requires more arithmetical operations than strictly necessary, it yields a significant reduction in the amount of computational work for computing B, compared to its definition based on the degree polynomial.

3.3.2 Multi-homogeneous product start systems

Definition 3.3.6 Consider an m-partition Z and a degree matrix $M_D \in \mathbb{N}^{n\times m}$. Denote the jth set in Z by $Z_j = \{z_{j1}, z_{j2}, \ldots, z_{jk_j}\}$ and denote by d_{ij} the (i,j)th entry in M_D. Then *the polynomial matrix* $M_{\mathbf{x}} \in \mathbb{C}[\mathbf{x}]^{n\times m}$ *associated with* Z *and* M_D is defined as follows. If $d_{ij} = 0$, then the (i,j)th entry of $M_{\mathbf{x}}$ is given by $f_{ij} := 1$, otherwise

$$f_{ij}(\mathbf{z}) := \prod_{l_1=1}^{d_{ij}} \left(\sum_{l_2=1}^{k_j} c_{ijl_1l_2} z_{jl_2} + c_{ijl_1} \right), \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, m, \tag{3.16}$$

with the coefficients $c_{ijl_1l_2}$ and c_{ijl_1} chosen at random in $\mathbb{C}_0$.

Definition 3.3.7 Let there be an m-homogeneous structure yielding the polynomial matrix $M_{\mathbf{x}} \in \mathbb{C}[\mathbf{x}]^{n\times m}$, then the ith equation of a *linear-product start system* $F^{(0)}$ *associated with an* m*-homogeneous structure* is defined by

$$f_i^{(0)}(\mathbf{x}) := \prod_{j=1}^{m} f_{ij}(\mathbf{x}), \quad \text{with} \quad f_{ij} = (M_{\mathbf{x}})_{ij}, \quad i = 1, 2, \ldots, n. \tag{3.17}$$

Example 3.3.8 (Example 3.1.11 continued.) Consider the 2-homogeneous structure for the polynomial system in (3.3). Then the polynomial matrix $M_{\mathbf{x}}$ and associated linear-product start system $F^{(0)}$ are given by

$$\left[\begin{array}{cc} (x_1 - c_{11})(x_1 - c_{21}) & 1 \\ x_1 - c_{21} & x_2 - c_{22} \end{array} \right] \quad \text{and} \quad \left\{ \begin{array}{l} (x_1 - c_{11})(x_1 - c_{12}) = 0 \\ (x_1 - c_{21})(x_2 - c_{22}) = 0 \end{array} \right. . \tag{3.18}$$

For random choices of the coefficients in the polynomial matrix, in particular when $c_{11} \neq c_{21}$, the resulting linear-product system will have exactly two regular solutions, i.e.: (c_{11}, c_{22}) and (c_{12}, c_{22}). ■

It is important to note that the algorithm for solving the product start system follows exactly the same lines as the algorithm for computing the corresponding m-homogeneous Bézout number. Actually, for efficient solving of the product system, we must avoid to consider the linear systems that correspond to factors in the sum of $\mathrm{per}(M_D, \mathbf{k})$ that do not yield a contribution to the Bézout number.

Proposition 3.3.9 *Consider a linear-product start system $F^{(0)}(\mathbf{x}) = \mathbf{0}$ associated with an m-homogeneous structure. Except for a choice of coefficients belonging to a set of measure zero, $F^{(0)}(\mathbf{x}) = \mathbf{0}$ has exactly as many regular solutions in $\mathbb{C}^n$ as B, the m-homogeneous Bézout number.*

Proof. Let N denote the number of coefficients that can be chosen freely in $M_{\mathbf{x}}$. Solve $F^{(0)}(\mathbf{x}) = \mathbf{0}$ by enumerating all possible linear systems defined by $F^{(0)}(\mathbf{x}) = \mathbf{0}$. As $F^{(0)}$ entirely consists of products of linear polynomials, there are exactly as many linear systems as its total degree $D_{F^{(0)}}$. Denote by $A^{(l)}\mathbf{x} = \mathbf{b}^{(l)}$ the lth linear system, for $l = 1, 2, \ldots, D_{F^{(0)}}$. For each separate linear system, there are two cases to consider.

1. $A^{(l)}\mathbf{x} = \mathbf{b}^{(l)}$ corresponds to a permutation $\mathbf{p}$ in the sum of the permanent.

   For every set Z_j of Z, there are exactly as many equations chosen as the corresponding cardinality k_j, for $j = 1, 2, \ldots, m$. So $A^{(l)}\mathbf{x} = \mathbf{b}^{(l)}$ is a well-determined linear system and a condition for it not to have a regular solution is given by $\det(A^{(l)}) = 0$. The equation $\det(A^{(l)}) = 0$ is a polynomial equation in N_l coefficients, $N_l < N$. Its solution set determines a space of dimension $N_l - 1$, which is a set of measure zero in $\mathbb{C}^N$.

2. $A^{(l)}\mathbf{x} = \mathbf{b}^{(l)}$ does not correspond to a permutation in the sum of the permanent.

   So there exists a set of Z, say Z_j, for which more than k_j, say k, equations have been chosen. Hence the corresponding linear system is overdetermined in the unknowns of Z_j. Denote by $A_E^{(l)}\mathbf{x} = \mathbf{b}_E$ the system of linear equations in the unknowns of Z_j. Denote by $n_k = \binom{k}{k_j}$ and by $A_{E_i}^{(l)}\mathbf{x} = \mathbf{b}_{E_i}$ the ith choice of k_j equations of $A_E^{(l)}\mathbf{x} = \mathbf{b}_E$. The system can only have a solution when for every $i = 1, 2, \ldots, n_k$, $\det(A_{E_i}^{(l)}|\mathbf{b}_{E_i}) = 0$. These conditions yield n_k polynomial equations. Each corresponding solution set determines a space of dimension $N_l - 1$, $N_l < N$. The intersection of sets of measure zero is also a set of measure zero.

The case that two linear systems yield the same solution corresponds again with a choice of the coefficients belonging to a set of measure zero of $\mathbb{C}^N$. The union of sets of measure zero in $\mathbb{C}^N$ is still a set of measure zero. So, except for a set of measure zero, the number of solutions of $F^{(0)}(\mathbf{x}) = \mathbf{0}$ equals the permanent and hence equals B. □

Note that we can choose special coefficients, yielding special structured matrices, that can be solved in time $O(n^2)$ instead of time $O(n^3)$. In [167], we proposed to use Vandermonde matrices.

Sparse multi-homogeneous linear-product systems have been proposed by Wampler in [176]. These sparse start systems can be evaluated more efficiently than the general linear-product systems and yield a better performance of the homotopy continuation method. Following the notations of Definition 3.3.6, we can define sparse polynomial matrices.

Definition 3.3.10 Given an m-homogeneous structure by Z and M_D, *the sparse polynomial matrix* $M_{\mathbf{x}} \in \mathbb{C}[\mathbf{x}]^{n\times m}$ *associated with* Z *and* M_D is defined as follows. If $d_{ij} = 0$, then the (i,j)th entry of $M_{\mathbf{x}}$ is given by $f_{ij} := 1$, otherwise

$$f_{ij}(\mathbf{z}) := \left(\sum_{l=1}^{k_j} c_{ijl} z_{jl}\right)^{d_{ij}} - c_{ij0}, \quad i = 1,2,\ldots,n, \quad j = 1,2,\ldots,m, \tag{3.19}$$

with the coefficients c_{ijl} chosen at random in $\mathbb{C}_0$.

The definition of *sparse linear-product systems* is then completely analogous to Definition 3.3.7.

3.4 Minimal Bézout numbers

First we consider a more efficient algorithm for computing permanents, that is more suitable for implementation than the basic algorithm, presented as Algorithm 3.3.5. Then the exhaustive enumeration of all partitions for computing the minimal Bézout number is described. Finally we give a heuristic method for finding an optimal partition.

3.4.1 Efficient permanent computation

Despite the similarity with the determinant, computing the permanent is far more difficult, as proven by Valiant in [153].

Theorem 3.4.1 *The complexity of computing the permanent is #P-hard, even for* $(0,1)$-*matrices.*

This looks very discouraging, but consider the following algorithm for computing the permanent of a matrix, suggested by the proof of Proposition 3.3.9. By constructing $M_{\mathbf{x}}$ and the associated linear-product system $F^{(0)}$, we have to factorize exactly $D_{F^{(0)}}$ matrices, at a total cost of $O(D_{F^{(0)}} n^3)$ arithmetical operations. This method gives a rough bound on the amount of computational work, related to the cost of the homotopy continuation method for solving the given system F, where on the average about hundred linear systems need to be solved along a solution path. However, note that we may have $D_{F^{(0)}} \gg D_F \gg B$.

For $m = n$, the cost of Algorithm 3.3.5 is $O(n!)$, with $n!$ the number of permutations of $(1, 2, \dots, n)$. More precisely, in [175], Wampler analyzed the cost of this basic algorithm. Proposition 3.4.2 summarizes the result of this cost analysis.

Proposition 3.4.2 *The basic algorithm for computing the permanent* $\mathrm{per}(A, \mathbf{k})$ *of* $A \in \mathbb{N}^{n \times m}$*, with* $\mathbf{k} \in \mathbb{N}^m$ *and* $n = \sum_{j=1}^m k_j$*, requires* $N(n-1)$ *multiplications and* $N-1$ *additions, where* $N = \displaystyle\prod_{j=1}^{m-1} \binom{n_j}{k_j}$*, with* $n_1 = n$ *and* $n_j = n_{j-1} - k_{j-1}$*, for* $j = 2, \dots, m$.

Proof. N is the product of the number of combinations that have to be made. Each product in the sum of the permanent requires $n-1$ multiplications. Adding up all the products costs $N-1$ additions. ■

As an introduction to a more efficient algorithm for computing the permanent, consider the following example.

Example 3.4.3 (see [175]) Consider a general degree matrix $M_D \in \mathbb{N}^{4\times 2}$, with $\mathbf{k} = (2, 2)$.

$$M_D = \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \\ d_{31} & d_{32} \\ d_{41} & d_{42} \end{bmatrix} \qquad \begin{aligned} \mathrm{per}(M_D, \mathbf{k}) &= d_{11}d_{21}d_{32}d_{42} + d_{11}d_{31}d_{22}d_{42} + d_{11}d_{41}d_{22}d_{32} \\ &\quad + d_{21}d_{31}d_{12}d_{42} + d_{21}d_{41}d_{12}d_{32} + d_{31}d_{41}d_{12}d_{22} \\ &= d_{11}(d_{21}d_{32}d_{42} + d_{22}(d_{31}d_{42} + d_{32}d_{41})) \\ &\quad + d_{12}(d_{21}(d_{31}d_{42} + d_{32}d_{41}) + d_{22}d_{31}d_{41}). \end{aligned} \tag{3.20}$$

By a reorganization of the order of calculations, only 12 instead of 18 multiplications have to be made. The number of additions remains unchanged. ■

The calculations in the second formula of (3.20) have been organized by expanding the matrix row by row.

The idea of computing the permanent by expansion is analogue to the Laplace expansion theorem for permanents of square matrices. Following [101, Theorem 1.2], we can write

$$\mathrm{per}(A) = \sum_{i=1}^{n} a_{ij}\mathrm{per}(A(i|j)), \quad \text{for any } j,\ 1 \le j \le n, \tag{3.21}$$

where $A(i|j)$ denotes the submatrix obtained from A by deleting row i and column j. Note that in (3.21) we have expanded the jth column of A, but for square matrices, we have that $\mathrm{per}(A^T) = \mathrm{per}(A)$. We refer to [101] for additional properties of permanents.

This idea of expanding the rows is elaborated in Algorithm 3.4.4.

Algorithm 3.4.4 Row expansion algorithm for computing permanents.

```
Input:   A ∈ ℕ^{n×m}, k ∈ ℕ^m, n = Σ_{j=1}^m k_j.            matrix with cardinalities
Output: per(A, k).                                            permanent of A

per(A, k) := per(A, k, 1).                                    call for backtracking

with Σ_i := per(A, k^(i), i):                                 auxiliary function

   Input:   A ∈ ℕ^{n×m}, k^(i) ∈ ℕ^m, n = Σ_{j=1}^m k_j^(i),   matrix with cardinalities
            i ∈ {1, 2, ..., n}.                               index of current row
   Output: Σ_i.                                               result of level i

   Σ_i := 0;  k^(i+1) := k^(i);                               initialization
   for all j in {1, 2, ..., m} loop
     if k_j^(i) ≠ 0 and a_ij ≠ 0
      then k_j^(i+1) := k_j^(i) − 1;                          choose a_ij
           if i < n                                           update the sum
            then Σ_i := Σ_i + a_ij × per(A, k^(i+1), i + 1);  go to next level
            else  Σ_i := Σ_i + a_ij;
           end if;
     end if;
   end loop.
```

Note that by the test $a_{ij} \neq 0$, Algorithm 3.4.4 efficiently deals with sparse matrices. This algorithm is more suitable for computer implementations than the basic algorithm, given in Algorithm 3.3.5, also because the number of multiplications can be reduced by a factor that is $O(n)$. We refer to [175] for a more detailed cost analysis.

In [175], an additional improvement to Algorithm 3.4.4 has been proposed, which led to the algorithm called 'row expansion with memory'. We briefly mention the idea. Consider again the second formula of (3.20), explained in Example 3.4.3. Observe that the factor $d_{31}d_{42}+d_{32}d_{41}$ occurs twice in the formula. We can save computational work by avoiding the recalculation of this factor. So, by storing the intermediate results of the calls $\mathrm{per}(A, \mathbf{k}^{(i+1)}, i+1)$ made in Algorithm 3.4.4, the permanent of A can be computed more efficiently. Because all intermediate results are uniquely characterized by the vectors $\mathbf{k}^{(j)}$, for $j = 2, 3, \ldots, m$, it suffices to take a matrix of dimension $(k_2+1)\times(k_3+1)\times\cdots\times(k_m+1)$ to store all intermediate results.

3.4.2 Exhaustive enumeration of the partitions

To achieve a good performance of homotopy continuation for computing all isolated solutions, we are interested in using a start system based on the best root count available, i.e., in *the minimal Bézout number* B, according to an *optimal partition* Z of the set of unknowns X. Because the number of solution paths is determined by B, it might be worthwhile to search for an optimal partition Z. Note that the minimal Bézout number B can be generated by different partitions, which are in that case all optimal.

In [158], we proposed to enumerate all partitions generated by the following recursive relations. Denote by 2^X the set of all subsets of the set X. In (3.22), the recursive relation to generate 2^X is displayed.

$$2^X = 2^{\{x\} \cup (X \setminus \{x\})} = \left\{ \{x\} \cup 2^{X \setminus \{x\}}, 2^{X \setminus \{x\}} \right\}, \quad \text{with} \quad x \in X. \tag{3.22}$$

Denote by $P(X)$ the set of all partitions of X. The recursive relation to generate $P(X)$ is given in (3.23).

$$P(X) = P(\{x\} \cup X) = \bigcup_{\substack{X_1 \in 2^{X \setminus \{x\}} \\ X_2 \in P(X \setminus \{x\} \setminus X_1)}} \left\{ \{\{x\} \cup X_1\} \cup X_2 \right\}, \quad \text{with} \quad x \in X. \tag{3.23}$$

The algorithm to generate $P(X)$ is based on the nesting of two recursive relations (3.22) and (3.23). Example 3.4.5 illustrates how to generate $P(X)$.

Example 3.4.5 Consider $X = \{x_1, x_2, x_3\}$ and take $x = x_1$. Table 3.1 contains in its rows the different objects generated during the generation of $P(X)$, along the lines of the formulas (3.22) and (3.23). ∎

X_1	$\{\{x\} \cup X_1\}$	$X \setminus \{x\} \setminus X_1$	X_2	$\{\{\{x\} \cup X_1\} \cup X_2\}$
$\emptyset$	$\{\{x_1\}\}$	$\{x_2, x_3\}$	$\{\{x_2\}, \{x_3\}\}$	$\{\{x_1\}, \{x_2\}, \{x_3\}\}$
			$\{\{x_2, x_3\}\}$	$\{\{x_1\}, \{x_2, x_3\}\}$
$\{x_2\}$	$\{\{x_1, x_2\}\}$	$\{x_3\}$	$\{\{x_3\}\}$	$\{\{x_1, x_2\}, \{x_3\}\}$
$\{x_3\}$	$\{\{x_1, x_3\}\}$	$\{x_2\}$	$\{\{x_2\}\}$	$\{\{x_1, x_3\}, \{x_2\}\}$
$\{x_2, x_3\}$	$\{\{x_1, x_2, x_3\}\}$	$\emptyset$	$\emptyset$	$\{\{x_1, x_2, x_3\}\}$

Table 3.1: The generation of all partitions of $\{x_1, x_2, x_3\}$.

In Table 3.2, the rapid growth of the number of all partitions p_n is illustrated by comparing it to the number of all subsets 2^n, of a set of n elements. The number p_n is the sum of the Stirling numbers $S(r, n)$ of the second type, where $r = 1, 2, \ldots, n$, see [92]. $S(r, n)$ represents the number of ways to place r distinct objects in n cells.

n	1	2	3	4	5	6	7	8	9	10
2^n	2	4	8	16	32	64	128	256	512	1,024
p_n	1	2	5	15	52	203	877	4,140	21,147	115,975

Table 3.2: Growth of the number of partitions p_n, for increasing n.

We can only recommend an exhaustive enumeration of all partitions for small-dimensional systems. Because we are only interested in the minimal Bézout number, work can be saved by modifying the algorithm for computing the permanent by stopping the calculations as soon as the partial result result exceeds the best computed Bézout number so far. For high-dimensional systems, we can only rely on a heuristic method for creating an optimal partition.

3.4.3 A heuristic partitioner

The question is to find out whether two variables x_i and x_j are better separated or yield a lower Bézout number when taken together in Z.

For a given polynomial system F, the heuristic partitioner constructs a sequence $Z^{(i)} = \{Z_1^{(i)}, Z_2^{(i)}, \ldots, Z_{m_i}^{(i)}\}$ of n partitions of X, with $Z_{m_i}^{(i)} = \{x_{i+1}, \ldots, x_n\}$, for $i = 1, 2, \ldots, n$, and $Z_{m_n}^{(n)} = \emptyset$. We start with $i = 0$: $Z^{(0)} = \{X\}$. The partitions in the sequence are related as follows. Either $m_{i+1} = m_i + 1$ and then $Z_{m_i}^{(i+1)} = \{x_{i+1}\}$, with $Z_j^{(i+1)} = Z_j^{(i)}$, $\forall j = 1, 2, \ldots, m_i - 1$, or $m_{i+1} = m_i$ and then $\exists l$: $Z_l^{(i+1)} = Z_l^{(i)} \cup \{x_{i+1}\}$, with $\forall j < m_i$, $j \neq l$: $Z_j^{(i+1)} = Z_j^{(i)}$. The vector $\mathbf{k}^{(i)} \in \mathbb{N}^{m_i}$ contains the cardinalities of the sets in $Z^{(i)}$.

We construct this sequence of partitions by computing permanents, i.e., the $(i+1)$th partition $Z^{(i+1)}$ is determined so that $\mathrm{per}(\deg(F, Z^{(i+1)}), \mathbf{k}^{(i+1)})$ is minimal, where we consider all possibilities for $Z^{(i+1)}$, satisfying the constraints of the sequence. At the ith step, x_{i+1} is removed from the last set of $Z^{(i)}$ and considered w.r.t. the other sets in $Z^{(i)}$. To ignore the influence of the unknowns in $Z_{m_i}^{(i)}$, we can take all entries of the last column in the degree matrix $\deg(F, Z^{(i)})$ to be equal to one. We continue until all unknowns have been considered in this way.

The algorithm described above requires at each step about m_i permanent computations. The total number of permanent computations is bounded by n^2. For many applications, an optimal partition can be found in this way. However, the result depends on the order of the unknowns and the algorithm considers only a very limited number of partitions.

3.5 Applications

We have applied the methods described above to various examples taken from the literature. First we give a description of the polynomial systems and summarize thereafter the performance of the methods.

3.5.1 The polynomial systems

The polynomial systems listed below have been taken from the literature, see [28], [43], [105], [106] and [145], to illustrate various aspects of multi-homogenization. The first two applications allow the construction of an interpolating homotopy that has exactly as many known solutions as the Bézout number, while for the other applications, only the linear-product start system reaches this goal. Note that the heuristic partitioner finds an optimal partition for all the examples.

An easy example for elimination methods

In [145], the following system was given as an easy example for resultant based methods and as very difficult to solve by homotopy continuation.

$$F(\mathbf{x}) = \left\{ \begin{array}{rcrcrcrcl} 5x_1x_2 & + & 5x_1 & + & 3x_2 & + & 55 & = & 0 \\ 7x_2x_3 & + & 9x_2 & + & 9x_3 & + & 19 & = & 0 \\ 3x_3x_4 & + & 6x_3 & + & 5x_4 & + & 4 & = & 0 \\ 6x_4x_5 & + & 6x_4 & + & 7x_5 & + & 118 & = & 0 \\ x_5x_6 & + & 3x_5 & + & 9x_6 & + & 27 & = & 0 \\ 6x_6x_7 & + & 7x_6 & + & x_7 & + & 72 & = & 0 \\ 9x_7x_8 & + & 7x_7 & + & x_8 & + & 35 & = & 0 \\ 4x_8x_9 & + & 4x_8 & + & 6x_9 & + & 16 & = & 0 \\ 8x_9x_{10} & + & 4x_9 & + & 3x_{10} & + & 51 & = & 0 \\ 3x_{10}x_1 & + & x_{10} & + & 6x_1 & + & 5 & = & 0 \end{array} \right. \tag{3.24}$$

The total degree of this system equals 1024. The Bézout number based on the 10-homogeneous structure of F equals two. That is exactly the number of finite solutions of F. Using the start system based on the total degree, this system is very hard to solve. By multi-homogenization, we can again solve this system efficiently by continuation methods.

This example also illustrates the importance of root counts for the construction of interpolating homotopies. Without taking the Bézout number B into account (see Theorem 3.2.9), we would count three independent roots, instead of two.

A generalized eigenvalue problem

The next problem is a 5-dimensional generalized eigenvalue problem, given in [28].

$$F(\mathbf{x}, \lambda) = \left\{ \begin{array}{l} (A_2\lambda^2 + A_1\lambda + A_0)\mathbf{x} = \mathbf{0} \\ x_1 + x_2 + \cdots + x_5 - 1 = 0 \end{array} \right. \tag{3.25}$$

We refer to [28] for the specific values of the matrices A_i, $i = 0, 1, 2$. The total degree of this system is 243. By separating the eigenvalue λ from the components of the eigenvector $\mathbf{x}$, the 2-homogeneous Bézout number equals ten and counts exactly the number of finite solutions. For general matrix polynomials of dimension n and degree k, the 2-homogeneous Bézout number based on $Z = \{\{\lambda\}, \{x_1, x_2, \ldots, x_n\}\}$ equals kn, which yields a drastic improvement above the total degree $D = (k+1)^n$.

By generating the interpolation vectors along a random linear scaling function of the components of the eigenvector, we can construct a 2-homogeneous interpolating start system.

Conformal analysis of cyclic molecules

The system, displayed in (3.26), has been described and solved in [43]. It has to be solved to compute all conformations of a cyclic molecule. The system we present here corresponds to the first instance, as given in [43].

$$F(t_1, t_2, t_3) = \left\{ \begin{array}{l} -9 - t_2^2 - t_3^2 - 3t_2^2 t_3^2 + 8t_2 t_3 = 0 \\ -9 - t_3^2 - t_1^2 - 3t_3^2 t_1^2 + 8t_3 t_1 = 0 \\ -9 - t_1^2 - t_2^2 - 3t_1^2 t_2^2 + 8t_1 t_2 = 0 \end{array} \right. \tag{3.26}$$

The total degree of this system equals 64, whereas the exact number of roots is given by the 3-homogeneous Bézout bound $B = 16$. There are eight independent roots, when we would consider an interpolating start system.

Camera motion from point matches

The following system models the displacement of a camera between two positions in a static environment, with the coordinates of the matched points as given in [43]. The coordinates of the frames have been scaled, i.e., all components have been divided by 1000. In [47], this problem has been formulated using epipolar geometry.

$$\left\{ \begin{array}{rcl} -3.6d_1q_1 + 4.1d_1q_2 + 2.0d_1q_3 + 0.1d_1 + 4.1d_2q_1 + 1.8d_2q_2 + 3.7d_2q_3 - 0.2d_2 & & \\ +2.0d_3q_1 + 3.7d_3q_2 - 4.0d_3q_3 + 0.3d_3 + 0.1q_1 - 0.2q_2 + 0.3q_3 + 5.8 & = & 0 \\ -2.140796d_1q_1 - 3.998792d_1q_2 + 3.715992d_1q_3 - 0.2828d_1 - 3.998792d_2q_1 & & \\ -1.575196d_2q_2 - 3.998792d_2q_3 + 3.715992d_3q_1 - 3.998792d_3q_2 - 2.140796d_3q_3 & & \\ +0.2828d_3 - 0.2828q_1 + 0.2828q_3 + 5.856788 & = & 0 \\ 0.3464d_1q_1 + 0.1732d_1q_2 - 5.999648d_1q_3 - 0.1732d_1 + 0.1732d_2q_1 - 5.999648d_2q_2 & & \\ -0.1732d_2q_3 + 0.3464d_2 - 5.999648d_3q_1 - 0.1732d_3q_2 - 0.3464d_3q_3 - 0.1732d_3 & & \\ -0.1732q_1 + 0.3464q_2 - 0.1732q_3 + 5.999648 & = & 0 \\ -5701.3d_1q_1 - 2.9d_1q_2 + 3796.7d_1q_3 - 1902.7d_1 - 2.9d_2q_1 - 5698.7d_2q_2 & & \\ +1897.3d_2q_3 + 3803.3d_2 + 3796.7d_3q_1 + 1897.3d_3q_2 + 5703.1d_3q_3 & & \\ +0.7d_3 - 1902.7q_1 + 3803.3q_2 + 0.7q_3 + 5696.9 & = & 0 \\ -6.8d_1q_1 - 3.2d_1q_2 + 1.3d_1q_3 + 5.1d_1 - 3.2d_2q_1 - 4.8d_2q_2 - 0.7d_2q_3 - 7.1d_2 & & \\ +1.3d_3q_1 - 0.7d_3q_2 + 9.0d_3q_3 - d_3 + 5.1q_1 - 7.1q_2 - q_3 + 2.6 & = & 0 \\ -d_1q_1 - d_2q_2 - d_3q_3 + 1 & = & 0 \end{array} \right. \tag{3.27}$$

The total degree of this system equals 64. By partitioning the set of unknowns as $\{\{d_1, d_2, d_3\}, \{q_1, q_2, q_3\}\}$, the 2-homogeneous Bézout bound 20 is obtained, which equals the exact number of isolated solutions. With an interpolating homotopy, there are ten independent roots.

A PUMA robot

The hand position and orientation of a PUMA robot can be modeled [105] by the following

$$\left\{\begin{array}{r} x_1^2 + x_2^2 - 1 = 0 \\ x_3^2 + x_4^2 - 1 = 0 \\ x_5^2 + x_6^2 - 1 = 0 \\ x_7^2 + x_8^2 - 1 = 0 \\ 0.004731 x_1 x_3 - 0.3578 x_2 x_3 - 0.1238 x_1 - 0.001637 x_2 - 0.9338 x_4 + x_7 - 0.3571 = 0 \\ 0.2238 x_1 x_3 + 0.7623 x_2 x_3 + 0.2638 x_1 - 0.07745 x_2 - 0.6734 x_4 - 0.6022 = 0 \\ x_6 x_8 + 0.3578 x_1 + 0.004731 x_2 = 0 \\ -0.7623 x_1 + 0.2238 x_2 + 0.3461 = 0 \end{array}\right. . \tag{3.28}$$

The total degree of this system equals 128. By partitioning the set of unknowns in $Z = \{\{x_1, x_2\}, \{x_3, x_4, x_7, x_8\}, \{x_5, x_6\}\}$, the 3-homogeneous Bézout bound equals 16. There are exactly 16 finite real regular solutions. Considering the last equation as a linear scaling equation, we could choose at most three independent roots for an interpolating homotopy, because we have to discard the linear monomials x_1 and x_2.

A six-revolute-joint problem in mechanics

The last polynomial system is the inverse position problem for a six-revolute-joint problem in mechanics.

$$F(\mathbf{x}) = \left\{\begin{array}{rcl} x_l^2 + x_{l+1}^2 - 1 & = & 0, \quad l = 1, \ldots, 4 \\ c_{l1} x_1 x_3 + c_{l2} x_1 x_4 + c_{l3} x_2 x_3 + c_{l4} x_2 x_4 & & \\ + c_{l5} x_5 x_7 + c_{l6} x_5 x_8 + c_{l7} x_6 x_7 + c_{l8} x_6 x_8 & & \\ + c_{l9} x_1 + c_{l10} x_2 + c_{l11} x_3 + c_{l12} x_4 + c_{l13} x_5 & & \\ + c_{l14} x_6 + c_{l15} x_7 + c_{l16} x_8 + c_{l17} & = & 0, \quad l = 1, \ldots, 4. \end{array}\right. \tag{3.29}$$

See [152] for a technical paper concerning this problem and [104, Chapter 10], [113] and [177] for more introductory publications. The coefficients for this system can be found in [106]. See [91] for the application of a nonlinear homotopy to solve this problem. The total degree equals 256, the minimal 2-homogeneous Bézout number 96, with corresponding optimal partition $Z = \{\{x_1, x_2, x_5, x_6\}, \{x_3, x_4, x_7, x_8\}\}$. For the choice of the coefficients as in [106], there are 32 finite solutions.

For this application, we cannot construct an m-homogeneous interpolating start system and have to use a linear-product start system. Note that the minimal Bézout number for this system is a too large upper bound for the number of finite solutions.

3.5.2 Performance

Table 3.3 summarizes the performance of multi-homogenization on the applications listed above. The table contains for each system the following characteristics: the dimension n, the total degree D, the minimal Bézout number B, the number l of interpolation vectors that can be generated and $N(F)$, the number of solutions in $\mathbb{C}^n$. Note that for the last system, l has been computed along the lines of Theorem 3.2.9, whereas for the second system, the last equation was considered as a scaling equation for the interpolation vectors.

system	description	n	D	B	l	$N(F)$
(3.24)	easy for elimination	10	1024	2	2	2
(3.25)	generalized eigenvalues	6	243	10	10	10
(3.26)	conformal analysis	3	64	16	8	16
(3.27)	camera motion	6	64	20	10	20
(3.28)	model of PUMA robot	8	128	16	3	16
(3.29)	six-revolute-joint	8	256	96	5	32

Table 3.3: Performance of multi-homogenization.

We see that by casting the system in a multi-projective space, a better root count can be obtained, yielding a start system that captures more of the structure of the system that has to be solved.

3.6 Conclusions and future research

For many applications, counting the roots by multi-homogenization can improve the performance of homotopy continuation methods considerably. There are two different types of multi-homogeneous start systems that can be used. Interpolating start systems have the advantage that the start solutions can be chosen, but are not always applicable for computing all the isolated solutions. This goal can be achieved by the construction of linear-product homotopies.

Although the heuristic partitioner performs well in practice, it would be interesting to look for a faster algorithm for computing an optimal partition that is not based on the plain enumeration of all partitions of the set of unknowns.

The multivariate interpolation problem goes very deep and beyond the topic of multi-homogenization. It is an open question whether we can solve a system more efficiently, by knowing some of its independent roots. Even when the Bézout number yields an exact root count, no complete satisfactory answer is known.

Chapter 4

Product Homotopies

The homotopy methods presented in this chapter lead to general linear-product start systems and are based on generalized versions of Bézout's theorem.

In [85, 86], Li, Sauer and Yorke presented the random product homotopy for dealing with a special class of polynomial systems. The cheater's homotopy has been described in [87], by Li, Sauer and Yorke. Also coefficient-parameter polynomial continuation generalizes Bézout's theorem to systems with parameters, and has been developed by Morgan and Sommese in [111, 112]. The methods listed above are either restricted to a class of polynomial systems, such as the random product homotopy, or require first the use of a homotopy based on Bézout's theorem in multi-projective space, such as cheater's homotopy or a parameter homotopy.

Our generalizations of Bézout's theorem provide general applicable and direct root counting methods. In [168], we derived the GBQ-Algorithm, by taking for different equations different partitions of the sets of unknowns. This method generalizes multi-homogenization towards the random product homotopy. By applying set structures, we presented in [163] a general framework for constructing linear-product start systems. The first part of this chapter describes homotopy methods based on a generalized Bézout number, which yields for many applications a better estimate for the number of roots than the ordinary multi-homogeneous Bézout number. Recently, the way to construct set structures has been refined by Li, Wang and Wang, see [88].

The exploitation of symmetry in applications is very important because it decreases the computational time a lot. In [99], Meravý outlined the construction of a sign-symmetric homotopy and gave conditions upon start systems in the general case. However, Meravý only used the classical theorem of Bézout, which restricts the range of applications. Li and Sauer applied in [84] the random product homotopy to what they called a self-symmetric polynomial system. The second part of this chapter deals with the application of set structures to symmetric polynomial systems, which we described in [164].

The chapter is organized as follows. In the following section, set structure analysis will be presented. The homotopy construction method for symmetric polynomial systems is presented in the second section. The third section contains some applications. In the last section we state our conclusions and indicate some suggestions for future research.

4.1 Set-structure analysis

This section is organized as follows. First, based on the supporting set structure, a new upper bound for the number of finite solutions can be computed. Then, the construction of a start system follows. In the fourth section, the main theorem presented in [163] will be stated. We conclude with the description of a heuristic procedure for generating a supporting set structure and with some efficiency considerations concerning the computation of the new Bézout number.

4.1.1 Set structures

Let X denote $\{x_1, x_2, \ldots, x_n\}$, the set of unknowns of a polynomial system of n equations. With multi-homogenization, we use a partition Z of X to model the structure of a polynomial system F. Instead of a partition, a more refined data structure will be defined to represent the structure of F.

Definition 4.1.1 A *set structure* $\mathfrak{S}$ is defined as $\mathfrak{S} = (\mathfrak{S}_1, \mathfrak{S}_2, \ldots, \mathfrak{S}_n)$, where each $\mathfrak{S}_i = [\mathfrak{S}_{i1}, \mathfrak{S}_{i2}, \ldots, \mathfrak{S}_{im_i}]$ is an array of subsets $\mathfrak{S}_{ij} \subseteq X$, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m_i$.

Definition 4.1.2 Let $f(\mathbf{x}) \in \mathbb{C}[x_1, x_2, \ldots, x_n]$ and let S be an array of subsets of X. Then S is said to be *supporting* for the polynomial f if it satisfies the following conditions:

1. For each term $c_d x_k^d$ of the polynomial f, $c_d \neq 0$, there are at least d sets in S that contain x_k.
2. For each term $c_{\mathbf{d}} x_1^{d_1} x_2^{d_2} \cdots x_n^{d_n}$ of the polynomial f, $c_{\mathbf{d}} \neq 0$, there exist at least d_1 sets in S that contain x_1 such that, if they are removed from S, then the resulting array of subsets $\widetilde{S}$ is supporting for the term $c_{\mathbf{d}} x_2^{d_2} \cdots x_n^{d_n}$.

Note that the array that contains only the empty set is supporting for the constant polynomial $f = 1$. By its definition, the number of sets in a supporting array for a polynomial f is always larger than or equal to its degree $\deg(f)$.

Definition 4.1.3 Consider the tuple $F = (f_1, f_2, \ldots, f_n)$ of multivariate polynomials. A set structure $\mathfrak{S} = (\mathfrak{S}_1, \mathfrak{S}_2, \ldots, \mathfrak{S}_n)$ is said to be *supporting* for F if each array $\mathfrak{S}_k$ is supporting for the respective polynomial f_k, for $k = 1, 2, \ldots, n$. $\mathfrak{S}$ is called a *supporting set structure* for the tuple F.

Example 4.1.4 Consider the following polynomial system:

$$F(\mathbf{x}) = \left\{ \begin{array}{l} x_1^2 x_2 + x_2^2 + x_1 + 1 = 0 \\ x_2^2 x_1 + x_1^2 + x_2 + 1 = 0 \end{array} \right. . \tag{4.1}$$

Two supporting set structures $\mathfrak{S}^{(1)}$ and $\mathfrak{S}^{(2)}$ for the system F are displayed in Table 4.1.

Note that this is a more apparent way of presenting the set structure instead of putting $\mathfrak{S}^{(1)}$ as $([\{x_1\}, \{x_1, x_2\}, \{x_2\}], [\{x_2\}, \{x_2, x_1\}, \{x_1\}])$, as described in the definition of a set structure. The order of the sets in the array does not matter, we only have written them in a symmetric way. ∎

1	$\{x_1\}$	$\{x_1, x_2\}$	$\{x_2\}$
2	$\{x_2\}$	$\{x_2, x_1\}$	$\{x_1\}$

1	$\{x_1\}$	$\{x_1\}$	$\{x_2\}$	$\{x_2\}$
2	$\{x_2\}$	$\{x_2\}$	$\{x_1\}$	$\{x_1\}$

Table 4.1: Supporting set structures $\mathfrak{S}^{(1)}$ and $\mathfrak{S}^{(2)}$ for F.

As with multi-homogenization, where we have a great freedom for choosing a partition Z, there are many ways to choose a supporting set structure $\mathfrak{S}$, but in practice, this choice follows from the structure of the polynomial system. Enumerating all possible set structures is not that straightforward anymore. Therefore we can only rely on a heuristic procedure for constructing a supporting set structure.

4.1.2 The Bézout number based on a set structure

Based on a set structure $\mathfrak{S}$, a new Bézout number $B_{\mathfrak{S}}$ will be defined. First we need a formal way to decide whether a linear system can have a solution, based on the occurrences of the unknowns in its equations. This will be crucial when we consider the construction of a start system, based on a set structure $\mathfrak{S}$.

Definition 4.1.5 Let $\mathfrak{S} = (\mathfrak{S}_1, \mathfrak{S}_2, \ldots, \mathfrak{S}_n)$ be a set structure. An *acceptable tuple of* $\mathfrak{S}$, denoted by $T_{\mathfrak{S}}$, is an n-tuple of subsets of X for which the kth subset belongs to $\mathfrak{S}_k$ and such that any union of k subsets of $T_{\mathfrak{S}}$ contains at least k elements of X.

The following definition characterizes the number $B_{\mathfrak{S}}$.

Definition 4.1.6 Let F be a polynomial system and $\mathfrak{S}$ a supporting set structure for F. Then $B_{\mathfrak{S}}$ is defined as the number of all acceptable tuples of $\mathfrak{S}$.

We will prove that the number $B_{\mathfrak{S}}$ yields an upper bound for the number of finite solutions of F and hence can be considered as a generalized Bézout number.

Example 4.1.7 (Example 4.1.4 continued.) For the system in (4.1), with $\mathfrak{S}^{(1)}$ as supporting set structures $B_{\mathfrak{S}^{(1)}}$ is calculated as

$$\begin{aligned} B_{\mathfrak{S}^{(1)}} &= \underset{\{x_1\}\{x_2\}}{1} + \underset{\{x_1\}\{x_2,x_1\}}{1} + \underset{\{x_1,x_2\}\{x_2\}}{1} + \underset{\{x_1,x_2\}\{x_2,x_1\}}{1} \\ &+ \underset{\{x_1,x_2\}\{x_1\}}{1} + \underset{\{x_2\}\{x_2,x_1\}}{1} + \underset{\{x_2\}\{x_1\}}{1} = 7. \end{aligned} \tag{4.2}$$

The sets underneath the formula (4.2) indicate the sets associated with the acceptable tuples. $B_{\mathfrak{S}^{(2)}}$ can be computed analogously and equals eight. Because of the nature of the set structure $\mathfrak{S}^{(2)}$, $B_{\mathfrak{S}^{(2)}} = B_Z$, the 2-homogeneous Bézout number, with $Z = \{\{x_1\}, \{x_2\}\}$. ∎

4.1.3 Linear-product start systems

Because the cardinalities in the arrays of the set structure can vary greatly, it is not straightforward to associate a polynomial matrix to the set structure. Therefore we first define random linear-product start polynomials.

Definition 4.1.8 Let $S = [S_1, S_2, \ldots, S_m]$ be an array of subsets of X. A *random linear-product start polynomial* $f^{(0)}$ *based on* S is defined as

$$f^{(0)}(\mathbf{x}) = \prod_{j=1}^{m} \left(\sum_{x_l \in S_j} c_{jl} x_l + c_{j0} \right), \tag{4.3}$$

where all c_{jl} and c_{j0}, for $j = 1, 2, \ldots, m$, are randomly chosen numbers in $\mathbb{C}_0$.

Definition 4.1.9 Let $\mathfrak{S} = (\mathfrak{S}_1, \mathfrak{S}_2, \ldots, \mathfrak{S}_n)$ be a set structure. A *random linear-product start system* $F^{(0)}$ *based on* $\mathfrak{S}$ is defined by a tuple of random product start polynomials $f_i^{(0)}$ based on $\mathfrak{S}_i$, for $i = 1, 2, \ldots, n$.

Note that by definition, $\mathfrak{S}$ is a supporting set structure for $F^{(0)}$. In fact, $\mathfrak{S}$ determines the monomial structure of linear-product systems that have exactly $B_{\mathfrak{S}}$ regular solutions in $\mathbb{C}^n$ and that contain all monomials of F, as will be proven next. The proof of Theorem 4.1.10 uses properties of mixed volumes of Newton polytopes, we refer to the next chapter for precise definitions.

Theorem 4.1.10 *Consider a tuple $F^{(0)}$ of random linear-product start polynomials, based on the set structure $\mathfrak{S}$. Then for every random choice of the coefficients of the start polynomials, except for a set of measure zero, the system $F^{(0)}(\mathbf{x}) = \mathbf{0}$ has exactly $B_{\mathfrak{S}}$ regular solutions in $\mathbb{C}^n$. Moreover, $F^{(0)}(\mathbf{x}) = \mathbf{0}$ has exactly as many solutions as any system with the same monomials as $F^{(0)}$ but with randomly chosen coefficients in $\mathbb{C}_0$.*

Proof. In [9] it is proven that the mixed volume of the Newton polytopes of a polynomial system gives an exact root count, provided that the coefficients are sufficiently at random. Let $\mathcal{P} = (P_1, P_2, \ldots, P_n)$ be the n-tuple of Newton polytopes of the polynomials in the n-tuple $F^{(0)}$. The mixed volume $V_n(\mathcal{P})$ will be computed, by exploiting the fact that $F^{(0)}(\mathbf{x}) = \mathbf{0}$ is a linear-product system.

Because each polynomial $f_i^{(0)}$ of $F^{(0)}$ is the product of linear polynomials, each polytope P_i can be written as $P_i = L_{i1} + L_{i2} + \ldots + L_{im_i}$, where $m_i = \#\mathfrak{S}_i$ and where each L_{ij} is the Newton polytope of a linear polynomial. Because of the fact that the mixed volume $V_n(\mathcal{P})$ is multilinear, (see [9], [124, appendix A.4] or Proposition 5.1.8), $V_n(\mathcal{P})$ is the sum of all mixed volumes $V_n(L_{1j_1}, L_{2j_2}, \ldots, L_{nj_n})$, with $1 \leq j_i \leq m_i$, for all $i = 1, 2, \ldots, n$. Denote then the corresponding linear systems by $A^{(J)}\mathbf{x} = \mathbf{b}^{(J)}$, using a multi-index notation, $J = (j_1, j_2, \ldots, j_n)$, where for each linear system M_J nonzero coefficients are involved. Let N denote the total number of nonzero coefficients which can be chosen freely in the start polynomials, then $M_J \leq N$.

For each linear system $A^{(J)}\mathbf{x} = \mathbf{b}^{(J)}$, there are two possibilities:

1. If the linear system corresponds to an acceptable tuple, then the system has one finite solution and $V_n(L_{1j_1}, L_{2j_2}, \ldots, L_{nj_n}) = 1$. Except for the case where $\det(A^{(J)}) = 0$, which can be expressed by a polynomial equation in M_J unknowns determining a space of dimension $M_J - 1$, a set of measure zero in $\mathbb{C}^N$.

2. If the linear system does not correspond to an acceptable tuple, then there is no finite solution and $V_n(L_{1j_1}, L_{2j_2}, \ldots, L_{nj_n}) = 0$. In this case, the conditions of Definition 4.1.5 are not satisfied. This means that there are k sets, whose union contains less than k unknowns. Denote this union by the set S, with $s = \#S$ and let $k = s + r$, with $r \geq 1$. So the linear system contains $s + r$ equations in the unknowns of the set S, which has in general no finite solution. The exceptional case where there is a finite solution corresponds to the case where all possible choices of $s + 1$ equations out of these $s + r$ equations are linearly dependent. Denote the number of all possible choices by c and denote all choices of $s + 1$ equations by $A^{(Jl)}\mathbf{x} = \mathbf{b}^{(Jl)}$, for $l = 1, 2, \ldots, c$. The exceptional case can then be expressed by c polynomial equations, defined by $\det(A^{(Jl)}|\mathbf{b}^{(Jl)}) = 0$, yielding spaces of dimension $M_J - k_l$, with all $k_l \geq 1$, for $l = 1, 2, \ldots, c$. Hence, in order to have a finite solution, the coefficients of these $s + r$ equations must belong to the intersection of these spaces of measure zero, which is again a space of measure zero in $\mathbb{C}^N$.

The finite union of sets of measure zero is also a set of measure zero in $\mathbb{C}^N$. Except for this set of measure zero, there are exactly $B_{\mathfrak{S}}$ linear systems whose matrices are nonsingular. Multiple solutions can only occur when two linear systems are identical, which is again a choice of the coefficients belonging to a set of measure zero. Hence, except for some set of measure zero, $F^{(0)}(\mathbf{x}) = \mathbf{0}$ has exactly $B_{\mathfrak{S}}$ finite regular solutions and $B_{\mathfrak{S}}$ equals the mixed volume $V_n(\mathcal{P})$. Note that, as every right-hand side vector $\mathbf{b}^{(J)} \in \mathbb{C}_0^n$, all solutions of $F^{(0)}(\mathbf{x}) = \mathbf{0}$ lie in $\mathbb{C}_0^n$. □

Example 4.1.11 (Example 4.1.4 continued.) For the system in (4.1), based on the supporting set structure $\mathfrak{S}^{(1)}$, the following linear-product start system $F^{(0)}$ can be constructed:

$$F^{(0)}(\mathbf{x}) = \begin{cases} (x_1 + c_{11})(x_1 + c_{12}x_2 + c_{13})(x_2 + c_{14}) = 0 \\ (x_2 + c_{21})(x_2 + c_{22}x_1 + c_{23})(x_1 + c_{24}) = 0 \end{cases} . \tag{4.4}$$

For every random choice of the coefficients, except for a set of measure zero, the system $F^{(0)}$ will have seven finite regular solutions. With every acceptable tuple in the formula (4.2) of Example 4.1.7 there corresponds a linear system with a nonsingular matrix. For the first acceptable tuple in the formula (4.2) for the computation of $B_{\mathfrak{S}^{(1)}}$, the correspondence is displayed in (4.5).

$$\begin{pmatrix} \{x_1\} \\ \{x_2\} \end{pmatrix} \quad \Leftrightarrow \quad \begin{cases} x_1 + c_{11} = 0 \\ x_2 + c_{21} = 0 \end{cases} \tag{4.5}$$

For random values of c_{11} and c_{21}, the solution of the linear system in (4.5) determines a regular solution of the start system $F^{(0)}$ displayed in (4.4). ■

4.1.4 Homotopy continuation

In this section, we will prove that all solutions of the target system can be obtained, using a linear-product start system based on a set supporting set structure $\mathfrak{S}$ that has exactly as many regular solutions as the corresponding number $B_{\mathfrak{S}}$. Therefore we may consider $B_{\mathfrak{S}}$ as a generalized Bézout number.

Lemma 4.1.12 *Let F be a polynomial system with supporting set structure $\mathfrak{S}$ and $F^{(0)}$ the random product start system based on $\mathfrak{S}$. Define the homotopy H by*

$$H(\mathbf{x}, t) = F^{(0)}(\mathbf{x}) + tF(\mathbf{x}). \qquad (4.6)$$

Then for all t, the system $H(\mathbf{x}, t) = \mathbf{0}$ has no more than $B_{\mathfrak{S}}$ regular solutions in $\mathbb{C}^n$.

Proof. By Theorem 4.1.10, the system $F^{(0)}$ has exactly $B_{\mathfrak{S}}$ regular solutions in $\mathbb{C}^n$. By definition of the random product system $F^{(0)}$, the Newton polytopes of $F^{(0)}$ contain those of F. Therefore, the Newton polytopes remain invariant, for all t. Hence, the mixed volume of the Newton polytopes of all systems $H(\mathbf{x}, t) = \mathbf{0}$ equals $B_{\mathfrak{S}}$. □

Theorem 4.1.13 allows the usage of the random product start system based on a set structure in a homotopy continuation environment. Our proof is inspired on the proof of Morgan and Sommese, see [107]. Note that in [88], Li, Wang and Wang have given another proof of this theorem.

Theorem 4.1.13 *Consider $F(\mathbf{x}) = \mathbf{0}$, a polynomial system with supporting set structure $\mathfrak{S} = (\mathfrak{S}_1, \mathfrak{S}_2, \ldots, \mathfrak{S}_n)$. Let $F^{(0)}(\mathbf{x}) = \mathbf{0}$ be a random product start system based on $\mathfrak{S}$ with exactly $B_{\mathfrak{S}}$ regular solutions in $\mathbb{C}^n$. Then the convex-linear homotopy $H(\mathbf{x}, t) = \mathbf{0}$ is a good homotopy for computing all isolated solutions of $F(\mathbf{x}) = \mathbf{0}$.*

Proof. First a homogenization of the homotopy will be described. To the kth equation of $F(\mathbf{x}) = \mathbf{0}$ and $F^{(0)}(\mathbf{x}) = \mathbf{0}$ corresponds the supporting array of sets $\mathfrak{S}_k$. If x_j occurs in j_l sets of $\mathfrak{S}_k$, then, for the sth occurrence of x_j, x_j will be replaced by x_{js}. As this introduces new unknowns, the following linear equations will be added in order to keep the same solutions:

$$x_{j1} - x_{js} = 0, \quad \text{for } s = 2, 3, \ldots, j_l. \qquad (4.7)$$

By replacing x_{js} in the kth equation by x_{kjs} and adding the following linear equations

$$x_{1j1} - x_{kj1} = 0, \quad \text{for } k = 2, 3, \ldots, n, \quad \text{for } j = 1, 2, \ldots, n, \qquad (4.8)$$

the solutions remain unchanged and all sets belonging to the set structure $\mathfrak{S}$ can be linearized into one partition Z. With respect to this partition Z, both systems have the same multi-homogeneous structure. Denote the classical projective space by $\mathbb{P}^1$. The unknowns belonging to the ith set $\mathfrak{S}_i$ of the partition Z will be embedded in an m_i-dimensional projective space $\mathbb{P}^{m_i}$, where $m_i = \#\mathfrak{S}_i$. The direct product of all projective spaces $\mathbb{P}^{m_i}$ will be denoted by $\mathbb{P}$.

Consider the multi-homogeneous homotopy

$$\widetilde{H} = \mu_0 \widetilde{F}^{(0)}(\mathbf{z}) + \mu_1 \widetilde{F}(\mathbf{z}), \quad (\mu_0, \mu_1) \in \mathbb{P}^1, \qquad (4.9)$$

where $\mathbf{z}$ belongs to the multi-projective space $\mathbb{P}$. Let $\widetilde{Y}$ be the union of the irreducible components of $\widetilde{H}^{-1}(\mathbf{0})$ in $\mathbb{P}$ which contain at least one of the $B_{\mathfrak{S}}$ finite regular solutions of $\widetilde{F}^{(0)}$, $\widetilde{Y}$ is an algebraic set in $\mathbb{P} \times \mathbb{P}^1$. By Theorem 4.1.10, for $(1,0) \in \mathbb{P}^1$, $\widetilde{Y}$ contains exactly $B_{\mathfrak{S}}$ regular solutions in $\mathbb{C}^n$. Denote the natural projection on $\mathbb{P}^1$ by

$$\pi_2 : \mathbb{P} \times \mathbb{P}^1 \rightarrow \mathbb{P}^1. \tag{4.10}$$

Let $U \subset \widetilde{Y}$ be the set of points where singularities occur. By [49, Lemma, p. 97], U is an analytic set, and by Chow's Theorem [61, p. 167], U is an algebraic set. By the Main Theorem of elimination theory [120, p. 33], the projection of U, $\pi_2(U)$ is an algebraic set. $\pi_2(U)$ is a proper subset of $\mathbb{P}^1$, because for $(1,0)$ all solutions are regular. Hence, $\pi_2(U)$ is finite.

Let $V \subset \widetilde{Y}$ be the set of points were solutions at infinity occur. V is an algebraic set and so is its projection $\pi_2(V)$. Because for $(1,0)$ all $B_{\mathfrak{S}}$ solutions are finite, $\pi_2(V)$ is a proper algebraic set in $\mathbb{P}^1$. Hence, $\pi_2(V)$ is finite. Let $W = \pi_2(U) \cup \pi_2(V)$. Because $W \subset \mathbb{P}^1$ is finite, only a finite number of rays $c = re^{i\theta}$ can intersect W. Since then no singularities occur for the interval $[0,1)$, so $\frac{dt}{ds} > 0$ and paths do not turn back. Hence, the smoothness property is proved.

Consider the homotopy H in affine space, with the corresponding set of solution paths Y. Let $\mathbf{z}_0$ be a regular isolated finite solution of $F(\mathbf{x}) = \mathbf{0}$. By the Implicit Function Theorem [120, pp. 10–11], there are unique convergent power series in t to denote the solutions in the neighborhood of $\mathbf{z}_0$. So the solution $\mathbf{z}_0$ can be extended for $t < 1$. Because the solution is finite, for $t < 1$, the extended solution is also finite. By the smoothness property, there exists a path, parameterized by $t \in (0,1)$. By Lemma 4.1.12, the path that ends at $\mathbf{z}_0$, belongs to Y.

Let $\mathbf{z}_0$ be an isolation solution of F with multiplicity m_0. By a slight perturbation of F, for $t < 1$, m_0 isolated regular solutions lie in the neighborhood of $\mathbf{z}_0$. According to previous reasoning, every isolated regular solution is reached by a path starting at a solution of $F^{(0)}$. Hence, for $t \rightarrow 1$, every isolated solution $\mathbf{z}_0$ with multiplicity m_0, has m_0 paths converging to it. □

In the proof, a transformation has been made to a higher- dimensional space. Because of practical considerations, continuation is performed in n-dimensional space. Otherwise, the computational advantage of this approach would be destroyed.

The following can be considered as a generalization of the theorem of Bézout.

Corollary 4.1.14 *Consider a tuple F of polynomials and a supporting set structure $\mathfrak{S}$. If the system $F(\mathbf{x}) = \mathbf{0}$ has a finite number of solutions in $\mathbb{C}^n$, counting multiplicities, then this number cannot be larger than $B_{\mathfrak{S}}$.*

Based on a set structure, a more efficient homotopy can often be constructed, as is illustrated by the applications. Equally important is the fact that set structure analysis can also be used more often than multi-homogenization to model the symmetric structure of a polynomial system.

4.1.5 Generating the set structure

The following illustrates the usefulness of the upper bound $B_{\mathfrak{S}}$ w.r.t. the total degree D of the polynomial system.

Proposition 4.1.15 *Let $F(\mathbf{x}) = \mathbf{0}$ be a polynomial system with supporting set structure $\mathfrak{S} = (\mathfrak{S}_1, \mathfrak{S}_2, \ldots, \mathfrak{S}_n)$. If the number of sets in $\mathfrak{S}_k$, for all $k = 1, 2, \ldots, n$, does not exceed d_k, the degree of the kth polynomial in F, then $B_{\mathfrak{S}} \leq D$, where D is the total degree of F.*

Proof. Based on the set structure $\mathfrak{S}$, a random product start system $F^{(0)}$ can be constructed. $F^{(0)}$ has exactly $B_{\mathfrak{S}}$ regular solutions. While the number of sets in $\mathfrak{S}_k$ does not exceed d_k, $\deg(f_k^{(0)}) \leq d_k$, where $f_k^{(0)}$ is the kth equation of $F^{(0)}$. Hence, $B_{\mathfrak{S}} \leq D$. □

Algorithm 4.1.16 provides a heuristic construction of a supporting array of sets for one polynomial.

Algorithm 4.1.16 Heuristic construction of a supporting array.

Input: $f(\mathbf{x}) = \sum_{k=0}^{l} c_{\mathbf{d}_k} \mathbf{x}^{\mathbf{d}_k}$, $d = \deg(f)$.	*multivariate polynomial*
Output: $S = [S_1, S_2, \ldots, S_d]$.	*supporting array*
$d := \deg(f)$;	*number of sets*
$S := [\emptyset, \emptyset, \ldots, \emptyset]$;	*d empty sets to initialize S*
for k in $0, 1, \ldots, l$ loop	*consider all monomials of f*
if $\nexists i < k$: $d_{ij} \geq d_{kj}$, $\forall j$	*is S already supporting for $\mathbf{x}^{\mathbf{d}_k}$?*
then $m := 1$;	*counter in array*
for i in $1, 2, \ldots, n$ loop	*distribute all unknowns x_i*
for j in $1, 2, \ldots, d_{ki}$ loop	*over as many sets as d_{ki}*
if $x_i \notin S_m$	
then $S_m := S_m \cup \{x_i\}$;	*update mth set*
end if;	
$m := m + 1$;	*update counter*
end loop;	
end loop;	
end if;	
end loop.	

By using the algorithm proposed in Algorithm 4.1.16, a supporting set structure for a polynomial system can be constructed. The application of the algorithm is illustrated in Example 4.1.17.

Example 4.1.17 Consider the following polynomial $f(x_1, x_2) = x_1^2 + x_1 x_2 + 3x_1 - 1$. Because $\deg(f) = 2$, the supporting array will contain two sets. In (4.11), the execution of the main loop of Algorithm 4.1.16 is displayed.

$$
\begin{array}{lllllll}
k=0 & \nexists i<0 & & & & & \\
 & m:=1; \; i:=1; & d_{01}=2; & j:=1; & S_1:=S_1\cup\{x_1\}; & m:=m+1; & \\
 & & & j:=2; & S_2:=S_2\cup\{x_1\}; & m:=m+1; & \\
 & \quad i:=2; & d_{02}=0 & & & & \\
k=1 & \multicolumn{5}{l}{\nexists k<1, \text{ because } d_{12}=1>d_{02}=0} & (4.11)\\
 & m:=1; \; i:=1; & d_{11}=1 & j:=1; & x_1\in S_1; & m:=m+1; & \\
 & \quad i:=2; & d_{12}=1 & j:=1; & S_2:=S_2\cup\{x_2\}; & m:=m+1; & \\
k=2 & \multicolumn{5}{l}{\exists i=0,\; d_{01}=2\geq d_{21}=1,\; d_{02}=0\geq d_{22}=0} & \\
k=3 & \multicolumn{5}{l}{\exists i=0,\; d_{01}=2\geq d_{31}=0,\; d_{02}=0\geq d_{32}=0} &
\end{array}
$$

The resulting supporting array is $T=\{\{x_1\},\{x_1,x_2\}\}$. ■

Although Algorithm 4.1.16 constructs set structures that satisfy the conditions of Proposition 4.1.15, this algorithm is only a heuristic. It can happen that *better* supporting set structures exist, which are not generated by this algorithm. With *better*, we mean a set structure that yields a lower upper bound. Analogously to multi-homogenization, an *optimal set structure* is a set structure for which the corresponding Bézout number is minimal. In that case we can speak of *the minimal Bézout number* $B_{\mathfrak{S}}$.

When using the heuristic algorithm presented in Algorithm 4.1.17 on the polynomials of a tuple F, we consider only one polynomial at a time, without taking the structure of the other polynomials into account. As pointed out by Li, Wang and Wang in [88], this can lead to a Bézout number that is too high, as illustrated by the following example.

Example 4.1.18 Consider a two-dimensional eigenvalue problem, with a linear scaling equation for the components of the eigenvector. In Table 4.2, two supporting set structures $\mathfrak{S}^{(1)}$ and $\mathfrak{S}^{(2)}$ are presented. In the first equation, there is only one nonlinear term, that is λx_1 and there is a choice where to put x_2, either together with λ, or together with x_1. The first choice yields $\mathfrak{S}^{(1)}$, while $\mathfrak{S}^{(2)}$ is obtained by the second alternative, see Table 4.2.

1	$\{\lambda, x_2\}$	$\{x_1\}$
2	$\{\lambda, x_1\}$	$\{x_2\}$
3		$\{x_1, x_2\}$

1	$\{\lambda\}$	$\{x_1, x_2\}$
2	$\{\lambda\}$	$\{x_1, x_2\}$
3		$\{x_1, x_2\}$

Table 4.2: Supporting set structures $\mathfrak{S}^{(1)}$ and $\mathfrak{S}^{(2)}$ for two-dimensional eigenvalue problem.

The first set structure $\mathfrak{S}^{(1)}$ yields the Bézout number $B_{\mathfrak{S}^{(1)}}=3$, whereas $B_{\mathfrak{S}^{(2)}}=2$ renders the exact number of roots. ■

To improve the construction of a supporting set structure, Li, Wang and Wang proposed in [88] to work in two stages. During the first stage, they apply a variant of Algorithm 4.1.16, leaving the choices still open. In the second, the so-called adjusting stage, the choices for grouping the unknowns are made such that the resulting Bézout number is as low as possible. We refer to [88] for a more detailed description of these methods, which can yield a significant improvement over the heuristic procedure we presented in Algorithm 4.1.16.

Another important issue is the efficient computation of the Bézout number $B_{\mathfrak{S}}$. There is a formal and a numeric way. The formal algorithm follows Definition 4.1.6 and enumerates all $\prod_{i=1}^{n} m_i$ candidates for acceptable tuples. According to Definition 4.1.5, to decide whether an n-tuple of sets is an acceptable tuple, we consider the union of all k subsets, generated by enumeration, for all $k = 1, 2, \dots, n$. The numeric algorithm for computing $B_{\mathfrak{S}}$ consists of solving the random linear product system $F^{(0)}(\mathbf{x}) = \mathbf{0}$. The advantage of computing first symbolically all acceptable tuples is that the one-to-one correspondence between the acceptable tuples and the linear systems defined by $F^{(0)}(\mathbf{x}) = \mathbf{0}$ can be exploited to save time when solving the start system.

To compute $B_{\mathfrak{S}}$ more efficiently, we have to relate it to some kind of generalized permanent.

Definition 4.1.19 Consider a set structure $\mathfrak{S} = (\mathfrak{S}_1, \mathfrak{S}_2, \dots, \mathfrak{S}_n)$. The *degree set structure table of* $\mathfrak{S}$ is the couple $(M_{\mathfrak{S}}, S)$ with $M_{\mathfrak{S}} \in \mathbb{N}^{n \times m}$, S is the m-dimensional array of all different sets in $\mathfrak{S}$, denoted by $S = [S_1, S_2, \dots, S_m]$. The (i, j)th entry in $M_{\mathfrak{S}}$ counts the number of times S_j occurs in $\mathfrak{S}_i$.

Definition 4.1.20 Consider an m-dimensional array of sets $S = [S_1, S_2, \dots, S_m]$ and let $A \in \mathbb{N}^{n \times m}$. The *permanent of* A *w.r.t.* S is defined as

$$\mathrm{per}(A, S) = \sum_{\mathbf{p}} \prod_{i=1}^{n} a_{ip_i}, \tag{4.12}$$

where $\mathbf{p} \in \mathbb{N}^n$ ranges over all vectors with $p_i \in \{1, 2, \dots, m\}$, such that $\#\left(\cup_{i=1}^{k} S_{p_i}\right) \geq k$, for all $k = 1, 2, \dots, n$.

Note that if S is a partition of the set of unknowns, then $\mathrm{per}(M_{\mathfrak{S}}, S) = \mathrm{per}(M_D, \mathbf{k})$, with $\mathbf{k} \in \mathbb{N}^m$, $k_j = \#S_j$, for $j = 1, 2, \dots, m$. By using the couple $(M_{\mathfrak{S}}, S)$, we can directly take advantage of multiple occurrences of the same set.

Example 4.1.21 (Example 4.1.7 continued.) Consider the set structure $\mathfrak{S}^{(1)}$ presented in Example 4.1.4. The array of all different sets is given by $S = [\{x_1\}, \{x_2\}, \{x_1, x_2\}]$. Table 4.3 displays the degree set structure table $(M_{\mathfrak{S}^{(1)}}, S)$ of $\mathfrak{S}^{(1)}$.

$\{x_1\}$	$\{x_2\}$	$\{x_1, x_2\}$
1	1	1
1	1	1

Table 4.3: Degree set structure table $(M_{\mathfrak{S}^{(1)}}, S)$ of $\mathfrak{S}^{(1)}$.

By expanding the rows of $M_{\mathfrak{S}^{(1)}}$ analogue to Algorithm 3.4.4, the permanent is computed as follows

$$\mathrm{per}(M_{\mathfrak{S}^{(1)}}, S) = 1 \times (1 + 1) + 1 \times (1 + 1) + 1 \times (1 + 1 + 1) = 7. \tag{4.13}$$

Note that $\mathrm{per}(M_{\mathfrak{S}^{(1)}}, S)$ equals $B_{\mathfrak{S}^{(1)}}$, see (4.2). ∎

Proposition 4.1.22 *Let* $(M_{\mathfrak{S}}, S)$ *be the degree set structure table of the set structure* $\mathfrak{S}$. *Then* $\mathrm{per}(M_{\mathfrak{S}}, S) = B_{\mathfrak{S}}$.

Proof. It suffices to see that the conditions on the vectors $\mathbf{p}$ in (4.12) also characterize an acceptable tuple. By enumerating all $\mathbf{p}$ that satisfy these conditions, we enumerate all acceptable tuples and obtain after summation $B_{\mathfrak{S}}$. □

Algorithm 4.1.23 Row expansion algorithm for computing $\mathrm{per}(A, S)$.

Input: (A, S), $A \in \mathbb{N}^{n\times m}$, $S = [S_1, S_2, \ldots, S_m]$.	*matrix and array of sets*
Output: $\mathrm{per}(A, S)$.	*permanent of A w.r.t. S*
$\mathrm{per}(A, \mathbf{k}) := \mathrm{per}(A, S, 1, T)$.	*call for backtracking*
with $\Sigma_i := \mathrm{per}(A, S, i, T)$:	*auxiliary function*
Input: $A \in \mathbb{N}^{n\times m}$, $S = [S_1, S_2, \ldots, S_m]$,	*matrix with array of sets*
$i \in \{1, 2, \ldots, n\}$,	*index of current row*
$T = (T_1, T_2, \ldots, T_n)$, $T_i \subseteq X$.	*current acceptable tuple*
Output: Σ_i.	*result of level i*
$\Sigma_i := 0$;	*initialization*
for all j in $\{1, 2, \ldots, m\}$ loop	
if $a_{ij} \neq 0$ and $\forall k \leq i-1$,	*test* S_j
$\forall Y \subseteq \{1, 2, \ldots, i-1\}$, $\#Y = k$:	*enumerate all subsets*
$\#\left(\cup_{l\in Y} T_l \cup S_j\right) \geq k+1$	*acceptable tuple?*
then $T_i := S_j$;	*choose* a_{ij}
if $i < n$	*update the sum*
then $\Sigma_i := \Sigma_i + a_{ij} \times \mathrm{per}(A, S, i+1, T)$;	*go to next level*
else $\Sigma_i := \Sigma_i + a_{ij}$;	
end if;	
end if;	
end loop.	

Due to the general concept of the permanent, Algorithm 4.1.23 contains an unavoidable combinatorial component, that is the incremental test for an acceptable tuple. To speed up this test, we order the sets in the candidate tuple T along the cardinalities and start the enumeration with k sets, where k equals the cardinality of the smallest set.

In order to solve the start system more efficiently, we can share the factorizations of the matrices and apply an incremental solver, treating the equations one by one. When arrived at the kth equation, all linear equations share the same factorization of the previous $k-1$ linear equations. Analogously to the formal algorithm, when at the kth equation we notice that the linear system cannot have a finite solution, we do not have to factorize the matrix further. These observations can significantly reduce the cost of solving the linear-product start system, especially when the number of sets in $\mathfrak{S}$ is larger than the dimension.

4.2 Symmetric set-structure analysis

This section is organized as follows. First, preliminaries concerning symmetric polynomial systems are given. Then, based on a symmetric set structure, it will be explained how to construct a symmetric random linear-product system. Finally, during continuation, only the generating solution paths must be traced.

4.2.1 Matrix representations

It is common to use groups in the mathematical description of symmetry, see [100]. Denote $GL(n, \mathbb{C})$ as the group of all regular $n \times n$ matrices with complex entries. For any finite group G, a *matrix representation* is given by the following homomorphism:

$$V : G \to GL(n, \mathbb{C}) : g \mapsto V(g) = V_g \tag{4.14}$$

such that the action of an element $g \in G$ on a vector $\mathbf{x}$ can be calculated as the matrix-vector product $V_g\mathbf{x}$. Note that hereby we regard $\mathbf{x}$ as written as a column vector.

Definition 4.2.1 Let $F(\mathbf{x}) = \mathbf{0}$ be a polynomial system and G a finite group. The solution set of $F(\mathbf{x}) = \mathbf{0}$, denoted by $F^{-1}(\mathbf{0})$, is *G-invariant* or simply *invariant* if $\forall g \in G$ and $\forall \mathbf{x} \in F^{-1}(\mathbf{0})$: $V_g\mathbf{x} \in F^{-1}(\mathbf{0})$.

The set of solutions generated by applying all transformations of G to one solution $\mathbf{x}^*$, is called a *G-orbit* or simply an *orbit* and is denoted by $\mathcal{O}_{\mathbf{x}^*}$. It is sufficient to compute only one solution of a G-orbit, if the solution set is known to be G-invariant.

Definition 4.2.2 (cf. Meravý [99]) Let F be a polynomial system and G a finite group. Let V, W be matrix representations. F is *(G, V, W)-symmetric* if $\forall g \in G$ there holds:

$$W_g F(\mathbf{x}) = F(V_g\mathbf{x}), \quad \forall \mathbf{x} \in \mathbb{C}^n. \tag{4.15}$$

One can also say that F *has a (G, V, W)-symmetric structure.*

Symmetric systems with $V = W$ are usually called *equivariant.*

The following example illustrates that systems exist, having both a G-invariant solution set, but with a different (G, V, W)-symmetric structure. This difference is crucial when considering homotopies.

Example 4.2.3 Consider the following polynomial systems F_A and F_B:

$$F_A(\mathbf{x}) = \left\{ \begin{array}{l} x_1^2 - 1 = 0 \\ x_2^2 - 1 = 0 \\ x_3^2 - 1 = 0 \end{array} \right. \quad \text{and} \quad F_B(\mathbf{x}) = \left\{ \begin{array}{l} x_3^2 - 2 = 0 \\ x_1^2 - 2 = 0 \\ x_2^2 - 2 = 0 \end{array} \right. . \tag{4.16}$$

Let S_3 denote the group of all permutations of a set of 3 elements. The generators of S_3 are represented by the matrices

$$V_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad V_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}. \tag{4.17}$$

Both systems have an S_3-invariant solution set. Denote $W_1^{F_A}$ for $W_1^{F_A} F_A(\mathbf{x}) = F_A(V_1\mathbf{x})$ and $W_1^{F_B}$ for $W_1^{F_B} F_B(\mathbf{x}) = F_B(V_1\mathbf{x})$, then

$$W_1^{F_A} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad W_1^{F_B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{4.18}$$

Because $(V_1, W_1^{F_A}) \neq (V_1, W_1^{F_B})$, the systems F_A and F_B do not have the same (S_3, V, W)-symmetric structure. ■

Because we consider only linear-product start systems, some restrictions on the group G and the system F are necessary. The symmetry group G will be restricted to a group which permutes and scales the unknowns of the polynomial system F, i.e., the matrix representations of elements $g \in G$ are of the form $V_g = DE$, where D is a diagonal and E a permutation matrix. The matrices W_g can be defined analogously. For notational simplicity, we restrict ourselves to permutation groups in the sequel.

4.2.2 Symmetric set structures

Before we can come to the construction of a (G, V, W)-symmetric start system, the underlying set structure must also be (G, V, W)-symmetric. In this section, the group actions will be applied to sets to define a (G, V, W)-symmetric set structure. Furthermore, the basic data structures will be given, together with the algorithm to verify if a given set structure is (G, V, W)-symmetric. As a byproduct, the data structures for the construction of a (G, V, W)-symmetric start system will be obtained.

The group action of $g \in G$ on a set $T \subseteq X$ can be defined as follows.

Definition 4.2.4 Let V_g be a matrix representation of $g \in G$. Associate with each $x_k \in T$ a vector $\mathbf{e}_k = (\delta_{k1}, \delta_{k2}, \ldots, \delta_{kn})^T$ where δ_{kl} equals the Kronecker delta. Then *the group action of g on T* is defined as $TV_g = \{\ x_l \mid (V_g\mathbf{e}_k)_l \neq 0,\ x_k \in T\ \}$.

Example 4.2.5 Let $X = \{x_1, x_2, x_3\}$ and $T = \{x_1, x_2\}$. Let V be a permutation matrix in $\mathbb{N}^{3\times 3}$ with nonzero entries v_{13}, v_{22} and v_{31} (like V_2 in (4.17)), then $TV = \{x_3, x_2\}$. ■

Definition 4.2.4 can be extended in a natural way for arrays of sets and for set structures.

Definition 4.2.6 Let V_g be the matrix representation of $g \in G$. The *group action on an array of sets* $S = [S_1, S_2, \ldots, S_m]$ is defined as $SV_g = [S_1V_g, S_2V_g, \ldots, S_mV_g]$, $\forall g \in G$. The *group action on a set structure* $\mathfrak{S} = (\mathfrak{S}_1, \mathfrak{S}_2, \ldots, \mathfrak{S}_n)$ is defined as $\mathfrak{S}V_g = (\mathfrak{S}_1V_g, \mathfrak{S}_2V_g, \ldots, \mathfrak{S}_nV_g)$, $\forall g \in G$.

Definition 4.2.7 Let G be a finite group with matrix representations V and W. The set structure $\mathfrak{S}$ is *(G, V, W)-symmetric* if $\forall g \in G: \ W_g\mathfrak{S} = \mathfrak{S}V_g$. Then $\mathfrak{S}$ is a *(G, V, W)-symmetric set structure.*

For the construction of a (G, V, W)-symmetric supporting set structure, we can generate a supporting array by the heuristic presented in Algorithm 4.1.16 and apply the group actions to construct the rest of the set structure.

Example 4.2.8 (Example 4.1.4 continued.) The solution set of system F in (4.1) remains invariant when x_1 and x_2 are interchanged. The symmetry group G contains two elements, represented by the matrices

$$V_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad V_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{4.19}$$

The corresponding matrices are $W_1 = V_1$ and $W_2 = V_2$. The supporting set structures, see Table 4.1, both have the same (G, V, W)-symmetric structure as F. ■

Now that we have defined what a (G, V, W)-symmetric set structure is, we have come to the point where we can define the data structures needed in the algorithm for verifying whether a given set structure is (G, V, W)-symmetric.

Definition 4.2.9 Let $\mathfrak{S} = (\mathfrak{S}_1, \mathfrak{S}_2, \ldots, \mathfrak{S}_n)$ be a (G, V, W)-symmetric set structure, $\mathfrak{S}_k = [\mathfrak{S}_{k1}, \mathfrak{S}_{k2}, \ldots, \mathfrak{S}_{km_k}]$, with $m_k = \#\mathfrak{S}_k$. Let T be part of $\mathfrak{S}_i$, which will be denoted as $T \in \mathfrak{S}_i$. The *dependency structure of T in $\mathfrak{S}$ w.r.t. G* is defined by the following array of indices:

$$D(T, \mathfrak{S}, G) = [(k_j, l_j)]_{j=1,2,\ldots,\#G}, \quad \text{with} \quad (W_{g^j})_{k_j i} \neq 0 \quad \text{and} \quad \mathfrak{S}_{k_j l_j} = T V_{g^j}. \tag{4.20}$$

Note that for large symmetry groups, e.g., the full permutation group, it could suffice to work only with the generators of the group, instead of the whole group.

Example 4.2.10 (Example 4.2.8 continued.) Consider again the set structure $\mathfrak{S}^{(1)}$, see Table 4.1. Take the first set $\{x_1\} \in \mathfrak{S}_1^{(1)}$. The dependency structure $D(\{x_1\}, \mathfrak{S}^{(1)}, G)$ will be constructed by considering all matrix representations V_j and W_j of Example 4.2.8 as follows

$$(W_1)_{11} \neq 0 \;\Rightarrow\; k_1 = 1 \quad \text{and} \quad \{x_1\} V_1 = \mathfrak{S}_{k_1 1} \;\Rightarrow\; l_1 = 1, \tag{4.21}$$

so that $(1, 1)$ will be added to $D(\{x_1\}, \mathfrak{S}^{(1)}, G)$. Analogously, for W_2 and V_2, $(2, 1)$ joins the dependency structure. So, $D(\{x_1\}, \mathfrak{S}^{(1)}, G) = [(1, 1), (2, 1)]$. ■

Definition 4.2.11 Let $\mathfrak{S}$ be a (G, V, W)-symmetric set structure. A *covering of $\mathfrak{S}$ w.r.t. G*, denoted by $C(\mathfrak{S}, G)$, is an array of dependency structures with the following property: $\forall k$, $\forall \mathfrak{S}_{kl} \in \mathfrak{S}_k$: (k, l) belongs to exactly one dependency structure of the covering.

By a covering, we have a partition of the sets in $\mathfrak{S}$ according to the symmetry group G. Algorithm 4.2.12 contains a description of the main steps to construct a covering.

Algorithm 4.2.12 Construction of a covering for set structure.

Input: $\mathfrak{S}$, (G, V, W). *(G, V, W)-symmetric set structure $\mathfrak{S}$*
Output: $C(\mathfrak{S}, G)$. *a covering*

while $\mathfrak{S} \neq (\emptyset, \emptyset, \ldots, \emptyset)$ loop
  $\exists k$: $T \in \mathfrak{S}_k$, $T \neq \emptyset$; *take a set T*
  if $D(T, \mathfrak{S}, G)$ can be constructed *dependency structure*
   then add $D(T, \mathfrak{S}, G)$ to $C(\mathfrak{S}, G)$; *update covering*
   else stop and report failure; *$\mathfrak{S}$ not symmetric*
  end if;
  for all $(k, l) \in D(T, \mathfrak{S}, G)$ loop
    $\mathfrak{S}_1 := \mathfrak{S}_1 \setminus T_{kl}$; *update $\mathfrak{S}$*
  end loop;
end loop.

Example 4.2.13 (Example 4.2.10 continued.) For the system F in (4.1), consider the dependency structures of the supporting sets of $\mathfrak{S}^{(1)}$ for the first equation, see Table 4.1:

$$D(\{x_1\}, \mathfrak{S}^{(1)}, G) = [(1,1),(2,1)], \quad D(\{x_1, x_2\}, \mathfrak{S}^{(1)}, G) = [(1,2),(2,2)] \tag{4.22}$$

and

$$D(\{x_2\}, \mathfrak{S}^{(1)}, G) = [(1,3),(2,3)], \tag{4.23}$$

yields a covering of the set structure $\mathfrak{S}^{(1)}$ w.r.t. the group G. ∎

Note that for one given set structure, more than one covering may exist. This is e.g. the case for the second set structure $\mathfrak{S}_2$ in Table 4.1. As some sets occur twice in the array, we can choose which one to connect with another. However, if no covering for a set structure can be constructed, then this indicates that it is not symmetric.

The role of a covering is twofold. First, the construction algorithm given in Algorithm 4.2.12, can be used for verifying if a certain set structure is (G, V, W)-symmetric. Second, a covering serves as the basic data structure for generating the coefficients of the (G, V, W)-symmetric start system.

4.2.3 Symmetric linear-product start systems

This section is concerned with the determination of the coefficients of a (G, V, W)-symmetric start system $F^{(0)}$. In order to obtain the same symmetric structure, not all coefficients of the start system may be chosen at random. The coefficients that may be randomly chosen are called *free coefficients*. Due to the symmetry relations, the original system itself might have no isolated solutions. Example 4.2.14 gives a very simple instance of what we call a *degenerate polynomial system*.

Example 4.2.14 Let $F(\mathbf{x}) = \mathbf{0}$ be a two-dimensional linear system whose solution set is invariant under the permutation $x_1 \leftrightarrow x_2$. More precisely, suppose that the symmetry group G has the following representations

$$V_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad V_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \text{and} \quad W_1 = V_1, \quad W_2 = V_1. \tag{4.24}$$

The only possibility for a (G, V, W)-symmetric linear system is a system like

$$F(\mathbf{x}) = \left\{ \begin{array}{l} c_1 x_1 + c_1 x_2 + c_0 = 0 \\ c_2 x_1 + c_2 x_2 + c_0 = 0 \end{array} \right. , \quad c_0, c_1, c_2 \in \mathbb{C}. \tag{4.25}$$

For almost all choices of the three free coefficients, in particular if $c_1 \neq c_2$ and $c_0 \neq 0$, then $F(\mathbf{x}) = \mathbf{0}$ can have no isolated roots. ∎

In the sequel, we will exclude degenerate systems. Based on a covering of a given set structure, we will deal with the coefficients as follows. For each dependency structure in the covering, choose free coefficients for the first set and apply then all group actions to generate the rest of the coefficients. This must be repeated until all coefficients are determined. On this idea Algorithm 4.2.16 is based, where a template $\mathfrak{T}$ is used to count the number of free coefficients $n_{\mathfrak{T}}$ in a symmetric linear-product start system.

Definition 4.2.15 A *template* $\mathfrak{T}$ is defined by the tuple $\mathfrak{T} = (\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n)$, where each $\mathcal{T}_k$ is an array of vectors in $\mathbb{N}^{n+1}$, denoted as $\mathcal{T}_k = [\mathbf{a}^{(kl)}]$, for $l = 1, 2, \ldots, \#\mathcal{T}_k$.

Algorithm 4.2.16 Counting the number of free coefficients for a (G, V, W)-symmetric linear-product system.

Input: $\mathfrak{S}$, G, $C(\mathfrak{S}, G)$.	*set structure, group and covering*
Output: $n_{\mathfrak{T}}$, $\mathfrak{T}$.	*#free coefficients, template*
$n_{\mathfrak{T}} := 0$; $\mathfrak{T} := (\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n)$;	*initialization*
for all k in $\{1, 2, \ldots, n\}$ loop	*make correspondence*
for all l in $\{1, 2, \ldots, \#\mathfrak{S}_k\}$ loop	*with* $\mathfrak{S}$
$\mathbf{a}^{(kl)} := (0, 0, \ldots, 0) \in \mathbb{N}^{n+1}$;	
end loop;	
end loop;	
for all $D(T, \mathfrak{S}, G) \in C(\mathfrak{S}, G)$ loop	*main loop*
let $T = \mathfrak{S}_{ij}$, then $\mathbf{a} := \mathbf{a}^{(ij)}$;	*corresponding vector*
$n_{\mathfrak{T}} := n_{\mathfrak{T}} + 1$; $a_0 := n_{\mathfrak{T}}$;	*next free coefficient*
for all $x_j \in T$ loop	
$n_{\mathfrak{T}} := n_{\mathfrak{T}} + 1$; $a_j := n_{\mathfrak{T}}$;	*next free coefficient*
end loop;	
for all $(k, l) \in D(T, \mathfrak{S}, G)$ loop	*generate template*
let $g \in G$: $\mathfrak{S}_{kl} = TV_g$,	*by applying*
then $a_0^{(kl)} := a_0$;	*corresponding*
$(a_1^{(kl)}, \ldots, a_n^{(kl)})^T := V_g(a_1, \ldots, a_n)^T$;	*group actions*
end loop;	
end loop.	

Note that Algorithm 4.2.16 assumes that all coefficients corresponding to the first set in each dependency structure can by chosen freely.

Example 4.2.17 (Example 4.2.13 continued.) Based on the covering $C(\mathfrak{S}^{(1)}, G)$ proposed in Example 4.2.13, we have that $n_{\mathfrak{T}} = 7$, as counted by constructing template $\mathfrak{T}$ displayed in Table 4.4.
Equivalently, we can write $\mathfrak{T} = ([(1, 2, 0), (3, 4, 5), (6, 0, 7)], [(1, 0, 2), (3, 5, 4), (6, 7, 0)])$. ∎

1	(1,2,0)	(3,4,5)	(6,0,7)
2	(1,0,2)	(3,5,4)	(6,7,0)

Table 4.4: Template $\mathfrak{T}$ based on $C(\mathfrak{S}^{(1)}, G)$.

Based on a template, we can define a (G, V, W)-symmetric linear-product start system.

Definition 4.2.18 Consider a template $\mathfrak{T} = (\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n)$, with $\mathcal{T}_k = [\mathbf{a}^{(kl)}]$, for $k = 1, 2, \ldots, n$. Denote by $n_{\mathfrak{T}}$ the number of free coefficients and let $\mathbf{c} \in \mathbb{C}_0^{n_{\mathfrak{T}}}$ be a random vector. We define a (G, V, W)*-symmetric random linear-product system* $F^{(0)}$ as $F^{(0)} = (f_1^{(0)}, f_2^{(0)}, \ldots, f_n^{(0)})$, where

$$f_k^{(0)}(\mathbf{x}) = \prod_{l=1}^{\#\mathcal{T}_k} \left(c_{a_0^{(kl)}} + \sum_{j=1}^{n} c_{a_j^{(kl)}} x_j \right), \quad k = 1, 2, \ldots, n. \tag{4.26}$$

Example 4.2.19 (Example 4.2.17 continued.) Based on the template in Table 4.4 the following symmetric random product system can be constructed:

$$F^{(0)}(\mathbf{x}) = \begin{cases} (c_1 + c_2 x_1)(c_3 + c_4 x_1 + c_5 x_2)(c_6 + c_7 x_2) = 0 \\ (c_1 + c_2 x_2)(c_3 + c_4 x_2 + c_5 x_1)(c_6 + c_7 x_1) = 0 \end{cases}, \tag{4.27}$$

where c_k, for $k = 1, 2, \ldots, 7$, are randomly chosen coefficients in $\mathbb{C}_0$. ■

Precise general conditions on the set structure and the symmetry relations under which a (G, V, W)-symmetric random product start system has exactly $B_{\mathfrak{S}}$ regular solutions are hard to find. Nevertheless, the work to be done for constructing such a start system is little compared to the actual solving of the target system.

4.2.4 Symmetric homotopy continuation

By using a start system with the same (G, V, W)-symmetric structure as the target system, we obtain a (G, V, W)*-symmetric homotopy*, which allows us to trace only the generating solution paths.

Lemma 4.2.20 *Let F and $F^{(0)}$ be two (G, V, W)-symmetric polynomial systems and the homotopy*

$$H(\mathbf{x}, t) = \gamma(1 - t)F^{(0)}(\mathbf{x}) + tF(\mathbf{x}), \quad \gamma \in \mathbb{C}_0, \quad t \in \mathbb{C}. \tag{4.28}$$

Then $\forall t \in \mathbb{C}$: $H(\mathbf{x}, t)$ is (G, V, W)-symmetric.

Proof. Let $g \in G$ with matrix representations V_g and W_g.

$$\begin{array}{rclcl} H(V_g\mathbf{x}) & = & \gamma(1-t)F^{(0)}(V_g\mathbf{x}) + tF(V_g\mathbf{x}) & = & \gamma(1-t)W_gF^{(0)}(\mathbf{x}) + tW_gF(\mathbf{x}) \\ & = & W_g(\gamma(1-t)F^{(0)}(\mathbf{x}) + tF(\mathbf{x})) & = & W_gH(\mathbf{x}) \end{array} \tag{4.29}$$

This completes the proof. □

Theorem 4.2.21 ensures that only those solution paths corresponding to the generating solutions should be traced.

Theorem 4.2.21 *Consider a (G, V, W)-symmetric polynomial system F with supporting set structure $\mathfrak{S}$. Let $F^{(0)}(\mathbf{x}) = \mathbf{0}$ be a (G, V, W)-symmetric random product start system based on $\mathfrak{S}$ with exactly $B_{\mathfrak{S}}$ regular solutions in $\mathbb{C}^n$. Then the usual convex-linear homotopy $H(\mathbf{x}, t) = \mathbf{0}$ is a good homotopy for computing all isolated solutions of $F(\mathbf{x}) = \mathbf{0}$. Moreover, only the generating solution paths need to be traced.*

Proof. By Lemma 4.2.20, all systems $H(\mathbf{x}, t) = \mathbf{0}$ have the same (G, V, W)-symmetric structure, for all t. By Theorem 4.1.13, no singularities can occur, so we have that all solutions of F can be reached and moreover, the symmetric structure of the solution paths remains invariant, for $t \in [0, 1)$, so it suffices to trace only the generation ones. □

Example 4.2.22 (Example 4.2.19 continued.) The start system $F^{(0)}$ presented in (4.27) has the same symmetric structure as the system F in (4.1). Table 4.5 displays the orbit structure of the solution list. The first column contains the type of solution, the second one the number of generating solutions and the last column indicates the number of generated solutions. The bottom row lists the total number of generating solutions and generated solutions.

type	$\#\mathbf{x}^*$	$\#\mathcal{O}_{\mathbf{x}^*}$
(a, a)	3	3
(a, b)	2	4
total:	5	7

Table 4.5: The orbit structure of the solution list.

By the fact that there are five generating solutions, only five continuation paths instead of seven need to be traced if a symmetric homotopy is applied. Theorem 4.2.21 guarantees that then all generating solutions of F will be found. Note that for the computation of the solutions of type (a, a) we can set $x_1 = x_2$, which yields an univariate equation of degree three, that can be solved more efficiently by other means than by following three paths. ■

We will summarize the method for constructing a (G, V, W)-symmetric linear-product start system. If a covering $C(\mathfrak{S}, G)$ can be constructed, then $\mathfrak{S}$ is (G, V, W)-symmetric. The next step consists in the construction of a template $\mathfrak{T}$ to check whether the $C(\mathfrak{S}, G)$ is degenerate and to count the number of free coefficients. When $C(\mathfrak{S}, G)$ is nondegenerate, the template contains all information to generate a (G, V, W)-symmetric a random linear-product system that will allows us to compute all generating solutions of the target system.

4.3 Applications

The algorithms has been applied to symmetric polynomial systems which appeared in the literature [13, 53, 123, 137]. We have chosen symmetric systems to illustrate the flexibility of the presented approach.

4.3.1 The polynomial systems

The polynomial systems are listed together with their symmetric supporting set structure and the orbit structure of their solutions.

Critical points of a Lotka–Volterra equation

An interesting class of symmetric polynomial systems has been given in [123]. The interior critical points modeled by a Lotka–Volterra equation satisfy the following

$$1 - cx_i + \sum_{j=1}^{n} \delta_{ij} x_i x_j^2 = 0, \qquad 1 \le i \le n, \tag{4.30}$$

where the connection matrix $\Delta_p^* = (\delta_{ij})$ is given by

$$\delta_{ij} = \left\{ \begin{array}{rl} 0, & \text{if} \quad i = j, \\ 1, & \text{if} \quad j \le p, \ i \ne j, \\ -1, & \text{if} \quad j > p, \ i \ne j, \end{array} \right. \qquad 1 \le i \le n, \tag{4.31}$$

where p is a given integer between 0 and n and where c is a certain constant.

This system has been solved analytically in [123] for $p = 0$. For $p = 0$ and $n = 3$, this system has been solved symbolically in [53].

For $n = p = 4$ the system to be solved is the following:

$$F_A(\mathbf{x}) = \left\{ \begin{array}{l} x_1x_2^2 + x_1x_3^2 + x_1x_4^2 - cx_1 + 1 = 0 \\ x_2x_1^2 + x_2x_3^2 + x_2x_4^2 - cx_2 + 1 = 0 \\ x_3x_1^2 + x_3x_2^2 + x_3x_4^2 - cx_3 + 1 = 0 \\ x_4x_1^2 + x_4x_2^2 + x_4x_3^2 - cx_4 + 1 = 0 \quad . \end{array} \right. \tag{4.32}$$

The total degree equals 81, while there are only 73 finite solutions. The solution set to F_A is symmetric w.r.t. the group of all permutations of the 4 unknowns. A (G, V, W)-symmetric supporting set structure is displayed in Table 4.6.

Note that this set structure remains useful for *any* value of p. In order to solve the system for a general choice of c, only 13 solution paths remain to be traced. We can compute the orbit structure of the solution list, see Table 4.7, already by solving the symmetric linear-product start system.

For $p = n$, we solved this system for dimensions $n = 4, 5, 6, 7$. The total degree of the system grows exponentially, while the number of generating solutions remains moderate, see Table 4.13. Note that applying m-homogenization does not improve the total degree as root count for this system.

1	$\{x_1\}$	$\{x_2, x_3, x_4\}$	$\{x_2, x_3, x_4\}$
2	$\{x_2\}$	$\{x_1, x_3, x_4\}$	$\{x_1, x_3, x_4\}$
3	$\{x_3\}$	$\{x_1, x_2, x_4\}$	$\{x_1, x_2, x_4\}$
4	$\{x_4\}$	$\{x_1, x_2, x_3\}$	$\{x_1, x_2, x_3\}$

Table 4.6: A (G, V, W)-symmetric supporting set structure for F_A

type	$\#\mathbf{x}^*$	$\#\mathcal{O}_{\mathbf{x}^*}$
(a, a, a, a)	3	3
(a, a, a, b)	4	16
(a, a, b, b)	3	18
(a, a, b, c)	3	36
total:	13	73

Table 4.7: The orbit structure of the solution list of F_A, $n = 4$.

The system of E.R. Speer

The following system has been given by E.R. Speer. In [53], a Gröbner basis has been computed for this system.

$$F_B(\mathbf{x}) = \left\{ \begin{array}{l} 4\beta(n + 2a_1 - 8x_1)(a_2 - a_3) - x_2x_3x_4 + x_2 + x_4 = 0 \\ 4\beta(n + 2a_1 - 8x_2)(a_2 - a_3) - x_1x_3x_4 + x_1 + x_3 = 0 \\ 4\beta(n + 2a_1 - 8x_3)(a_2 - a_3) - x_1x_2x_4 + x_2 + x_4 = 0 \\ 4\beta(n + 2a_1 - 8x_4)(a_2 - a_3) - x_1x_2x_3 + x_1 + x_3 = 0 \quad , \end{array} \right. \tag{4.33}$$

where

$$a_1 = x_1 + x_2 + x_3 + x_4, \qquad a_2 = x_1x_2x_3x_4, \qquad a_3 = x_1x_2 + x_2x_3 + x_3x_4 + x_4x_1 \tag{4.34}$$

and where β and n are parameters to the system. The total degree of this system is 625. This means that with the traditional homotopy, 625 continuation paths have to be traced. The minimal m-homogeneous Bézout number is obtained with a 4-homogenization, yielding $B_Z = 384$. The solution set to this system is invariant w.r.t. the Kleinian group. Table 4.8 contains a (G, V, W)-symmetric supporting set structure for F_B, with $B_{\mathfrak{S}} = 271$.

Starting at generating solutions, instead of 271 only 83 paths are left to trace in order to solve system F_B for any random choice of the parameters β and n. See Table 4.9 for the orbit structure of the solution list. Note that we can find a set structure that leads to 246 as corresponding generalized Bézout bound, though it does not respect the symmetric structure.

1	$\{x_1\}$	$\{x_1,x_2\}$	$\{x_2,x_3\}$	$\{x_3,x_4\}$	$\{x_4\}$
2	$\{x_2\}$	$\{x_2,x_1\}$	$\{x_1,x_4\}$	$\{x_4,x_3\}$	$\{x_3\}$
3	$\{x_3\}$	$\{x_3,x_4\}$	$\{x_4,x_1\}$	$\{x_1,x_2\}$	$\{x_2\}$
4	$\{x_4\}$	$\{x_4,x_3\}$	$\{x_3,x_2\}$	$\{x_2,x_1\}$	$\{x_1\}$

Table 4.8: A (G, V, W)-symmetric supporting set structure for F_B.

type	$\#\mathbf{x}^*$	$\#\mathcal{O}_{\mathbf{x}^*}$
(a,a,a,a)	5	5
(a,a,a,b)	2	8
(a,a,b,b)	23	46
(a,a,b,c)	12	48
(a,b,c,d)	41	164
total:	83	271

Table 4.9: The orbit structure of the solution list of F_B.

The construction of Virasoro algebras

The following system F_C showed up in the construction of Virasoro algebras, see [137]:

$$\left\{\begin{array}{l}
8x_1^2+8x_1(x_2+x_3)-8x_2x_3+2x_1(x_4+x_5+x_6+x_7)-2x_4x_7-2x_5x_6-x_1=0\\
8x_2^2+8x_2(x_1+x_3)-8x_1x_3+2x_2(x_4+x_5+x_6+x_7)-2x_4x_6-2x_5x_7-x_2=0\\
8x_3^2+8x_3(x_1+x_2)-8x_1x_2+2x_3(x_4+x_5+x_6+x_7)-2x_4x_5-2x_6x_7-x_3=0\\
8x_4^2+2x_1(x_4-x_7)+2x_2(x_4-x_6)+2(x_3+3x_8)(x_4-x_5)+2x_4(4x_5+x_6+x_7)-x_4=0\\
8x_5^2+2x_1(x_5-x_6)+2x_2(x_5-x_7)+2(x_3+3x_8)(x_5-x_4)+2x_5(4x_4+x_6+x_7)-x_5=0\\
8x_6^2+2x_1(x_6-x_5)+2x_2(x_6-x_4)+2(x_3+3x_8)(x_6-x_7)+2x_6(4x_7+x_4+x_5)-x_6=0\\
8x_7^2+2x_1(x_7-x_4)+2x_2(x_7-x_5)+2(x_3+3x_8)(x_7-x_6)+2x_7(4x_6+x_4+x_5)-x_7=0\\
8x_8^2+6(x_4+x_5+x_6+x_7)x_8-6x_4x_5-6x_6x_7-x_8=0\;.
\end{array}\right. \tag{4.35}$$

The symmetry group G has 8 elements and is generated by the following permutations of the unknowns

$$(x_4,x_5,x_6,x_7)\mapsto(x_5,x_4,x_7,x_6),\quad (x_4,x_5,x_6,x_7)\mapsto(x_6,x_7,x_4,x_5) \tag{4.36}$$

and

$$(x_1,x_2,x_6,x_7)\mapsto(x_2,x_1,x_7,x_6). \tag{4.37}$$

In (4.36) and (4.37), we have omitted the unknowns that remain at their position under the generating permutations. A (G, V, W)-symmetric supporting set structure for F_C is displayed in Table 4.10. In order to find all 256 finite solutions, only 76 paths have to be traced. Table 4.11 lists the orbit structure of the solutions. Note that, although $B_{\mathfrak{S}}$ equals the total degree, the use of set structures enables us to exploit the symmetry relations.

1	$\{x_1, x_2, x_4, x_5, x_6, x_7\}$	$\{x_1, x_3, x_4, x_5, x_6, x_7\}$
2	$\{x_2, x_1, x_4, x_5, x_6, x_7\}$	$\{x_2, x_3, x_4, x_5, x_6, x_7\}$
3	$\{x_3, x_1, x_2, x_4, x_5, x_6, x_7\}$	$\{x_3, x_1, x_2, x_4, x_5, x_6, x_7\}$
4	$\{x_4, x_1, x_2, x_3, x_8\}$	$\{x_4, x_5, x_6, x_7\}$
5	$\{x_5, x_1, x_2, x_3, x_8\}$	$\{x_5, x_4, x_7, x_6\}$
6	$\{x_6, x_1, x_2, x_3, x_8\}$	$\{x_6, x_7, x_4, x_5\}$
7	$\{x_7, x_1, x_2, x_3, x_8\}$	$\{x_7, x_6, x_5, x_4\}$
8	$\{x_8, x_4, x_5, x_6, x_7\}$	$\{x_8, x_4, x_5, x_6, x_7\}$

Table 4.10: A (G, V, W)-symmetric supporting set structure for F_C.

type	$\#\mathbf{x}^\star$	$\#\mathcal{O}_{\mathbf{x}^\star}$
(a, a, b, c, c, c, c, d)	16	16
(a, a, b, c, c, c, d, e)	8	32
(a, a, b, c, c, d, d, e)	16	48
(a, a, b, c, d, e, e, f)	20	80
(a, b, c, d, d, d, d, e)	8	16
(a, b, c, d, d, d, e, f)	4	32
(a, b, c, d, e, f, f, g)	4	32
total:	76	256

Table 4.11: The orbit structure of the solution list of F_C.

The cyclic 5-roots problem

The last system is the so-called cyclic 5-roots problem, given in [13]:

$$F_D(\mathbf{x}) = \left\{ \begin{array}{r} x_1 + x_2 + x_3 + x_4 + x_5 = 0 \\ x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_1 = 0 \\ x_1x_2x_3 + x_2x_3x_4 + x_3x_4x_5 + x_4x_5x_1 + x_5x_1x_2 = 0 \\ x_1x_2x_3x_4 + x_2x_3x_4x_5 + x_3x_4x_5x_1 + x_4x_5x_1x_2 + x_5x_1x_2x_3 = 0 \\ x_1x_2x_3x_4x_5 - 1 = 0 \end{array} \right. . \tag{4.38}$$

The total degree equals 120, but there are only 70 finite regular solutions. Moreover the system is symmetric. The generating permutations of the symmetry group are given by

$$\mathbf{x} \mapsto (x_2, x_3, x_4, x_5, x_1) \quad \text{and} \quad \mathbf{x} \mapsto (x_5, x_4, x_3, x_2, x_1). \tag{4.39}$$

By exploiting the symmetry, it suffices to compute only seven solutions.

We can find a set structure that yields a generalized Bézout number equal to 108, but it does not allow the exploitation of the symmetric structure. A (G, V, W)-symmetric supporting set structure is given in Table 4.12. The generalized Bézout bound that corresponds with this symmetric set structure equals 120. Because the components of all solutions are all different from each other, i.e., there is only one type of orbit, only 12 generating paths need to be traced, which makes this set structure more interesting than the other one.

1	$\{x_1, x_2, x_3, x_4, x_5\}$				
2	$\{x_1\}$	$\{x_2\}$	$\{x_3\}$	$\{x_4\}$	$\{x_5\}$
3	$\{x_1\}$	$\{x_2\}$	$\{x_3\}$	$\{x_4\}$	$\{x_5\}$
4	$\{x_1\}$	$\{x_2\}$	$\{x_3\}$	$\{x_4\}$	$\{x_5\}$
5	$\{x_1\}$	$\{x_2\}$	$\{x_3\}$	$\{x_4\}$	$\{x_5\}$

Table 4.12: A (G, V, W)-symmetric supporting set structure for F_D.

4.3.2 Performance

For each system, the dimension n, the total degree D, the minimal m-homogeneous Bézout number B_Z, the upper bound $B_{\mathfrak{S}}$ and the number of solution paths that have to be traced are listed in Table 4.13.

system	description	n	D	B_Z	$B_{\mathfrak{S}}$	$N(F)$	#paths
(4.30)	Lotka-Volterra	4	81	81	73	73	13
		5	243	243	233	233	19
		6	729	729	717	717	26
		7	2,187	2,187	2,173	2,173	34
(4.33)	system of Speer	4	625	384	271	50	83
(4.35)	Virasoro algebras	8	256	256	256	256	76
(4.38)	cyclic 5-roots	5	120	120	120	70	12

Table 4.13: Performance of the homotopies.

We see that the generalized Bézout number yields not only a better root count, but that in addition, by the exploitation of the symmetric structure, fewer paths need to be traced.

4.4 Conclusions and future research

Set-structure analysis is a valuable tool for the construction of an efficient homotopy. For many applications, the generalized Bézout number yields a sharper root count and allows to exploit more of the structure of the system to solve, which leads to a more performant homotopy.

When applied to a symmetric polynomial system, we can construct a random linear-product system with the same symmetric structure as the system that has to be solved. Hereby the structure of the solution set can be predicted, so that only the generating solutions must be computed. This can lead to a tremendous reduction of the computational work needed to solve the polynomial system.

As with multi-homogenization, one of the major problems for future research lies in the investigation of efficient algorithms for constructing an optimal set structure. It could also be worthwhile to find out in which circumstances a multi-homogeneous Bézout number can be improved using set structures.

Chapter 5

Polyhedral Homotopies

This chapter describes algorithms that yielded a computational breakthrough in solving sparse polynomial systems.

In the mid seventies, Bernshteĭn [9], Kushnirenko [77] and Khovanskiĭ [76] developed a root count in $\mathbb{C}_0^n$ for systems of Laurent polynomials, allowing negative exponents. Recently, Canny and Rojas nicknamed in [25] this root count *the BKK bound* after its inventors and proved the Vertex Coefficient Theorem.

In [72], Huber and Sturmfels introduced mixed subdivisions for the computation of the mixed volume. They developed polyhedral continuation and provided another proof of the BKK bound. Their approach is based on an idea of Betke [10] and consists in considering the lower hull of the polytopes embedded in a higher dimensional space. This is called the *lifting method*. It has been applied efficiently by Canny and Emiris, see [26, 41] and [43], who used it in their computation of sparse resultants. Pedersen and Sturmfels described in [129] the relation between Newton polytopes and resultants, see also [149]. We refer to [57] and [56] for the study of $\mathcal{A}$-hypergeometric differential equations as an illustration of powerful geometric techniques to solve algebraic problems. Another application of Newton polyhedra can be found in [20], where they have been applied to compute the local uniformization for all branches of a curve defined by a system of equations.

In [171], we reported on a combinatorial implementation of the algorithms Bernshteĭn proposed in [9]. Because we can interpret this method as a lifting method, we now call this method *implicit lifting*. The lifting method of Huber and Sturmfels is called a *static lifting* method, opposed to the *dynamic lifting algorithm*, developed in [166] and described in the next chapter. In this chapter we describe the explicit elaboration of the pruning methods needed in the lifting methods, as we presented them in [166]. Another amelioration of the static lifting method is its recursive elaboration, necessary when we developed *symmetric lifting*, see [165]. This extends the application range of the lifting method considerably.

This chapter is organized as follows. The next section contains the definition of the BKK bound. The following sections deal with the computation of mixed volumes and polyhedral continuation. A section containing applications illustrates the effectiveness of the polyhedral homotopies. Finally, we state our conclusions and give some suggestions for future research.

5.1 The theorems of Bernsheĭn

Bernshteĭn, Kushnirenko and Khovanskiĭ introduced an upper bound for the number of isolated solutions of sparse Laurent polynomial systems. In this section, we give the basic definitions of Newton polytopes and mixed volumes and illustrate the root count with an example. We also present Bernshteĭn's second theorem which provides conditions on the exactness of the mixed volume as root count.

Definition 5.1.1 A *Laurent polynomial* f is defined as

$$f(\mathbf{x}) = \sum_{\mathbf{a}\in A} c_{\mathbf{a}}\mathbf{x}^{\mathbf{a}}, \quad c_{\mathbf{a}} \in \mathbb{C}_0, \ \mathbf{x}^{\mathbf{a}} = x_1^{a_1}x_2^{a_2}\cdots x_n^{a_n}, \ A \subset \mathbb{Z}^n, \ \#A < \infty. \tag{5.1}$$

The set A is the *support* of f, $A = \mathrm{supp}(f)$. Its convex hull, $P = \mathrm{conv}(A)$, is the *Newton polytope* of f.

Definition 5.1.2 A *Laurent polynomial system* $F(\mathbf{x}) = \mathbf{0}$ is defined by a tuple of Laurent polynomials, $F = (f_1, f_2, \ldots, f_n)$. The tuples $\mathcal{A}$ and $\mathcal{P}$ collect respectively the supports and Newton polytopes:

$$\begin{array}{ll} \mathcal{A} = (A_1, A_2, \ldots, A_n), & A_j = \mathrm{supp}(f_j) \\ \mathcal{P} = (P_1, P_2, \ldots, P_n), & P_j = \mathrm{conv}(A_j) \end{array}, \quad j = 1, 2, \ldots, n. \tag{5.2}$$

Recently, a lot of research has been devoted to the computation of the mixed volume.

Definition 5.1.3 The *mixed volume* $V_n(\mathcal{P})$ of an n-tuple of polytopes $\mathcal{P}$ is

$$V_n(\mathcal{P}) := \sum_{I\subset\{1,2,\ldots,n\}} (-1)^{n-\#I} \,\mathrm{vol}_n\left(\sum_{i\in I} P_i\right), \tag{5.3}$$

where $\mathrm{vol}_n(P)$ equals the Euclidean volume of P and where $P_1 + P_2 = \{\ \mathbf{x}+\mathbf{y} \mid \mathbf{x} \in P_1, \mathbf{y} \in P_2\ \}$ is the *Minkowski sum* of P_1 and P_2.

If all polytopes in $\mathcal{P}$ are identical, then $V_n(P, P, \ldots, P) = n!\mathrm{vol}_n(P)$[1].

The following properties of the mixed volume illustrate why we can count the roots by computing the mixed volume of the tuple of supports. We refer to [135] for the theory of Minkowski addition and volumes. See also [22] for more properties of mixed volumes. The first two propositions have a straightforward analogue to root counting, while the other ones offer new tools for solving polynomial systems.

Proposition 5.1.4 *The mixed volume is a symmetric function in its arguments:* $V_n(\mathcal{P}) = V_n(P_{\sigma_1}, P_{\sigma_2}, \ldots, P_{\sigma_n})$, *for every permutation* σ *of* $(1, 2, \ldots, n)$.

Indeed, if we interchange the equations in the polynomial system, then the solutions remain the same.

[1] Note that scaling by the factor $n!$ makes that the mixed volume of the polytopes of a linear system equals one.

Proposition 5.1.5 *The mixed volume is invariant under a shift of the polytopes:* $V_n(\mathcal{P}) = V_n(P_1 + \mathbf{a}^{(1)}, P_2 + \mathbf{a}^{(2)}, \ldots, P_n + \mathbf{a}^{(n)})$*, with* $\mathbf{a}^{(k)} \in \mathbb{Z}^n$*, for* $k = 1, 2, \ldots, n$.

Indeed, if we multiply an equation with a monomial, then the number of solutions in $\mathbb{C}_0^n$ remains unchanged.

Proposition 5.1.6 *The mixed volume is monotone increasing:* *if* $\forall k : P_k \subset Q_k$*, then* $V_n(\mathcal{P}) \leq V_n(\mathcal{Q})$.

By this property, we can say that the sparser a system is, the easier it can be solved by polyhedral homotopy methods. By continuation, solutions of one system can be extended naturally to a larger system with more monomials.

Proposition 5.1.7 *The mixed volume is invariant under a unimodular transformation:* $V_n(\mathcal{P}) = V_n(U\mathcal{P})$*, with* $det(U) = \pm 1$ *and where* $U\mathcal{P} = (UP_1, UP_2, \ldots, UP_n)$.

The unimodular transformation $U\mathcal{P}$ corresponds with a change of coordinates that leaves the number of solutions invariant. See Section 5.2.1 for dealing with unimodular transformations.

Proposition 5.1.8 *The mixed volume is multi-linear:* $V_n(P_{11} + P_{12}, P_2, \ldots, P_n) = V_n(P_{11}, P_2, \ldots, P_n) + V_n(P_{12}, P_2, \ldots, P_n)$.

Decomposing Newton polytopes corresponds to factoring the polynomials in the system. To count the number of roots we then need to add the number of roots of the components. We will elaborate this property further in Section 5.2.6.

Theorem 5.1.9, proven by Bernshteĭn in [9], gives the crucial importance of mixed volumes for systems of polynomial equations.

Theorem 5.1.9 (Bernshteĭn) *Let* F *be a tuple of* n *Laurent polynomials with Newton polytopes* $\mathcal{P}$*. Then the number of isolated solutions of* $F(\mathbf{x}) = \mathbf{0}$ *in* $\mathbb{C}_0^n$ *is bounded by the mixed volume* $V_n(\mathcal{P})$*. For almost all choices of the coefficients the system* $F(\mathbf{x}) = \mathbf{0}$ *has exactly as many roots in* $\mathbb{C}_0^n$ *as* $V_n(\mathcal{P})$.

We refer to [22, Chapter 4, §27] for more on the importance of mixed volumes in algebraic geometry. A nice interpretation of the Newton polytope of a Laurent polynomial is given by the moment map (see [56, Chapter 6, §1]), as it models the asymptotic behaviour of the roots.

Example 5.1.10 Consider the following polynomial system

$$F(\mathbf{x}) = \left\{ \begin{array}{r} c_{12}x_1^2 + c_{11}x_2^2 + c_{10} = 0 \\ c_{23}x_1^2x_2 + c_{22}x_1x_2^2 + c_{21}x_1 + c_{20}x_2 = 0 \end{array} \right., \quad c_{ij} \in \mathbb{C}_0. \tag{5.4}$$

Figure 5.1 depicts the Newton polytopes of the system and their Minkowski sum. Applying (5.3) the BKK bound can be computed as

$$V_2(P_1, P_2) = \mathrm{vol}_2(P_1 + P_2) - \mathrm{vol}_2(P_1) - \mathrm{vol}_2(P_2) = 10 - 2 - 2 = 6. \tag{5.5}$$

By Theorem 5.1.9, the system $F(\mathbf{x}) = \mathbf{0}$ has, for almost all choices of the coefficients, six isolated solutions in $\mathbb{C}_0^2$. ∎

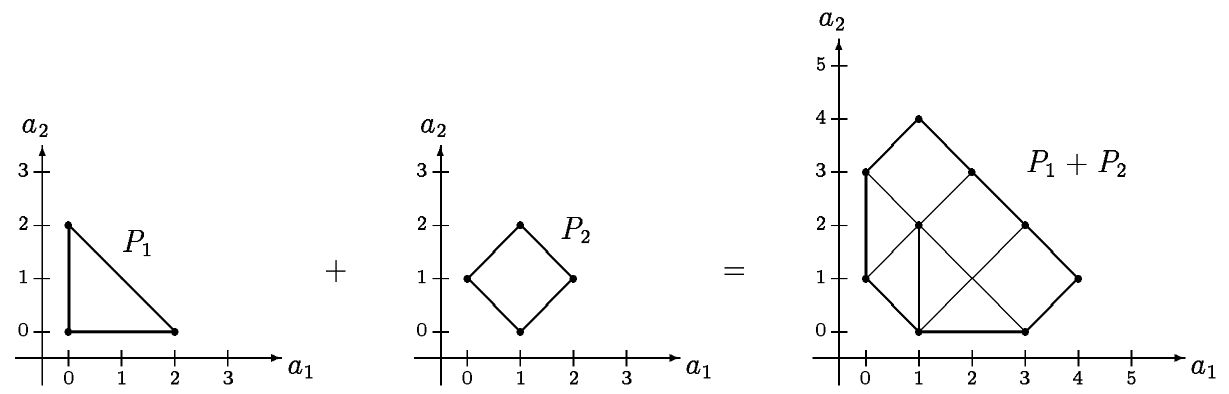


Figure 5.1: Newton polytopes (P_1, P_2) of F, with their Minkowski sum $P_1 + P_2$, constructed by placing P_2 at the vertices of P_1.

Note that formula (5.3) is not an efficient way for computing the mixed volume. In the sequel, we will develop more efficient algorithms. First we need some definitions. We restrict ourselves to polytopes spanned by integer vertices. General definitions concerning polytopes and support functions can be found in the literature, see e.g. [17, 80].

It is common practice (see e.g. [124]) to use the inner product $\langle \cdot, \cdot \rangle$ as a bilinear pairing between the space of points and the dual space of all linear functionals on this space. More precisely, we have

$$\langle \cdot, \cdot \rangle \; : \; V \times V^* \; \rightarrow \; \mathbb{R} \; : \; (\mathbf{v}, \mathbf{u}) \; \mapsto \; \langle \mathbf{v}, \mathbf{u} \rangle = \sum_{k=1}^{n} v_k u_k, \tag{5.6}$$

where $V \cong \mathbb{R}^n$ is the vector space where the polytopes live, and by V^* we denote the dual vector space $V^* = \{\mathbf{u} : V \rightarrow \mathbb{R} : \mathbf{v} \mapsto \langle \mathbf{v}, \mathbf{u} \rangle\}$, that is the space of all linear functionals on V. When we consider the support set $A \subset \mathbb{Z}^n$, instead of V, we have a module $M \cong \mathbb{Z}^n$ and view the inner product $\langle \cdot, \cdot \rangle$ as a $\mathbb{Z}$-bilinear pairing.

In the sequel, we use the following notation. Points in the primal space will be denoted as row vectors $\mathbf{a} = (a_1, a_2, \ldots, a_n)$, enclosed by ordinary parentheses, whereas the linear functionals $\mathbf{v}$ will be denoted as column vectors $\mathbf{v} = [v_1, v_2, \ldots, v_n]^T$, using square brackets to enclose its components. Whenever needed, the transpose operator ^{T} will be used to establish this convention.

Definition 5.1.11 Let $P = \mathrm{conv}(A)$, spanned by $A \subset \mathbb{Z}^n$. Then, *the support function p* on A is defined by

$$p : \; \mathbb{Z}^n \; \rightarrow \; \mathbb{Z} \; : \; \mathbf{v} \; \mapsto \; p(\mathbf{v}) \; = \; \max_{\mathbf{a} \in A} \; \langle \mathbf{a}, \mathbf{v} \rangle, \tag{5.7}$$

where each $\mathbf{v}$ belongs to the dual space of linear functionals. To extend the support function on P, we simply have to consider the $\mathbb{R}$-bilinear paring between the vector space $V \cong \mathbb{R}^n$ and its dual.

The following property follows naturally from the definition.

Proposition 5.1.12 *If* $\mathbf{0} \in A$, *then* $p(\mathbf{v}) \geq 0$, $\forall \mathbf{v} \in \mathbb{Z}^n$.

A polytope P is characterized by its support function p. All points of P that have the same maximal value when evaluated by the support function in the direction $\mathbf{v}$ determine a face of P, the so-called face in the direction $\mathbf{v}$.

Definition 5.1.13 Let $\partial_{\mathbf{v}} A = \{\ \mathbf{a} \in A \mid \langle \mathbf{a}, \mathbf{v} \rangle = \max_{\mathbf{b} \in A} \langle \mathbf{b}, \mathbf{v} \rangle\ \}$, then *the face of the polytope* P *in the direction* $\mathbf{v}$ is defined as the convex hull of $\partial_{\mathbf{v}} A$ and denoted by $\partial_{\mathbf{v}} P = \mathrm{conv}(\partial_{\mathbf{v}} A)$.

When we use the max-convention to define a face $\partial_{\mathbf{v}} P$, $\mathbf{v}$ is called an *outer normal* to P. Then $-\mathbf{v}$ is an *inner normal* to P and in the definition of $\partial_{\mathbf{v}} P$ we could accordingly substitute max by min. See Figure 5.2 for an example.

Definition 5.1.14 Consider a set A of n-dimensional vectors. The *dimension* of A equals d, denoted by $\dim(A) = d$, if A contains at most $d+1$ points $\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_d$ such that $\mathbf{a}_0 - \mathbf{a}_j$, $j = 1, 2, \ldots, d$, are linearly independent.

We call a face of dimension k *a k-face.* A face of a polytope of dimension 0 is called *a vertex, an edge* is a one-dimensional face. If $\dim(P) = d$, then a face of dimension $d-1$ is *a facet.* The polytope itself is considered as a *trivial face.* All other faces of a polytope, the empty set included, are regarded as *proper faces.*

The relation between faces of the Newton polytopes and the corresponding polynomial system is given in the following definitions.

Definition 5.1.15 Let $\mathbf{v} \in \mathbb{Z}^n \setminus \{\mathbf{0}\}$. Consider the polynomial $f(\mathbf{x})$ with support set A, then

$$f(\mathbf{x}) = \sum_{\mathbf{a} \in A} c_{\mathbf{a}}\, \mathbf{x}^{\mathbf{a}} \quad \text{and} \quad \partial_{\mathbf{v}} f(\mathbf{x}) = \sum_{\mathbf{a} \in \partial_{\mathbf{v}} A} c_{\mathbf{a}}\, \mathbf{x}^{\mathbf{a}} \tag{5.8}$$

is called *the projection of f along the direction* $\mathbf{v}$. Shortly, we say $\partial_{\mathbf{v}} f$ is *a face polynomial.*

Definition 5.1.16 Let $F = (f_1, f_2, \ldots, f_n)$ denote a polynomial system, with supports $\mathcal{A} = (A_1, A_2, \ldots, A_n)$. Consider $\mathbf{v} \in \mathbb{Z}^n \setminus \{\mathbf{0}\}$. Then *the face system* $\partial_{\mathbf{v}} F$ *of* F is defined as the subsystem $\partial_{\mathbf{v}} F = (\partial_{\mathbf{v}} f_1, \partial_{\mathbf{v}} f_2, \ldots, \partial_{\mathbf{v}} f_n)$ with $\partial_{\mathbf{v}} f_i$ the face polynomial of f_i, for $i = 1, 2, \ldots, n$.

Conditions for which the mixed volume is an exact root count are given by Bernshteĭn's second theorem, see [9, Theorem B].

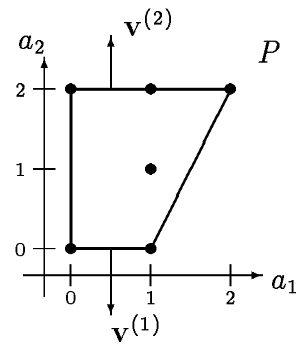
Theorem 5.1.17 (Bernshteĭn) *Let* $F(\mathbf{x}) = \mathbf{0}$ *be a polynomial system with* $\mathcal{P}$ *its corresponding tuple of Newton polytopes. If* $\forall \mathbf{v} \in \mathbb{Z}^n \setminus \{\mathbf{0}\} : \partial_{\mathbf{v}} F(\mathbf{x}) = 0$ *has no solution in* $\mathbb{C}_0^n$, *then* $N(F) = V_n(\mathcal{P})$. *Otherwise,* $N(F) < V_n(\mathcal{P})$.

Note that the solution of $\partial_{\mathbf{v}} F(\mathbf{x}) = 0$ in $\mathbb{C}_0^n$ does not need to be isolated. The face $\partial_{\mathbf{v}} \mathcal{P}$ for which the corresponding system $\partial_{\mathbf{v}} F(\mathbf{x}) = 0$ has a solution in $\mathbb{C}_0^n$ will be called a *defective face of* $\mathcal{P}$. It is sufficient to consider 'only' the normals $\mathbf{v}$ to all faces of the Minkowski sum $\sum_{i=1}^n P_i$. Canny and Rojas proved in [25] the Vertex Coefficient Theorem, which improves Theorem 5.1.17 in the case that the mixed volume is exact as root count, see also [133] for more results. However, for deficient polynomial systems, also coefficients corresponding to internal points of the polytope play an important role, as is illustrated by the following example.

Example 5.1.18 In Figure 5.2 the Newton polytope P is pictured of a deficient polynomial system $F(\mathbf{x}) = \mathbf{0}$.

$$F(\mathbf{x}) = \left\{ \begin{array}{l} x_1^2x_2^2 - 3x_1x_2^2 + 2x_2^2 + 3x_1x_2 + x_1 + 1 = 0 \\ x_1^2x_2^2 - 4x_1x_2^2 + 3x_2^2 + x_1x_2 + 2x_1 + 2 = 0 \end{array} \right. \tag{5.9}$$

As $P_1 = P_2$, denote $P := P_1$, the mixed volume $V_n(\mathcal{P}) = 2!\mathrm{vol}_2(P) = 6$. However, there are only 4 isolated solutions in $\mathbb{C}_0^2$.



$$\partial_{\mathbf{v}^{(1)}}F(\mathbf{x}) = \left\{ \begin{array}{r} x_1 + 1 \ = 0 \\ 2(x_1+1) = 0 \end{array} \right.$$

$$\partial_{\mathbf{v}^{(2)}}F(\mathbf{x}) = \left\{ \begin{array}{l} x_2^2(x_1-1)(x_1-2) = 0 \\ x_2^2(x_1-1)(x_1-3) = 0 \end{array} \right.$$

Figure 5.2: The Newton polytope P of F and the face systems $\partial_{\mathbf{v}^{(1)}}F(\mathbf{x})$ and $\partial_{\mathbf{v}^{(2)}}F(\mathbf{x})$.

For $\mathbf{v}^{(1)} = [0, -1]^T$, the face system $\partial_{\mathbf{v}^{(1)}}F(\mathbf{x}) = \mathbf{0}$ has the solution $(-1, \alpha)$, for any $\alpha \in \mathbb{C}$. This corresponds to the solution $(-1, 0)$ of the system $F(\mathbf{x}) = \mathbf{0}$. The multiplicity of this solution is determined by the coefficients of the monomial x_1x_2, i.e., if we take the coefficient of x_1x_2 equal to zero, then the Jacobian matrix of F at the solution $(-1, 0)$ is singular, otherwise we have a regular solution. For $\mathbf{v}^{(2)} = [0, 1]^T$, the face system $\partial_{\mathbf{v}^{(2)}}F(\mathbf{x}) = \mathbf{0}$ has the solution $(1, \alpha)$, for any $\alpha \in \mathbb{C}$. This corresponds to a solution at infinity of the system $F(\mathbf{x}) = \mathbf{0}$. ■

We conclude this section by observing that the mixed volume provides an exact root count when the relative position of the Newton polytopes w.r.t. each other is sufficiently generic, as we will elaborate next. The application of Theorem 5.1.17 on Example 5.1.10 inspired the following proposition.

Proposition 5.1.19 *Consider the polynomial system F with supports $\mathcal{A}$ and corresponding Newton polytopes $\mathcal{P}$. If $\forall \mathbf{v} \in \mathbb{Z}^n \setminus \{\mathbf{0}\}$, $\exists i : \#\partial_{\mathbf{v}}A_i = 1$, then $N(F) = V_n(\mathcal{P})$.*

Proof. $\#\partial_{\mathbf{v}}A_i = 1$ implies that the ith equation of the system $\partial_{\mathbf{v}}F(\mathbf{x}) = \mathbf{0}$ consists of the monomial $\mathbf{x}^{\mathbf{a}}$. Either $\mathbf{a} = \mathbf{0}$ yields the equation $1 = 0$ or $\exists i$: $a_i \neq 0$, which is equivalent to $x_i = 0$, so that also in this case, $\partial_{\mathbf{v}}F(\mathbf{x}) = \mathbf{0}$ has no solution in $\mathbb{C}_0^n$. Hence, by Theorem 5.1.17, $N(F) = V_n(\mathcal{P})$. □

At first sight, Proposition 5.1.19 only seems to have theoretical interests, as the number of possible values for $\mathbf{v}$ is infinite. However, by considering Figure 5.1 again, it becomes clear that there are only seven possible candidates for $\mathbf{v}$, namely all normals to the edges of P_1 and P_2. Any other normal will only yield two vertex points of each polytope. In fact, the only candidate normal that remains is the direction $\mathbf{v} = [1,1]^T$. Only for this normal, the face system $\partial_{\mathbf{v}} F(\mathbf{v}) = \mathbf{0}$ can have a solution in $\mathbb{C}_0^n$.

The tuple of Newton polytopes of a system which satisfies the condition of Proposition 5.1.19 is said to be *in generic position*: for every nonzero choice of the coefficients, it has exactly as many isolated solutions as its mixed volume. Consequently we can state that a coefficient of a monomial which corresponds to a vertex point of a support set does not necessarily influence the exactness of the BKK bound as root count.

In order to decide whether a given face system has a solution in $\mathbb{C}_0^n$, it might be worth to apply mixed resultants, see also the Appendix of [72]. Hereby we only have to consider the facet resultants. But first the condition of Proposition 5.1.19 needs to be checked. This can be done by studying the sequence of directions generated by the implicit lifting algorithm, presented in the following section. If no recursion is needed, then the polytopes are in generic position.

5.2 Implicit Lifting

This section describes how to compute (mixed) volumes recursively. First we give the basic algorithms for constructing unimodular transformations. The recursive algorithm is then elaborated for volumes and mixed volumes. The mixed-homotopy continuation runs analogously to the same lines of the recursive algorithm for computing the mixed volume. The last two subsections deal with some efficiency considerations, where we explain how to solve simplicial systems efficiently and how the new polyhedral methods can be combined with the product homotopies.

5.2.1 Integer linear algebra

We need to exploit the fact that we deal with polytopes with integer vertices, because the manipulations with Newton polytopes have significant interpretations in solving Laurent polynomial systems. Euclid's algorithm is the basic tool for calculations with matrices with integer entries, as explained in [15, pages 303–320]. See [1, 93, 122] for a more thorough understanding of module theory.

The greatest common divisor d of two integer numbers a and b is

$$d = \gcd(a,b) = ka + lb, \quad k, l \in \mathbb{Z} \tag{5.10}$$

and the numbers k and l can be obtained with Euclid's algorithm. In matrix notation, this can be formulated as

$$U \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} d \\ 0 \end{bmatrix} \text{ where } U = \begin{bmatrix} k & l \\ -\frac{b}{d} & \frac{a}{d} \end{bmatrix} \text{ and } \det(U) = 1. \tag{5.11}$$

The matrix U can be considered as the discrete analogue of a Givens rotation. It can be used to introduce zeros in vectors and matrices. In the multivariate case, let $U(i,j,a,b)$ be the identity matrix except for the entries

$$(U(i,j,a,b))_{ii} = k, \quad (U(i,j,a,b))_{ij} = l, \quad (U(i,j,a,b))_{ji} = -\frac{b}{d} \text{ and } (U(i,j,a,b))_{ij} = \frac{a}{d}. \tag{5.12}$$

Then $\det(U(i,j,a,b)) = 1$. Invertible integer matrices U are called *unimodular*. The unimodular transformations U of interest are those that do not alter the (mixed) volume, i.e., with $\det(U) = \pm 1$, see Proposition 5.1.7.

These elementary matrices are used to implement the following algorithms. The unimodular transformations created by Algorithm 5.2.1 applied to the columns of an integer matrix can be used to triangulate this matrix.

Algorithm 5.2.1 Rotation of a vector by unimodular transformation.

Input: $\mathbf{q} \in \mathbb{Z}^n$, $\mathbf{q} \neq \mathbf{0}$.	*a nonzero vector*
Output: $U \in \mathbb{Z}^{n \times n}$, $\det(U) = \pm 1$, $U\mathbf{q} = (d, 0, \dots, 0)^T$,	*unimodular transformation*
$d = \gcd(q_1, q_2, \dots, q_n)$.	
Let $U_1 \in \mathbb{Z}^n$: $\mathbf{q}' = U_1 \mathbf{q}$, $q_1' \neq 0$;	U_1 *is permutation matrix*
$d_1 := q_1'$;	*initialization*
for $j \in \{2, 3, \dots, n\}$ loop	*successive*
$d_j := \gcd(d_{j-1}, q_j')$;	gcd-*computations for*
$U_j := U(1, j, d_{j-1}, q_j')$;	*discrete Givens rotations*
end loop;	
$U := U_n \cdots U_2 U_1$.	*rotation matrix*

By applying the unimodular transformation created in Algorithm 5.2.2 to an outer normal $\mathbf{v}$ of the face $\partial_{\mathbf{v}} P$, all points in $U \partial_{\mathbf{v}} A$ will have the same first coordinate, see Figure 5.3. Note that, although $\mathrm{vol}_n(P) = \mathrm{vol}_n(UP)$, $\mathrm{vol}_n(\partial_{\mathbf{v}} P) \neq \mathrm{vol}_n(U \partial_{\mathbf{v}} P)$.

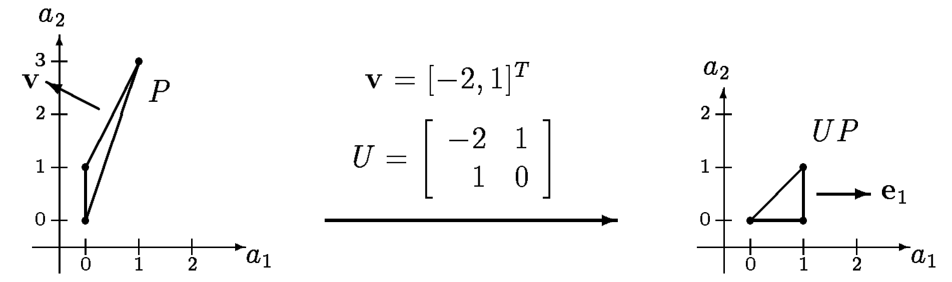


Figure 5.3: Unimodular transformation that makes the face $\partial_{\mathbf{v}} P$ to stand orthogonal to the first axis, with outer normal $\mathbf{e}_1$.

Algorithm 5.2.2 Construction of a unimodular transformation that contains a vector.

Input: $\mathbf{v} \in \mathbb{Z}^n$, $\mathbf{v} \neq \mathbf{0}$.	*a nonzero vector*
Output: $U \in \mathbb{Z}^{n \times n}$, $\det(U) = \pm 1$, $\forall \mathbf{q} \in \mathbb{Z}^n$:	*unimodular transformation*
$(U\mathbf{q})_1 = \frac{\langle \mathbf{v}, \mathbf{q} \rangle}{d}$, $d = \gcd(v_1, v_2, \dots, v_n)$.	*with first row* $\frac{\mathbf{v}}{d}$
Let $W = W_n \cdots W_2 W_1 \in \mathbb{Z}^{n \times n}$, $\det(W) = \pm 1$:	*unimodular transformation*
$W\mathbf{v} = d\mathbf{e}_1$, with $\mathbf{e}_1 = [1, 0, \dots, 0]^T$;	*Apply Algorithm* 5.2.1
$U := W^{-T}$.	$(U\mathbf{q})_1 = \mathbf{q}^T U^T \mathbf{e}_1 = \frac{\mathbf{q}^T \mathbf{v}}{d}$

To construct an orthogonal basis of the space spanned by a set of vectors, Algorithm 5.2.3 can be applied successively to the vectors in that set.

Algorithm 5.2.3 Orthogonalization of vectors.

Input: $\mathbf{v}^{(k)} \in \mathbb{Z}^n$, $k = 1, 2, \ldots, m$, $m < n$.	*sequence of vectors*
Output: $\mathbf{v} \in \mathbb{Z}^n$, $\mathbf{v} \neq \mathbf{0}$: $\mathbf{v} \perp \mathbf{v}^{(k)}$, $k = 1, 2, \ldots, m$.	*orthogonal vector*
Let $A = [\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(m)}]^T \in \mathbb{Z}^{m\times n}$: $A\mathbf{v} = \mathbf{0}$.	$\mathbf{v}$ *as solution of linear system*
Let $U \in \mathbb{Z}^{n\times n}$, $\det(U) = \pm 1$: $A' = UA$,	*Apply Algorithm* 5.2.1
$a'_{ij} = 0$, $\forall i > j$;	A' *is upper triangular*
$\mathbf{v} \in \mathbb{Z}^n$: $A'\mathbf{v} = \mathbf{0}$.	*solve triangular system*

These algorithms are the basic operations needed for computing the volume and the mixed volume. For efficiency reasons, we need Bareiss-like algorithms, see [55], to control the growth of the coefficients in the integer matrices. Because the elements of the matrices are (tiny) exponents of polynomials, we trust that no symbolic computation of integer numbers will be necessary.

The matrix $U \in \mathbb{Z}^{n\times n}$, with $\det(U) = \pm 1$, defines a change of variables

$$x_j = \prod_{i=1}^{n} y_i^{U_{ij}}, \; j = 1, 2, \ldots, n, \text{ abbreviated as } \mathbf{x} = \mathbf{y}^U. \tag{5.13}$$

This change of variables induces the one-to-one correspondence $\mathbf{x}^{\mathbf{a}} = \mathbf{y}^{U\mathbf{a}}$ between monomials in $\mathbf{x}$ and $\mathbf{y}$. Because U is unimodular, the number of solutions in $\mathbb{C}_0^n$ of $F(\mathbf{x} = \mathbf{y}^U) = \mathbf{0}$ is the same as the BKK bound of $F(\mathbf{x}) = \mathbf{0}$, see Proposition 5.1.7.

5.2.2 The volume of a polytope

By characterizing a polytope P by its support function p, $\mathrm{vol}_n(P)$ can be computed according to the following formula [17, page 38], [80, page 161]:

$$\mathrm{vol}_n(P) = \frac{1}{n} \sum_{\mathbf{v}} p(\mathbf{v}) \; \mathrm{vol}_{n-1}(\partial_{\mathbf{v}} P), \tag{5.14}$$

where $\mathbf{v} \in \mathbb{Z}^n$ ranges over all normalized directions, i.e., $\gcd(v_1, v_2, \ldots, v_n) = 1$. The hyperplane containing $\partial_{\mathbf{v}} P$ is mapped to $\mathbb{R}^{n-1}$ so that the points with integer coordinates are mapped to $\mathbb{Z}^{n-1}$, $\mathrm{vol}_{n-1}(\partial_{\mathbf{v}} P)$ represents the volume of $\partial_{\mathbf{v}} P$ mapped into $\mathbb{R}^{n-1}$, see Figure 5.3. The factor $\frac{1}{n}$ ensures that the unit cube has volume one. Formula (5.14) gives a recursive algorithm for the computation of the volume. In Algorithm 5.2.4, we will discard this factor so that the unit simplex has volume one.

There are only a finite number of normalized outer normals for which $p(\mathbf{v})\mathrm{vol}_{n-1}(\partial_{\mathbf{v}} P)$ is different from zero. In [171], we proposed Algorithm 5.2.4, which consists of an exhaustive enumeration of all n-tuples C from the set A. Each tuple of n points defines $n - 1$ vectors, which are input to Algorithm 5.2.3 for computing an outer normal $\mathbf{v}$. The function Outer_Normal used in Algorithm 5.2.4 has also A as argument to obtain the right orientation of the normal $\mathbf{v}$, so that $p(\mathbf{v}) \geq 0$. For nonsimplicial polytopes, some directions $\mathbf{v}$ will be generated more than once. Therefore we maintain a list $\mathcal{V}$ of outer normals $\mathbf{v}$ that yield a contribution to the volume of P. The argument of vol_{n-1} can be constructed by application of an unimodular transformation U created by Algorithm 5.2.2, so that the first component of each transformed point can be deleted. We denote the volume of this transformed polytope by $\mathrm{vol}_{n-1}((U\partial_{\mathbf{v}}P)_2^n)$.

Algorithm 5.2.4 Enumerative combinatorial volume computation.

Input: $A \subset \mathbb{Z}^n$, $\#A < \infty$. — $P = \mathrm{conv}(A)$
Output: $\mathrm{vol}_n(P)$, $\mathcal{V} = \{\ \mathbf{v} \mid p(\mathbf{v})\mathrm{vol}_{n-1}(\partial_{\mathbf{v}}P) > 0\ \}$. — *volume and outer normals*

when $n = 1$: — *trivial case*
  return $\left(\max\limits_{\mathbf{a}\in A} a_1 - \min\limits_{\mathbf{a}\in A} a_1\right)$. — *length of interval*

when $n > 1$: — *recursion needed*
  $\Sigma := 0$; $\mathcal{V} := \emptyset$; — *initialization*
  for all $C \subset A$, $\#C = n$ loop — *enumerate all n-tuples*
    $\mathbf{v} :=$ Outer_Normal(A, C); — *$C \subset \partial_{\mathbf{v}}A$ and $p(\mathbf{v}) \geq 0$*
    if $\mathbf{v} \notin \mathcal{V}$ and $p(\mathbf{v}) > 0$
     then $\Sigma := \Sigma + p(\mathbf{v}) \times \mathrm{vol}_{n-1}((U\partial_{\mathbf{v}}P)_2^n)$; — *transform to $n-1$*
        $\mathcal{V} := \mathcal{V} \cup \{\mathbf{v}\}$; — *update $\mathcal{V}$*
    end if;
  end loop. — *return Σ and $\mathcal{V}$*

We have made the enumerative Algorithm 5.2.4 more efficient by using linear programming methods for enumerating the normals to all facets of a polytope, by using similar concepts as proposed in [5]. For enumerating all faces of a polytope we refer to [139], and for computing volumes of convex polytopes, see e.g. [30] and [78]. A general treatment of computational geometry can be found in [130]. In [39], an introduction to the combinatorial aspects of geometric problems is given.

5.2.3 Mixed-volume computation

In [17, page 42] and in [80, page 170], formula (5.14) for computing the volume of a polytope is generalized into a formula for the mixed volume of an n-tuple of polytopes:

$$V_n(P_1, P_2, \ldots, P_n) = \sum_{\mathbf{v}} p_1(\mathbf{v})\ V_{n-1}(\partial_{\mathbf{v}}P_2, \ldots, \partial_{\mathbf{v}}P_n), \tag{5.15}$$

where $\mathbf{v} \in \mathbb{Z}^n$ ranges over all normalized directions, i.e., $\gcd(v_1, v_2, \ldots, v_n) = 1$ and where p_1 denotes the support function of the first polytope P_1. The factor $\frac{1}{n}$ has been left out, so that we have that $V_n(P, P, \ldots, P) = n!\ \mathrm{vol}_n(P)$.

Only for a finite number of normalized directions $\mathbf{v}$, we have that $p_1(\mathbf{v})\ V_{n-1}(\partial_{\mathbf{v}}P_2, \ldots, \partial_{\mathbf{v}}P_n) \neq 0$. If a direction $\mathbf{v}$ is not orthogonal to any edge of a certain polytope P_k, for $k \in \{2, 3, \ldots, n\}$, then $\partial_{\mathbf{v}}P_k$ consists of a single vertex, whence $V_{n-1}(\partial_{\mathbf{v}}P_2, \ldots, \partial_{\mathbf{v}}P_n) = 0$. Based on this idea, we presented in [171] a combinatorial enumeration of all outer normals that contribute to the mixed volume. Algorithm 5.2.5 computes the mixed volume according to (5.15) and consists of two parts. First, for each polytope from $(P_2, \ldots, P_n)$ all pairs of vertices that span edges are computed. Second, we take for each polytope one pair. Each pair of vertices $(\mathbf{a}, \mathbf{b})$ defines one vector $\mathbf{v} = \mathbf{a} - \mathbf{b}$. So we have $n-1$ vectors $\mathbf{v}^{(k)}$, $k = 2, 3, \ldots, n$, where the kth vector stems from a pair of vertices of the kth polytope. By the application of Algorithm 5.2.3, we can generate a normal $\mathbf{v}$. The argument $\mathcal{A}$ of the function Outer_Normal can be used to orientate $\mathbf{v}$ as an outer normal. This has to be iterated for all possible combinations of pairs of vertices. For nonsimplicial polytopes, some directions $\mathbf{v}$ will be generated more than once. Therefore, we maintain a list $\mathcal{V}$ of outer normals of $\mathbf{v}$ for which $p_1(\mathbf{v})V_{n-1}(\partial_{\mathbf{v}} P_2, \ldots, \partial_{\mathbf{v}} P_n) > 0$. The arguments of V_{n-1} have to be computed by the use of a transformation generated by Algorithm 5.2.2, so that the first component of the transformed vertices can be deleted. Analogously to Algorithm 5.2.4, this is denoted by $((U\partial_{\mathbf{v}} P_2)_2^n, \ldots, (U\partial_{\mathbf{v}} P_n)_2^n))$.

Algorithm 5.2.5 Enumerative combinatorial mixed-volume computation.

Input: $\mathcal{A} = (A_1, A_2, \ldots, A_n)$, $A_i \subset \mathbb{Z}^n$, $\#A_i < \infty$.	*spans $\mathcal{P}$: $P_i = \mathrm{conv}(A_i)$*
Output: $V_n(\mathcal{P})$,	*mixed volume*
$\mathcal{V} = \{ \mathbf{v} \mid p_1(\mathbf{v})V_{n-1}(\partial_{\mathbf{v}} P_2, \ldots, \partial_{\mathbf{v}} P_n) > 0 \}$.	*outer normals*
when $n = 1$:	*trivial case*
return $\left(\max\limits_{\mathbf{a}\in A_1} a_1 - \min\limits_{\mathbf{a}\in A_1} a_1\right)$.	*length of interval*
when $n > 1$:	*recursion needed*
$\mathcal{E} := (\emptyset, \emptyset, \ldots, \emptyset)$;	*stage one*
for all $i \in \{2, 3, \ldots, n\}$ loop	*enumerate edges*
$\mathcal{E}_i := \{ (\mathbf{a}, \mathbf{b}) \mid (\mathbf{a}, \mathbf{b})$ spans edge of $P_i \}$;	*all edges of P_i*
end loop;	
$\Sigma := 0$; $\mathcal{V} := \emptyset$;	*stage two*
for all $\mathcal{C} \subset \mathcal{E}$ loop	*enumerate all edge-tuples*
$\mathbf{v} :=$ Outer_Normal$(\mathcal{A}, \mathcal{C})$;	*$\mathcal{C} \subset \partial_{\mathbf{v}}\mathcal{A}$ and $p_1(\mathbf{v}) \geq 0$*
if $\mathbf{v} \notin \mathcal{V}$ and $p_1(\mathbf{v}) > 0$	
then $\Sigma := \Sigma + p_1(\mathbf{v}) \times V_{n-1}((U\partial_{\mathbf{v}} P_2)_2^n,$	*transform and*
$\ldots, (U\partial_{\mathbf{v}} P_n)_2^n)$;	*apply recursion*
$\mathcal{V} := \mathcal{V} \cup \{\mathbf{v}\}$;	*update $\mathcal{V}$*
end if;	
end loop.	*return Σ and $\mathcal{V}$*

Algorithm 5.2.5 can be made much more efficient by applying linear programming methods to avoid the computation of normals that do not contribute to the mixed volume, instead of the plain enumeration of all possible tuples of edge-edge combinations. In particular, the same concepts of Algorithm 5.3.29 can be applied here. Example 5.2.6 shows how the mixed volume can be computed by applying (5.15) for a two-dimensional polynomial system.

Example 5.2.6 (Example 5.1.10 continued.) Consider again the polynomial system displayed in (5.4). In Figure 5.4, we see the Newton polytopes of the system and the outer normals $\mathbf{v}^{(k)}$ necessary for computing the mixed volume.

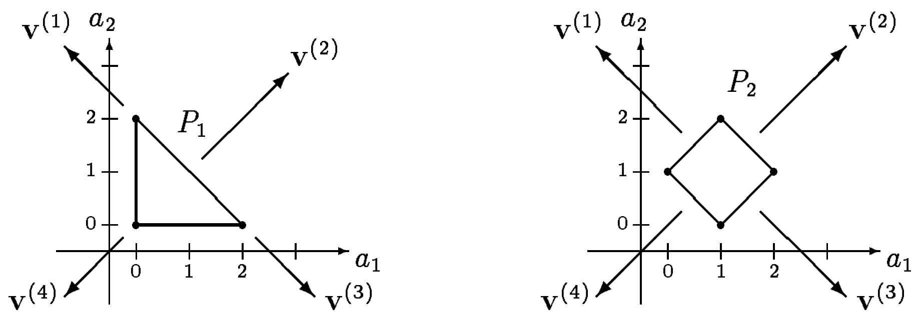


Figure 5.4: Newton polytopes P_1 and P_2, with the outer normals to P_2.

In this case, it is sufficient to consider only the outer normals to the faces of the second polytope P_2.

k	$\mathbf{v}^{(k)}$	$p_1(\mathbf{v}^{(k)})$
1	$[-1,1]^T$	2
2	$[1,1]^T$	2
3	$[1,-1]^T$	2
4	$[-1,-1]^T$	0

Table 5.1: Outer normals $\mathbf{v}^{(k)}$ and $p_1(\mathbf{v}^{(k)})$.

Table 5.1 lists the outer normals $\mathbf{v}^{(k)}$ together with the values $p_1(\mathbf{v}^{(k)})$ for the support function p_1. The length of each transformed edge of the second polytope P_2 equals $V_1(\partial_{\mathbf{v}^{(k)}} P_2) = 1$. Applying (5.15), the mixed volume can be computed as

$$V_2(P_1, P_2) \;=\; 2 \times 1 \;+\; 2 \times 1 \;+\; 2 \times 1 \;+\; 0 \times 1 \;=\; 6, \tag{5.16}$$

which yields the BKK bound of $F(\mathbf{x}) = \mathbf{0}$. ∎

5.2.4 Mixed-homotopy continuation

This section is concerned with the efficient solution of a random coefficient start system $F^{(0)}(\mathbf{x}) = \mathbf{0}$ and thus with the efficient construction of *the coefficient homotopy*, as defined by Morgan and Sommese in [111]. By adding random constants to the equations of the start system, Li, Sauer and Yorke presented in [87] the *cheater's homotopy*, hereby allowing to deal with singular solutions with zero components.

Definition 5.2.7 Let f be a Laurent polynomial. *A random coefficient start polynomial $f^{(0)}$ for f* has the same support as f, but all coefficients are randomly chosen numbers in $\mathbb{C}_0^n$.

Definition 5.2.8 Consider a tuple $F = (f_1, f_2, \ldots, f_n)$ of Laurent polynomials. Then $F^{(0)} = (f_1^{(0)}, f_2^{(0)}, \ldots, f_n^{(0)})$ is *a random coefficient start system for* F, if $f_k^{(0)}$ is a random coefficient start polynomial for f_k, for $k = 1, 2, \ldots, n$.

According to Theorem 5.1.9, $F^{(0)}$ has exactly as many roots as its BKK bound. Like in [9], we say that $F^{(0)}$ is *in general position.*

For the actual computation of all solutions to a system in general position we propose a homotopy continuation method based on the construction Bernshteĭn used in his proof of the theorem. The algorithm is recursive: the resolution of a system of n equations in n unknowns requires the resolution of systems of $n-1$ equations in $n-1$ unknowns. The recursion corresponds to the recursion formula (5.15) for the mixed volume.

Definition 5.2.9 Let $F^{(0)}$ be a random coefficient start system. In order to have a constant term in $f_1^{(0)}$, multiply $f_1^{(0)}$ with the appropriate monomial. Let c denote the constant term of $f_1^{(0)}$. Then, *the mixed homotopy* is defined by the family of systems

$$H(\mathbf{x}, t) = \left\{ \begin{array}{r} f_1^{(0)}(\mathbf{x}) - c + \frac{1}{t}c = 0 \\ f_2^{(0)}(\mathbf{x}) = 0 \\ \vdots \\ f_n^{(0)}(\mathbf{x}) = 0 \quad . \end{array} \right. \tag{5.17}$$

For all but finitely many values of t, the parameterized system (5.17) is in general position. As c is a randomly chosen complex number, the exceptional values of t do not belong to $]0, 1]$.

Now the algorithm for solving the random coefficient start system $F^{(0)}(\mathbf{x}) = \mathbf{0}$ can be described. The output of Algorithm 5.2.5 contains the list $\mathcal{V}$ of all normalized directions $\mathbf{v}$ such that $p_1(\mathbf{v}) V_{n-1}(\partial_{\mathbf{v}} P_2, \ldots, \partial_{\mathbf{v}} P_n) > 0$. For each $\mathbf{v}$ that contributes to the mixed volume, we apply Algorithm 5.2.10 to compute $p_1(\mathbf{v}) V_{n-1}(\partial_{\mathbf{v}} P_2, \ldots, \partial_{\mathbf{v}} P_n)$ solutions of a random coefficient start system $F^{(0)}(\mathbf{x}) = \mathbf{0}$.

Algorithm 5.2.10 Mixed-homotopy continuation for one direction.

Input:	$F^{(0)}(\mathbf{x}) = \mathbf{0}$,	*random coefficient start system*
	$\mathbf{v}$: $p_1(\mathbf{v}) V_{n-1}(\partial_{\mathbf{v}} P_2, \ldots, \partial_{\mathbf{v}} P_n) \neq 0$.	*normalized direction*
Output:	$X_{\mathbf{v}}^* = \{ \mathbf{x}^* \mid F^{(0)}(\mathbf{x}^*) = \mathbf{0} \}$.	$\#X_{\mathbf{v}}^* = p_1(\mathbf{v}) V_{n-1}(\partial_{\mathbf{v}} P_2, \ldots, \partial_{\mathbf{v}} P_n)$

There are four steps:

1. Use Algorithm 5.2.2 to define U. Then the system $H(\mathbf{x}, t)$ can be transformed into

$$H(\mathbf{x} = \mathbf{y}^U, t) = \widetilde{H}(\mathbf{y}, t) = \left\{ \begin{array}{ccccc} \widetilde{f}_{11}^{(0)}(y_2, \ldots, y_n)\, y_1^{p_1(\mathbf{v})} - c & + & \frac{1}{t}c & + & \widetilde{f}_{12}^{(0)}(\mathbf{y}) = 0 \\ \widetilde{f}_{21}^{(0)}(y_2, \ldots, y_n)\, y_1^{p_2(\mathbf{v})} & + & & & \widetilde{f}_{22}^{(0)}(\mathbf{y}) = 0 \\ \vdots & \vdots & & & \vdots \\ \widetilde{f}_{n1}^{(0)}(y_2, \ldots, y_n)\, y_1^{p_n(\mathbf{v})} & + & & & \widetilde{f}_{n2}^{(0)}(\mathbf{y}) = 0 \end{array} \right. \tag{5.18}$$

where $\widetilde{f}_{k1}^{(0)}(y_2, \ldots, y_n) y_1^{p_k(\mathbf{v})} = \partial_{\mathbf{v}} f_k^{(0)}(\mathbf{y}^U)$ and $\deg(\widetilde{f}_{k2}^{(0)}, y_1) < p_k(\mathbf{v})$, $k = 1, 2, \ldots, n$.

2. Substitute $y_1 = \frac{1}{sz_1}$, $y_k = z_k$, $k = 2, \dots, n$, $t = s^{p_1(\mathbf{v})}$ and multiply every equation by $(sz_1)^{p_k(\mathbf{v})}$. Then we have the following homotopy

$$H_{\mathbf{v}}(\mathbf{z}, s) = \begin{cases} \widetilde{f}_{11}^{(0)}(z_2, \dots, z_n) + cz_1^{p_1(\mathbf{v})} + O(s) & = 0 \\ \widetilde{f}_{21}^{(0)}(z_2, \dots, z_n) + O(s) & = 0 \\ \vdots & \vdots \\ \widetilde{f}_{n1}^{(0)}(z_2, \dots, z_n) + O(s) & = 0 \end{cases} \quad . \tag{5.19}$$

3. Substitute $s = 0$ in (5.19). In order to obtain solutions for $z_2, \dots, z_n$, the following $(n-1)$-dimensional system has to be solved

$$\begin{cases} \widetilde{f}_{21}^{(0)}(z_2, \dots, z_n) = 0 \\ \vdots \\ \widetilde{f}_{n1}^{(0)}(z_2, \dots, z_n) = 0 \end{cases} \quad . \tag{5.20}$$

Applying the transformation U on $\partial_{\mathbf{v}} P_k$, the Newton polytope of $\partial_{\mathbf{v}} \widetilde{f}_k^{(0)}$ renders the Newton polytope $U \partial_{\mathbf{v}} P_k$ of $\widetilde{f}_{k1}^{(0)}$, $k = 2, \dots, n$. As $\det(U) = \pm 1$, $\mathrm{vol}_n(U \partial_{\mathbf{v}} P_k) = \mathrm{vol}_n(\partial_{\mathbf{v}} P_k)$ and hence $V_{n-1}(U \partial_{\mathbf{v}} P_2, \dots, U \partial_{\mathbf{v}} P_n) = V_{n-1}(\partial_{\mathbf{v}} P_2, \dots, \partial_{\mathbf{v}} P_n)$.

As $F^{(0)}$ is in general position, so is (5.20), and hence there are exactly $V_{n-1}(\partial_{\mathbf{v}} P_2, \dots, \partial_{\mathbf{v}} P_n)$ regular solutions to (5.20). Because the coefficients of $f_1^{(0)}$ are randomly chosen, $\widetilde{f}_{11}^{(0)}(z_2, \dots, z_n) \neq 0$ for the solutions of (5.20). Solving the first equation of (5.19) yields $p_1(\mathbf{v})$ values for z_1. In this way we have computed $p_1(\mathbf{v}) V_{n-1}(\partial_{\mathbf{v}} P_2, \dots, \partial_{\mathbf{v}} P_n)$ regular solutions of (5.19) for $s = 0$.

4. Follow numerically the solution curves of the homotopy $H_{\mathbf{v}}(\mathbf{z}, s) = \mathbf{0}$ from $s = 0$ to $s = 1$. As $s : 0 \to 1$ corresponds to $t : 0 \to 1$ we have also followed solution curves of $H(\mathbf{x}, t) = \mathbf{0}$ yielding $p_1(\mathbf{v}) V_{n-1}(\partial_{\mathbf{v}} P_2, \dots, \partial_{\mathbf{v}} P_n)$ regular solutions of $H(\mathbf{x}, 1) = \mathbf{0}$.

By applying Algorithm 5.2.10 for all $\mathbf{v} \in \mathcal{V}$, $V_n(\mathcal{P})$ solution curves of $H(\mathbf{x}, t) = \mathbf{0}$ are followed numerically for $t \in]0, 1]$. Provided the followed solution curves are all distinct, all solutions to $F^{(0)}(\mathbf{x}) = \mathbf{0}$ are obtained in this way.

Theorem 5.2.11 *All solution curves defined by the mixed homotopy have a different asymptotic behavior as $t \to 0$.*

Proof. Denote a solution of $\partial_{\mathbf{v}} \widehat{F}^{(0)}(\mathbf{x}, 1) = 0$ by $\mathbf{z}^*$. Let $\mathbf{u}^* = (u_1^* = \frac{1}{z_1^*}, u_2^* = z_2^*, \dots, u_n^* = z_n^*)$. Then we have

$$y_1 \sim \frac{u_1^*}{s}, \quad y_j \sim u_j^*, \text{ as } s \to 0, \text{ for } j = 2, \dots, n. \tag{5.21}$$

By the use of (5.13),

$$x_j = \prod_{i=1}^{n} y_i^{U_{ij}} = y_1^{U_{1j}} \prod_{i=2}^{n} y_i^{U_{ij}}, \; j = 1, 2, \dots, n. \tag{5.22}$$

Then,

$$x_j(s) \sim \left(\frac{u_1^*}{s}\right)^{U_{1j}} \prod_{i=2}^{n} u_i^{*U_{ij}}, \text{ as } s \to 0, \text{ for } j = 2, \ldots, n. \tag{5.23}$$

As $t = s^{p_1(\mathbf{v})}$,

$$x_j(t) \sim t^{-\frac{U_{1j}}{p_1(\mathbf{v})}} \prod_{i=1}^{n} u_i^{*U_{ij}}, \text{ as } t \to 0, \text{ for } j = 2, \ldots, n. \tag{5.24}$$

As U is constructed by Algorithm 5.2.2, its first row equals the (normalized) direction $\mathbf{v}$. This means that the asymptotic behavior of solution curves corresponding to different directions is not the same. Hence, we can be sure that all $V_n(\mathcal{P})$ solutions to the system $F^{(0)}(\mathbf{x}) = \mathbf{0}$ are found by application of the algorithm described above. □

Example 5.2.12 (Example 5.2.6 continued.) Consider again the polynomial system presented in (5.4). The mixed homotopy equals

$$H(\mathbf{x}, t) = \left\{ \begin{array}{r} c_{12}x_1^2 + c_{11}x_2^2 + \frac{1}{t}c_{10} = 0 \\ c_{23}x_1^2x_2 + c_{22}x_1x_2^2 + c_{21}x_1 + c_{20}x_2 = 0 \end{array} \right. . \tag{5.25}$$

All outer normals $\mathbf{v}^{(k)}$ for which $p(\mathbf{v}^{(k)})V_1(\partial_{\mathbf{v}^{(k)}}P_2) \neq 0$ have to be considered, see Table 5.1. For $\mathbf{v}^{(1)} = [-1, 1]^T$, following Algorithm 5.2.10:

1. By application of Algorithm 5.2.2, $U = \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix}$ is obtained. This defines the transformation $\mathbf{x} = \mathbf{y}^U$, i.e., $x_1 = y_1^{-1}y_2$ and $x_2 = y_1$, which transforms $H(\mathbf{x}, t)$ into

$$H(\mathbf{x} = \mathbf{y}^U, t) = \widetilde{H}(\mathbf{y}, t) = \left\{ \begin{array}{r} c_{12}y_1^{-2}y_2^2 + c_{11}y_1^2 + \frac{1}{t}c_{10} = 0 \\ c_{23}y_1^{-1}y_2^2 + c_{22}y_1y_2 + c_{21}y_1^{-1}y_2 + c_{20}y_1 = 0 \end{array} \right. . \tag{5.26}$$

2. Substitute $y_1 = \frac{1}{sz_1}$, $y_2 = z_2$ and $t = s^2$ and multiply the equations respectively with $(sz_1)^2$ and sz_1, because $p_1(\mathbf{v}^{(1)}) = 2$ and $p_2(\mathbf{v}^{(1)}) = 1$. This yields the homotopy

$$H_{\mathbf{v}^{(1)}}(\mathbf{z}, s) = \left\{ \begin{array}{r} c_{12}z_2^2 + c_{11}s^4z_1^4 + c_{10}z_1^2 = 0 \\ c_{23}s^2z_1^2z_2 + c_{22}z_2 + c_{21}s^2z_1^2z_2 + c_{20} = 0 \end{array} \right. . \tag{5.27}$$

3. For $s = 0$, the second equation $\partial_{\mathbf{v}^{(1)}}f_2 = c_{22}z_2 + c_{20} = 0$ yields one solution for z_2, and we have that $V_1(\partial_{\mathbf{v}^{(1)}}P_2) = 1$. Because $p(\mathbf{v}^{(1)}) = 2$, we have for each solution in z_2, two solutions in z_1 for the first equation $c_{12}z_2^2 + c_{10}z_1^2 = 0$. Multiplication yields $p(\mathbf{v}^{(1)})V_1(\partial_{\mathbf{v}^{(1)}}P_2) = 1 \times 2 = 2$. Hence two regular solutions are obtained. They will serve as start solutions in the next step.

4. For $s = 0$ to 1, follow numerically the solution paths of $H_{\mathbf{v}^{(1)}}(\mathbf{z}, s)$. After the appropriate substitutions and transformations, two solutions of $H(\mathbf{x}, 1) = \mathbf{0}$ have been obtained.

The other solutions of $F^{(0)}$ can be found analogously, by considering the outer normals $\mathbf{v}^{(2)}$ and $\mathbf{v}^{(3)}$. ■

5.2.5 Solving simplicial systems

We observe that systems obtained by projecting polynomials are in many cases sparse systems that can be solved very efficiently. To describe the solution procedure we introduce some auxiliary definitions and notations.

Definition 5.2.13 Let $A = \{ \mathbf{a}_l \in \mathbb{Z}^n \mid l = 0, 1, \ldots, n \}$, then a *simplicial system* F with support A is a system of the form $F = (f_1, f_2, \ldots, f_n)$, where

$$f_k = \sum_{l=0}^{n} c_{k\mathbf{a}_l} \mathbf{x}^{\mathbf{a}_l}, \quad \text{with } c_{k\mathbf{a}_l} \in \mathbb{C}, \quad \text{for } k = 1, 2, \ldots, n. \tag{5.28}$$

If all equations in a simplicial system have exactly two monomials, then we call it a *binomial system.*

If the number of different monomials in a Laurent polynomial system is not larger than $n+1$, then solving the system is equivalent to first solving a linear system followed by the solution of a binomial system. More precisely, divide every polynomial f_k in the simplicial system (5.28) by $\mathbf{x}^{\mathbf{a}_0}$ and introduce a vector $\mathbf{y} \in \mathbb{C}^n$ of new variables: $y_l = \mathbf{x}^{\mathbf{a}_l - \mathbf{a}_0}$, for $l = 1, 2, \ldots, n$. The original system $F(\mathbf{x}) = \mathbf{0}$ is now equivalent to a linear system in $\mathbf{y}$, whose solution $\mathbf{y}^*$ can be computed by Gaussian elimination.
Then the following binomial system in the unknowns $x_1, x_2, \ldots, x_n$ has to be solved:

$$\mathbf{x}^{\mathbf{a}_l - \mathbf{a}_0} - y_l^* = 0, \quad \text{for } l = 1, 2, \ldots, n. \tag{5.29}$$

If there is a component $y_l^* = 0$, then there are no solutions in $\mathbb{C}_0^n$. Otherwise, we can compute all solutions by triangulating the matrix of exponents in (5.29), as suggested in the algorithm in the proof of Theorem 5.2.14. By Proposition 5.1.19, we know that the number of isolated solutions of a binomial system exactly equals its mixed volume.

Theorem 5.2.14 *Let $\mathbf{d}_l \in \mathbb{Z}^n$ and $c_l \in \mathbb{C}_0$, $l = 1, 2, \ldots, n$. Consider the binomial system with lth equation*

$$\mathbf{x}^{\mathbf{d}_l} - c_l = 0. \tag{5.30}$$

Denote $M_D = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_n]$. Then the number of nonsingular solutions of the binomial system equals $|\det(M_D)|$.

Proof. By induction, two cases have to be considered:

1. For $n = 1$, $M_D = [d_{11}]$. If $d_{11} = 0$, then there are no regular solutions. Otherwise, de Moivre's rule can be applied in order to obtain all solutions to $x_1^{d_{11}} - c_1 = 0$, if $d_{11} > 0$. For $d_{11} < 0$, $x_1^{-d_{11}} - \frac{1}{c_1} = 0$ has to be solved.

2. For $n > 1$, consider the first equation. Let k be the first position in $\mathbf{d}_1$ with a $d_{1k} \neq 0$. If such a k does not exist, then there are no regular solutions in $\mathbb{C}_0^n$. Otherwise, apply Algorithm 5.2.1 to construct a unimodular matrix U_1. Substitute in (5.30) $\mathbf{x}$ by $\mathbf{y}^{U_1}$. Then, because $U_1 \mathbf{d}_1 = d \mathbf{e}_k$, where $d = \gcd(d_{11}, d_{12}, \ldots, d_{1n})$, by de Moivre's rule we can obtain d values for y_k. Filling in the values for y_k in (5.30), we have to solve d binomial systems of dimension $n-1$ in order to find

all solutions to (5.30). We have $|\det(M_D)| = d \times |\det(U_1 M_D)_{1k}|$, with $(U_1 M_D)_{1k}$ the matrix M_D after multiplication with U_1, where the first column and kth row have been deleted. After the computation of the values for $y_1, \ldots, y_{k-1}, y_{k+1}, \ldots, y_n$, the transformation defined by U_1 has to be applied for transforming the computed solution vectors back into the original coordinate system.

This completes the proof. □

Example 5.2.15 In (5.31), the unimodular transformation U that allows to make the first equation of the binomial system F univariate is applied. See (5.13) for the precise definition of $\mathbf{x} = \mathbf{y}^U$.

$$F(\mathbf{x}) = \left\{ \begin{array}{c} x_1^2 x_2^1 - 1 = 0 \\ x_1^4 x_2^3 - 1 = 0 \end{array} \right. \quad U = \left[\begin{array}{rr} 1 & -1 \\ -1 & 2 \end{array} \right] \quad F(\mathbf{x} = \mathbf{y}^U) = \left\{ \begin{array}{c} y_1 - 1 = 0 \\ y_1 y_2^2 - 1 = 0 \end{array} \right. \tag{5.31}$$

Hence $F(\mathbf{x}) = \mathbf{0}$ has two regular solutions. Note that the total degree $D = 3 \times 7 = 21$ and the 2-homogeneous Bézout bound $B = \mathrm{per}(M_D) = 2 \times 3 + 4 \times 1 = 10$. In general, the permanent of a matrix is always very large compared to its determinant. ■

In [72], solving binomial systems is explained by computing the Smith normal form of the matrix M_D. We can consider the Smith normal form as the discrete analogue of the singular value decomposition, see [1]. Alternatively, as communicated to me by T.Y. Li, following a discussion with him and X. Wang, it is possible to derive an algorithm for solving binomial systems with varying right-hand sides.

5.2.6 Factoring Newton polytopes

In this section we examine the multilinearity of the mixed volume $V_n(\mathcal{P})$ and relate it to the factoring of Newton polytopes. Hereby we indicate how polyhedral homotopies can be combined with product homotopies.

Definition 5.2.16 A polytope P can be *factored* if two polytopes P_1 and P_2 exist, with $\dim(P_1) \geq 1$ and $\dim(P_2) \geq 1$, so that $P = P_1 + P_2$ holds. The polytopes P_1 and P_2 define a *factorization* of the polytope P.

See Figure 5.5 for an example. Note that in general a factorization of a Newton polytope is not unique, as is illustrated in Figure 5.6.

If one of the Newton polytopes in $\mathcal{P}$ can be factored, say $P_1 = P_{11} + P_{12}$, then the mixed volume $V_n(\mathcal{P})$ can be computed (see also Proposition 5.1.8) as

$$V_n(P_{11} + P_{12}, P_2, \ldots, P_n) = V_n(P_{11}, P_2, \ldots, P_n) + V_n(P_{12}, P_2, \ldots, P_n). \tag{5.32}$$

Associate with $\mathcal{P}$ a random coefficient start system $F^{(0)}(\mathbf{x}) = \mathbf{0}$. As P_1 is the Newton polytope of $f_1^{(0)}$, the first polynomial of $F^{(0)}$ can be replaced by the product $f_{11}^{(0)} f_{12}^{(0)}$ of two random coefficient start polynomials $f_{11}^{(0)}$ and $f_{12}^{(0)}$, where $P_{11} = \mathrm{conv}(\mathrm{supp}(f_{11}^{(0)}))$ and $P_{12} = \mathrm{conv}(\mathrm{supp}(f_{12}^{(0)}))$. Solving the start system $F^{(0)}(\mathbf{x}) = \mathbf{0}$ is now reduced to the solution of two smaller systems.

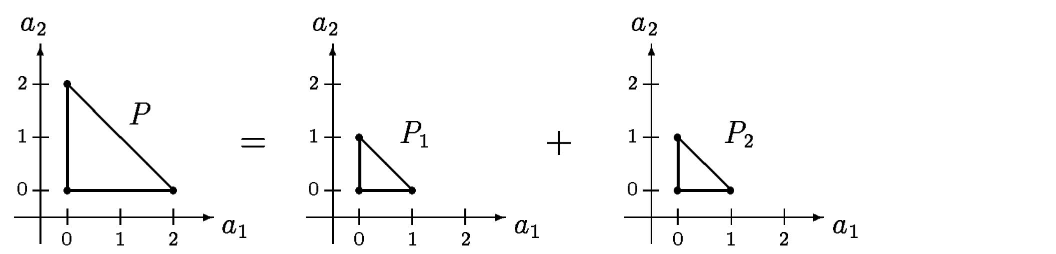


Figure 5.5: Factoring the Newton polytope P of $f = x_1^2 + x_2^2 - 1$ as $P = P_1 + P_2$.

Example 5.2.17 Consider $f(\mathbf{x}) = c_1 x_1^2 + c_2 x_2^2 + c_0$. In Figure 5.5, P is the Newton polytope of f. We have the factorization $P = P_1 + P_2$, as depicted in Figure 5.5.
The polynomials corresponding to P_1 and P_2 are linear. The polynomial corresponding to $P_1 + P_2$ is the following

$$f^{(0)}(\mathbf{x}) = (c_{11}x_1 + c_{12}x_2 + c_{10})(c_{21}x_1 + c_{22}x_2 + c_{20}). \tag{5.33}$$

Because the Newton polytopes of f and $f^{(0)}$ are the same, $f^{(0)}$ can be used as start polynomial instead of f. ■

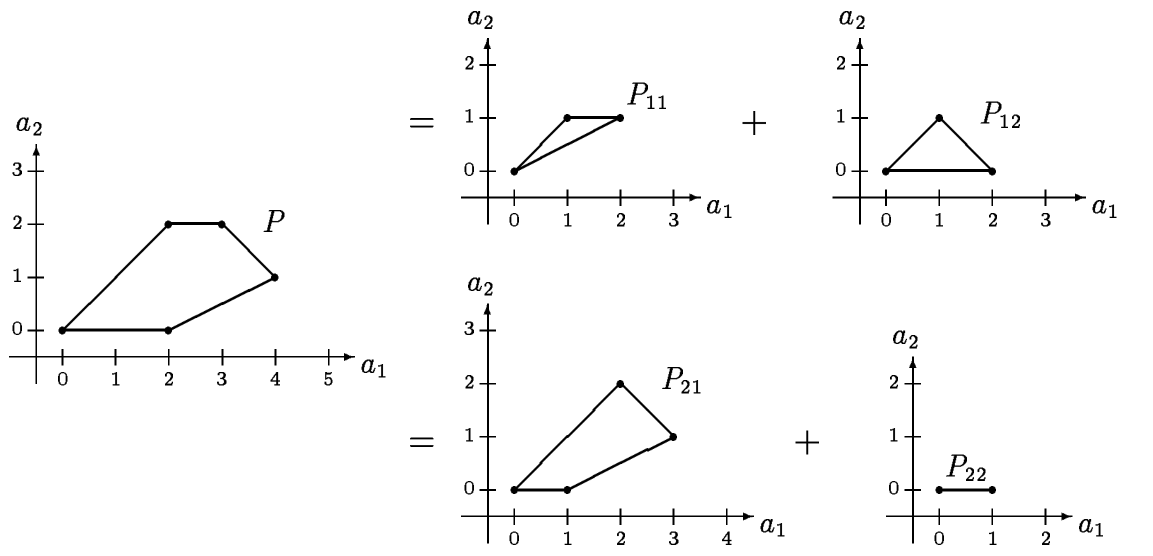


Figure 5.6: A factorization that is not unique: $P = P_{11} + P_{12} = P_{21} + P_{22}$.

Example 5.2.18 Consider $f(\mathbf{x}) = c_1 x_1^2 + c_2 x_1^2 x_2^2 + c_3 x_1^3 x_2^2 + c_4 x_1^4 x_2 + c_0$. The Newton polytope P of f can be factored in two ways, as depicted in Figure 5.6. ■

Consider now the special case where a Newton polytope can be factored so that the polynomials corresponding to the polytopes in the factorization are all linear. Then unknowns can be eliminated by the use of the linear equations, so that we can compute the BKK bound of a system in a lower dimensional space. This is particularly useful as polynomials like $x_1^2 + x_2^2 - 1$, a special case of Example 5.2.17, occur very frequently in kinematics of mechanics, see [104, Chapter 10].

We refer to [142] for recent results on decomposability of polytopes and polyhedra. A additional difficulty is that we are particularly interested in factorizations that are *integer*, i.e., all factors in the sum must be polytopes spanned by integer vertices. Example 5.2.19 illustrates that such a factorization may not always exist.

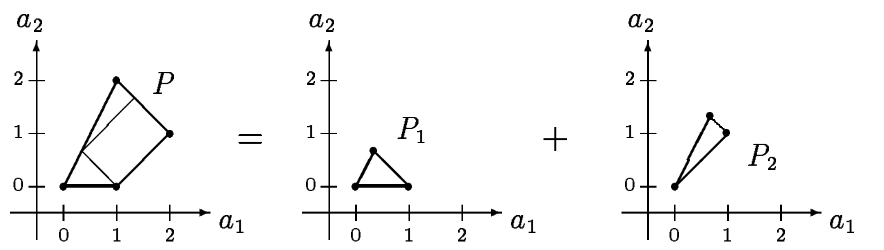


Figure 5.7: A factorization of a Newton polytope P that is not integer.

Example 5.2.19 In [142], Smilansky starts by mentioning the fact that all polygons, except the triangle, are decomposable. Even in this two-dimensional case, it is not true that every Newton polygon, different from a triangle, can be factored in Newton polygons, as illustrated by Figure 5.7. The Newton polygon P can be factored, but there exists no integer factorization for P. ∎

This concept of factoring yields an interpretation of the approach with generalized Bézout numbers described in the previous chapter. The construction of a random product start system $F^{(0)}(\mathbf{x}) = \mathbf{0}$ for a system $F(\mathbf{x}) = \mathbf{0}$ can be interpreted as follows. The Newton polytopes of the polynomials of $F^{(0)}$ contain the Newton polytopes of the polynomials of F and they can be factored so that the polynomials corresponding to the polytopes in the factorization are all linear.

5.3 Static Lifting

In this section we elaborate the lifting method. First we explain how a triangulation of a polytope can be computed by lifting the polytope. Huber and Sturmfels applied in [72] this idea to compute mixed volumes and developed polyhedral continuation. We extended their method by applying it recursively and presented a efficient pruning method for computing the mixed volume. This section concludes with some remarks on the complexity and cost estimates.

5.3.1 Regular triangulations

It is convenient to work with *inner* normals and to define the face $\partial_{\mathbf{v}} P$ of the polytope P by $\partial_{\mathbf{v}} P = \mathrm{conv}(\partial_{\mathbf{v}} A)$, with $\partial_{\mathbf{v}} A = \{\mathbf{x} \in A \mid \langle \mathbf{x}, \mathbf{v} \rangle = \min_{\mathbf{y} \in A} \langle \mathbf{y}, \mathbf{v} \rangle\}$.

Definition 5.3.1 The *lower hull* of a polytope P consists of all facets $\partial_{\mathbf{v}} P$ with $v_n > 0$.

Note that, if we would work with outer normals, then we would consider the upper hull instead. The following definitions are based on the definitions in [72] and in [79]. See [185, Lecture 5] for a more detailed mathematical background.

Definition 5.3.2 Given a set of points $A \subset \mathbb{Z}^n$, a ***subdivision*** S ***of*** A consists of a collection of cells $S = \{C_1, C_2, \ldots, C_m\}$, with $C_k \subset A$, $\forall k$, satisfying

1. $\dim(C_k) = n$;
2. $\mathrm{conv}(C_l) \cap \mathrm{conv}(C_k)$ is a proper face of both $\mathrm{conv}(C_l)$ and $\mathrm{conv}(C_k)$, $l \neq k$;
3. $\cup_{k=1}^{m} \mathrm{conv}(C_k) = \mathrm{conv}(A)$.

See [39, 63] for references on how subdivisions can be computed by means of a lifting function.

Definition 5.3.3 A *lifting function* ω lifts every point of a set A, $\omega: \ A \to \mathbb{Z}: \ \mathbf{a} \mapsto \omega(\mathbf{a})$. This yields the *lifted point* $\widehat{\mathbf{a}} = (\mathbf{a}, \omega(\mathbf{a})) \in \mathbb{Z}^{n+1}$.

The set of lifted points is denoted by $\widehat{A}$, we call it the lifted support. The lifted polytope is denoted by $\widehat{P} = \mathrm{conv}(\widehat{A})$.

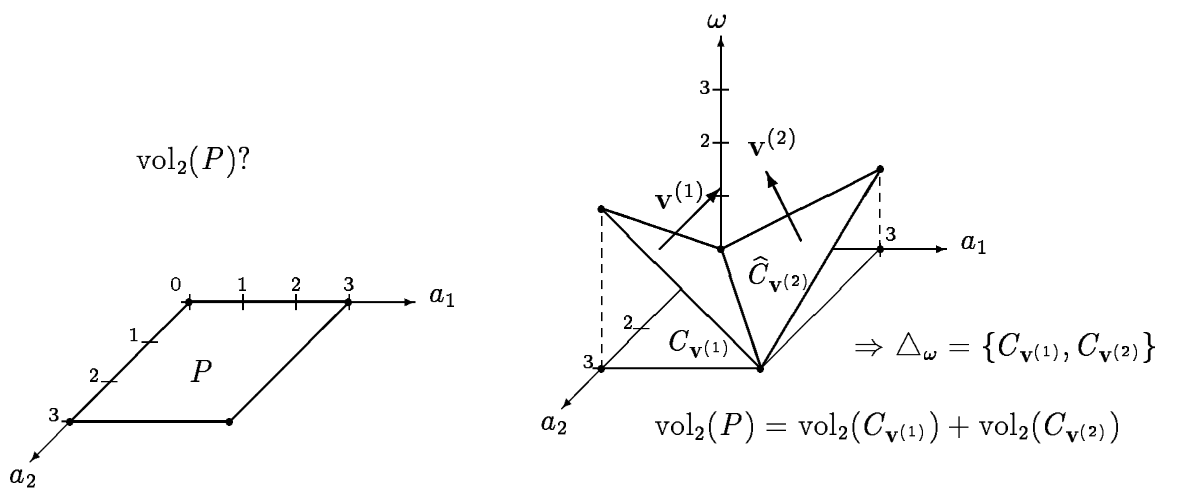


Figure 5.8: The construction of a regular triangulation by lifting.

For every lifting function a subdivision is induced by associating the cells with the projected facets $\partial_{\mathbf{v}} \widehat{P}$ of the lower hull of $\widehat{P}$, $P = \mathrm{conv}(A)$. As each cell C is characterized by the inner normal $\mathbf{v}$ of $\widehat{C} = \partial_{\mathbf{v}} \widehat{A}$, the cell C can be denoted by $C_{\mathbf{v}}$, see Figure 5.8.

Definition 5.3.4 A subdivision $S = \{C_1, C_2, \ldots, C_m\}$ of A is called *a regular subdivision* if there exists a lifting function ω which induces S. It will be denoted by S_ω. The collection of lifted cells $\widehat{C}$ will be denoted by $\widehat{S}$.

Note that in [72] a regular subdivision is called a *coherent* subdivision.

As mentioned in the introduction the usefulness of subdivisions lies in the volume computation. Given a subdivision S of a polytope P, its volume can be computed by

$$\mathrm{vol}_n(P) = \sum_{C \in S} \mathrm{vol}_n(C). \tag{5.34}$$

The computation of the volumes of the cells is straightforward for special subdivisions:

Definition 5.3.5 A subdivision S is called *a triangulation*, and therefore denoted by $\triangle = \{C_1, C_2, \ldots, C_m\}$ if $\#C_j = n+1$, $\forall j$, i.e., $\mathrm{conv}(C_j)$ is an n-dimensional simplex. A triangulation $\triangle$ is said to be *a regular triangulation* if it can be induced by a lifting function ω.

Note that not every triangulation is regular, see [79] for an example.

Each cell C of a triangulation spans a simplex, so we can denote $C = \{\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_n\}$ and

$$\mathrm{vol}_n(C) = \frac{1}{n!} |\det(\mathbf{c}_1 - \mathbf{c}_0, \ldots, \mathbf{c}_n - \mathbf{c}_0)|. \tag{5.35}$$

The triangulations we need are simply meant to compute volumes, not as e.g. Voronoi Diagrams (see [39, 130]) that are used to encode neighborhood relations.

A regular triangulation will be denoted as $\triangle_\omega$. It can be computed by random lifting functions. See [40] for an algorithm to construct a regular triangulation, with the assumption that the lifted points are in general position. In [45], an implementation of this algorithm in three dimensions has been presented.

Proposition 5.3.6 (Huber and Sturmfels) *If the lifting function ω is chosen sufficiently at random, then the induced subdivision is a triangulation. Even for integer lifting functions this holds.*

See [72] for a proof of this proposition. In [43], bounds are given on the size of the integers in order to achieve almost always a triangulation.

5.3.2 Regular simple mixed subdivisions

Theorem 5.3.7 states a fundamental result from Minkowski, see [17] and [22, Chapter 4]. This theorem will lead us to mixed subdivisions needed to compute mixed volumes.

Theorem 5.3.7 (Minkowski) *Let $\mathcal{P} = (P_1, P_2, \ldots, P_n)$ be a tuple of polytopes. Consider the positive linear combination of the polytopes in $\mathcal{P}$: $P = \lambda_1 P_1 + \lambda_2 P_2 + \cdots + \lambda_n P_n$, $\lambda_i \geq 0$, $\forall i$. The volume $\mathrm{vol}_n(P)$ of P is a homogeneous polynomial of degree n with as unknowns the λ_i's and as coefficients the mixed volumes of all possible subsets of $\mathcal{P}$. More precisely,*

$$n!\mathrm{vol}_n(\lambda_1 P_1 + \lambda_2 P_2 + \cdots + \lambda_n P_n) = \sum_{i_1=1}^{n} \sum_{i_2=1}^{n} \cdots \sum_{i_n=1}^{n} V_n(P_{i_1}, P_{i_2}, \ldots, P_{i_n}) \lambda_{i_1} \lambda_{i_2} \cdots \lambda_{i_n}. \tag{5.36}$$

The mixed volume $V_n(\mathcal{P})$ of $\mathcal{P}$ is given by the coefficient of the monomial $\lambda_1 \lambda_2 \cdots \lambda_n$ in the expansion formula (5.36) of $n!\mathrm{vol}_n(P)$.

We call the expansion (5.36) the *Minkowski polynomial of* $\mathcal{P}$. This polynomial can be computed efficiently by means of the Cayley trick, which we will explain in the next chapter.

It is computationally advantageous to exploit the fact that several polynomials have the same support. Denote the number of different supports by r. Moreover, if all supports are equal, then we automatically automatically arrive at the well-known case of one polytope.

Definition 5.3.8 Let $\mathcal{A}$ be a tuple of point sets with respective tuple of polytopes $\mathcal{P}$. Denote by $\mathbf{k} = (k_1, k_2, \ldots, k_r)$, with k_i the number of times the same ith support occurs in $\mathcal{A}$. Assume that the sets in $\mathcal{A}$ are ordered in the following way

$$\begin{array}{rcl} \mathcal{A} & = & (\underbrace{A_1,\ldots,A_1}_{k_1}, \underbrace{A_2,\ldots,A_2}_{k_2}, \ldots, \underbrace{A_r,\ldots,A_r}_{k_r}) \\ \mathcal{P} & = & (\overbrace{P_1,\ldots,P_1}, \overbrace{P_2,\ldots,P_2}, \ldots, \overbrace{P_r,\ldots,P_r}) \end{array} \quad \text{with} \quad \sum_{i=1}^{r} k_i = n. \tag{5.37}$$

The tuples $\mathcal{A}$ and $\mathcal{P}$ are said to be *unmixed* if $r = 1$, *semimixed* if $1 < r < n$ and *fully mixed* if $r = n$.

Let $\mathcal{C} = (C_1, C_2, \ldots, C_r)$, $C_i \subset A_i$, be a cell. The volume of C is written as $\mathrm{vol}_n(C) = \mathrm{vol}_n(\mathrm{conv}(C))$ where the following conventions are used:

$$\mathrm{conv}(C) = \mathrm{conv}(C_1 + C_2 + \cdots + C_r) \subset \mathbb{R}^n, \tag{5.38}$$

$$\mathrm{type}(\mathcal{C}) = (\dim(C_1), \dim(C_2), \ldots, \dim(C_r)) \in \mathbb{N}^r. \tag{5.39}$$

With $\mathcal{C}$ we denote the cell as tuple of subsets of the supports, while C stands for the sum of all subsets in $\mathcal{C}$. Analogously, we denote $A := \sum_{i=1}^r A_i$ and $P := \sum_{i=1}^r P_i$.

Given a tuple of point sets, based on a subdivision of the Minkowski sum, Huber and Sturmfels have extended Definition 5.3.2 in the following way, (see [72]):

Definition 5.3.9 Assume the union of the sets A_i in $\mathcal{A} = (A_1, A_2, \ldots, A_r)$ affinely spans $\mathbb{R}^n$. A *subdivision* of $\mathcal{A}$ is a collection $S = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m\}$ of m cells $\mathcal{C}_j = (C_{j1}, C_{j2}, \ldots, C_{jr})$, $C_{ji} \subset A_i$, satisfying

1. $\dim(C_j) = n$, for $j = 1, \ldots, m$;
2. $\mathrm{conv}(C_j) \cap \mathrm{conv}(C_k)$ is a proper common face of $\mathrm{conv}(C_j)$ *and* $\mathrm{conv}(C_k)$, $j \neq k$;
3. $\cup_{j=1}^m \mathrm{conv}(C_j) = \mathrm{conv}(A)$.

The subdivision is called *mixed* if the additional property

4. $\sum_{i=1}^r \dim(C_{ji}) = n$ for all cells $C_j \in S$ holds.

The subdivision is called *fine mixed* if

5. $\sum_{i=1}^r (\#(C_{ji}) - 1) = n$ for all cells $C_j \in S$.

Note that the fifth item in Definition 5.3.9 represents an additional property, not implied by the previous ones.

Definition 5.3.10 A cell C is called *mixed* if it has a contribution to the mixed volume.

If the subdivision is mixed, then all cells C with $\text{type}(\mathcal{C}) = (k_1, k_2 \ldots, k_r)$ are mixed, see [72]. For the computation of the mixed volume, it is sufficient that the collection of the mixed cells is fine mixed, which is weaker than the last part of Definition 5.3.9.

Definition 5.3.11 A mixed subdivision is called a *simple mixed subdivision* if for all mixed cells C_j: $\sum_{i=1}^{r}(\#(C_{ji}) - 1) = n$.

In Example 5.3.14, a regular simple mixed subdivision is given for which the subdivision is not fine mixed.

Definition 5.3.12 A mixed cell $\mathcal{C}$ that satisfies $\#C_i = k_i + 1, i = 1, \ldots, r$ is called *simple mixed.*

Definition 5.3.13 A subdivision S of a tuple $\mathcal{A} = (A_1, A_2, \ldots, A_r)$ is called *regular* if there exists a tuple ω of lifting functions, $\omega = (\omega_1, \omega_2, \ldots, \omega_r)$, so that the cells of S are the facets of the lower hull of $\sum_{i=1}^{r} \text{conv}(\widehat{A}_i)$.

The lifted tuple of supports is denoted by $\widehat{\mathcal{A}} = (\widehat{A}_1, \widehat{A}_2, \ldots, \widehat{A}_r)$ and the corresponding polytopes by $\widehat{\mathcal{P}} = (\widehat{P}_1, \widehat{P}_2, \ldots, \widehat{P}_r)$. As before, a regular subdivision induced by a lifting ω is denoted by S_ω and the cells by $\mathcal{C}_{\mathbf{v}}$, as they span facets $\partial_{\mathbf{v}}(\sum_{i=1}^{r} \text{conv}(\widehat{A}_i))$ of the lower hull characterized by their inner normal $\mathbf{v}$. The facets spanned by mixed cells are called *mixed facets.*

In [149], a mixed subdivision induced by a lifting function is called a *coherent mixed subdivision (CMD)* and a regular fine mixed subdivision a *tight coherent mixed subdivision (TCMD).* For the relationship with fiber polytopes, see [12] and [185].

Given a mixed subdivision, it is sufficient to consider only the mixed cells, as shown in [72], to compute the mixed volume. Because we take the type of mixture into account, we extend the notation of V_n and vol_n with the additional argument $\mathbf{k} \in \mathbb{N}^r$.

$$V_n(\mathcal{P}, \mathbf{k}) = \sum_{\substack{\mathcal{C} \in S_\omega \\ \text{type}(\mathcal{C}) = \mathbf{k}}} k_1! \cdots k_r! \, \text{vol}_n(C, \mathbf{k}). \tag{5.40}$$

A simple mixed cell $\mathcal{C}$ spans a cube and can be denoted by $\mathcal{C} = (C_1, C_2, \ldots, C_r)$, with $C_i = \{\mathbf{c}_{0i}, \mathbf{c}_{1i}, \ldots, \mathbf{c}_{k_i i}\}$, $i = 1, 2, \ldots, r$. We can compute its volume by

$$\text{vol}_n(C, \mathbf{k}) = \frac{1}{k_1! \cdots k_r!} \left| \det(\mathbf{c}_{11} - \mathbf{c}_{01}, \ldots, \mathbf{c}_{1i} - \mathbf{c}_{0i}, \ldots, \mathbf{c}_{k_i i} - \mathbf{c}_{0i}, \ldots, \mathbf{c}_{k_r r} - \mathbf{c}_{0r}) \right|. \tag{5.41}$$

In case the subdivision is not mixed, we have to apply the lifting method recursively, as will be explained below. Example 5.3.14 illustrates the definitions.

Example 5.3.14 (Example 5.1.10 continued.) Consider the following tuple of lifted point sets $\widehat{\mathcal{A}} = (\widehat{A}_1, \widehat{A}_2)$, with

$$\widehat{A}_1 = \{(0,0,1), (2,0,0), (0,2,1)\} \text{ and } \widehat{A}_2 = \{(1,0,0), (0,1,0), (2,1,0), (1,2,0)\}\,. \tag{5.42}$$

The projection of the polytopes in the lifted tuple $\widehat{\mathcal{P}} = (\widehat{P}_1, \widehat{P}_2)$ and the sum $\widehat{P}_1 + \widehat{P}_2$ are pictured in Figure 5.9.

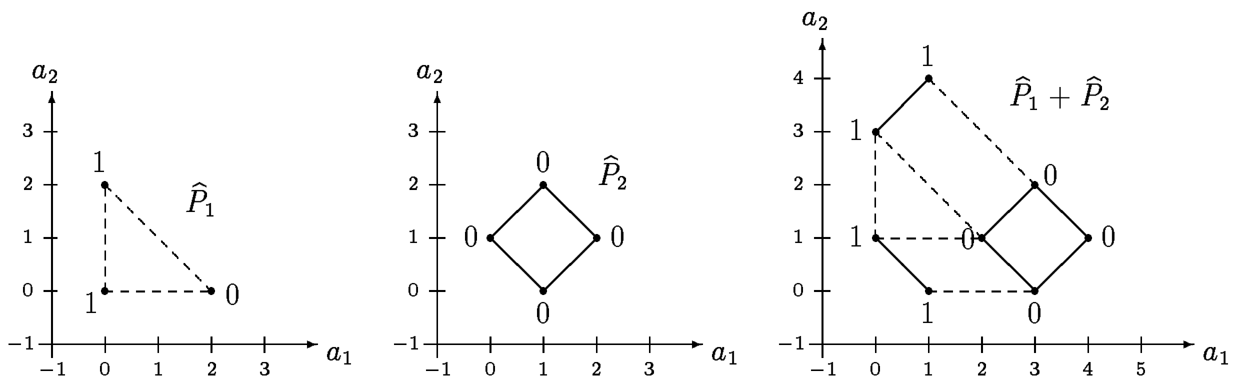


Figure 5.9: The projection of the lifted polytopes $\widehat{P}_1$ and $\widehat{P}_2$ with their sum. The big numbers indicate the values of the lifting function. The mixed cells can be found as parallelograms (with one side dashed and one side solid) in the lower hull of $\widehat{P}_1 + \widehat{P}_2$.

The two mixed cells $\widehat{\mathcal{C}}_{\mathbf{v}^{(1)}}$ and $\widehat{\mathcal{C}}_{\mathbf{v}^{(2)}}$ in the subdivision of $\widehat{\mathcal{A}}$ are

$$\widehat{\mathcal{C}}_{\mathbf{v}^{(1)}} = (\ \{(2,0,0),(0,2,1)\}\ ,\ \{(0,1,0),(1,2,0)\}\ ), \quad \text{with } \mathbf{v}^{(1)} = [1,-1,4]^T \tag{5.43}$$

and

$$\widehat{\mathcal{C}}_{\mathbf{v}^{(2)}} = (\ \{(0,0,1),(2,0,0)\}\ ,\ \{(1,0,0),(0,1,0)\}\ ), \quad \text{with } \mathbf{v}^{(2)} = [1,1,2]^T. \tag{5.44}$$

The other two unmixed cells $\widehat{\mathcal{C}}_{\mathbf{v}^{(3)}}$ and $\widehat{\mathcal{C}}_{\mathbf{v}^{(4)}}$ are

$$\widehat{\mathcal{C}}_{\mathbf{v}^{(3)}} = \Big(\ \widehat{A}_1\ ,\ \{(0,1,0)\}\ \Big), \quad \text{with } \mathbf{v}^{(3)} = [1,0,2]^T \tag{5.45}$$

and

$$\widehat{\mathcal{C}}_{\mathbf{v}^{(4)}} = \Big(\ \{(2,0,0)\}\ ,\ \widehat{A}_2\ \Big), \quad \text{with } \mathbf{v}^{(4)} = [0,0,1]^T. \tag{5.46}$$

This subdivision is not fine mixed because the second component of the last cell $\widehat{\mathcal{C}}_{\mathbf{v}^{(4)}}$ contains too many points. Note that by giving the rightmost point of P_2 a positive lifting value, the subdivision becomes fine mixed. However, we only need to have a simple mixed subdivision and the mixed volume can be computed by

$$V_n(\mathcal{P},(1,1)) \ = \ 1!\cdot 1!\ \frac{1}{1!\cdot 1!}\ \mathrm{vol}_2(C_{\mathbf{v}^{(1)}}) + 1!\cdot 1!\ \frac{1}{1!\cdot 1!}\ \mathrm{vol}_2(C_{\mathbf{v}^{(2)}}) \ = \ 4 + 2 \ = \ 6. \tag{5.47}$$

In Figure 5.9, we can find in the subdivision of $P_1 + P_2$ the unmixed cells as the cells with respectively either all sides dashed or all sides solid.
Their volumes equal respectively $\mathrm{vol}_2(P_1) = \frac{4}{2!}$ and $\mathrm{vol}_2(P_2) = \frac{4}{2!}$. Adding the volumes of all cells yields $\mathrm{vol}_2(P_1 + P_2) = 10$. Note how formula (5.3) can be applied: $V_n(\mathcal{P}) = \mathrm{vol}_2(P_1+P_2) - \mathrm{vol}_2(P_1) - \mathrm{vol}_2(P_2) = 10-2-2 = 6$. The Minkowski polynomial, see (5.36), equals

$$2!\mathrm{vol}_2(\lambda_1 P_1 + \lambda_2 P_2) = 4\lambda_1^2 + 6\lambda_1\lambda_2 + 4\lambda_2^2 \tag{5.48}$$

and summarizes the mixed-volume computation. ■

In [72], Huber and Sturmfels proposed a combinatorial algorithm for constructing a mixed subdivision, based on an idea of Betke, see [10]. We present it here as Algorithm 5.3.15. It consists of two stages. In the first stage, we lift the supports randomly and compute the sets that span all k_i-faces on the lower hull of the lifted ith polytope. During the second stage, we enumerate all combinations of faces to generate the candidate inner normals to the mixed facets on the lower hull of the lifted sum polytope.

Algorithm 5.3.15 The lifting method for computing mixed cells and mixed volumes.

Input: $\mathcal{A} = (A_1, A_2, \ldots, A_r)$, $A_i \subset \mathbb{Z}^n$, $\#A_i < \infty$,	*spans* $\mathcal{P}$: $P_i = \mathrm{conv}(A_i)$
$\mathbf{k} = (k_1, k_2, \ldots, k_r)$, $n = \sum_{i=1}^r k_i$.	*type of mixture*
Output: $\widehat{\mathcal{A}} = (\widehat{A}_1, \widehat{A}_2, \ldots, \widehat{A}_r)$,	*lifted supports*
$\widehat{\mathfrak{S}}_\omega = \{ \, \widehat{C} \in \widehat{S}_\omega \mid \mathrm{vol}_n(C, \mathbf{k}) > 0 \, \}$,	*collection of lifted mixed cells*
$V_n(\mathcal{P}, \mathbf{k})$.	*mixed volume*
$\widehat{\mathcal{F}} := (\emptyset, \emptyset, \ldots, \emptyset)$;	*r-tuple of faces*
for all $i \in \{1, 2, \ldots, r\}$ loop	*stage one*
$\widehat{A}_i := \omega_i(A_i)$;	*random lifting*
$\widehat{\mathcal{F}}_i := \{ \, C = \partial_{\mathbf{v}} \widehat{A}_i \mid \#C = k_i + 1, \; v_{n+1} > 0 \, \}$	*k_i-faces on lower hull of $\widehat{P}_i$*
end loop;	
$\Sigma := 0$; $\widehat{\mathfrak{S}}_\omega := \emptyset$;	*stage two*
for all $\widehat{C} \subset \widehat{\mathcal{F}}$ loop	*make face-face combinations*
$\mathbf{v} :=$ Inner_Normal($\widehat{C}$);	$\mathbf{v} \perp \widehat{C}$, $v_{n+1} > 0$
if $\partial_{\mathbf{v}}(\sum_{i=1}^r \widehat{A}_i) = \widehat{C}$	*$\widehat{C}$ on lower hull?*
then $\widehat{\mathfrak{S}}_\omega := \widehat{\mathfrak{S}}_\omega \cup \widehat{C}$;	*update $\widehat{\mathfrak{S}}_\omega$*
$\Sigma := \Sigma + k_1! \cdots k_r! \mathrm{vol}_n(C, \mathbf{k})$;	*update mixed volume*
end if;	
end loop.	*return $\widehat{\mathcal{A}}$, $\widehat{\mathfrak{S}}_\omega$ and Σ*

Example 5.3.16 (Example 5.3.14 continued.) Consider Figure 5.9. Because the tuple $\mathcal{A}$ is fully mixed, we have to compute first all edges of $\widehat{P}_1$ and $\widehat{P}_2$. During the second stage, there are 3×4 candidates for inner normals that characterize the two mixed cells. In order to compute all candidate inner normals, we have to solve twelve linear systems, which is usually only a tenth of the total amount of linear systems that needs to be solved during the following of one solution path. ■

This combinatorial enumeration of all face-face combinations, as presented in Algorithm 5.3.15 works well for moderate dimensions and for input sets of small size. In the sequel, we will elaborate a more efficient version.

5.3.3 Polyhedral homotopy continuation

As (5.37) in Definition 5.3.8, we assume that the polynomials in F are *ordered according to their supports* $\mathcal{A}$, i.e.:

$$\begin{array}{rcl} \mathcal{A} & = & (\underbrace{A_1, \ldots, A_1}_{k_1}, \quad \underbrace{A_2, \ldots, A_2}_{k_2}, \quad \ldots, \quad \underbrace{A_r, \ldots, A_r}_{k_r}) \\ F & = & (\overbrace{f_1, \ldots, f_{k_1}}, \ \overbrace{f_{k_1+1}, \ldots, f_{k_1+k_2}}, \ \ldots, \ \overbrace{f_{n-k_r+1}, \ldots, f_n}) \end{array} \quad \text{with} \quad \sum_{i=1}^{r} k_i = n. \tag{5.49}$$

To define polyhedral homotopies, we apply lifting functions to Laurent polynomials. For polynomial continuation, *integer lifting* functions have to be used, which we assume in the sequel.

Definition 5.3.17 Let ω be an integer lifted function, defined on the support A of a Laurent polynomial f, denoted by

$$f(\mathbf{x}) = \sum_{\mathbf{a} \in A} c_{\mathbf{a}} \mathbf{x}^{\mathbf{a}}, \quad \text{then} \quad \widehat{f}(\mathbf{x}, t) = \sum_{\mathbf{a} \in A} c_{\mathbf{a}} \mathbf{x}^{\mathbf{a}} t^{\omega(\mathbf{a})} \tag{5.50}$$

is called *the lifted Laurent polynomial of* f. Note that $\mathrm{supp}(\widehat{f}) = \omega(A)$.

Definition 5.3.18 Let ω denote a tuple $\omega = (\omega_1, \omega_2, \ldots, \omega_r)$ of integer lifted functions, defined on supports $\mathcal{A}$ of a tuple F of Laurent polynomials. Assume that F is ordered according to the supports $\mathcal{A}$. Then, *the polyhedral homotopy* $\widehat{F}$ *induced by* ω *on* F is defined by applying the lifting functions ω_i to all polynomials in F with support A_i, for all $i = 1, 2, \ldots, r$. Note that $\widehat{F}(\mathbf{x}, 1) = F(\mathbf{x})$.

As with mixed-homotopy continuation, we assume that the system F is in general position. In fact, we also construct first a random coefficient system $F^{(0)}(\mathbf{x}) = \mathbf{0}$ and use the polyhedral homotopy $\widehat{F}^{(0)}(\mathbf{x}, t) = \mathbf{0}$.

Analogously to a mixed homotopy, a polyhedral homotopy gives rise to a number of start systems. More precisely, to each mixed cell, there corresponds an initial form system and a transformation that allows to trace the solution curves numerically.

Definition 5.3.19 Let $\widehat{F}(\mathbf{x}, t) = \mathbf{0}$ be a polyhedral homotopy with corresponding lifted supports $\widehat{\mathcal{A}}$ and denote a lifted mixed cell with inner normal $\mathbf{v}$ by $\widehat{\mathcal{C}}_{\mathbf{v}}$. Consider the tuple $\partial_{\mathbf{v}} \widehat{F}$ of face polynomials $\partial_{\mathbf{v}} \widehat{F} = (\partial_{\mathbf{v}} \widehat{f}_1, \partial_{\mathbf{v}} \widehat{f}_2, \ldots, \partial_{\mathbf{v}} \widehat{f}_n)$, then *the initial form system* $F_{\mathcal{C}_{\mathbf{v}}}$ is defined by $F_{\mathcal{C}_{\mathbf{v}}}(\mathbf{x}) = \partial_{\mathbf{v}} \widehat{F}(\mathbf{x}, 1)$.

When F is in general position, only the initial form systems corresponding to mixed cells will yield solutions. For simple mixed subdivisions, they will be simplicial systems.

Analogously to Theorem 5.2.11, Huber and Sturmfels prove in [72] that by following the solution curves defined by the polyhedral homotopy $\widehat{F}(\mathbf{x}, t) = \mathbf{0}$, all solutions of $F(\mathbf{x}) = \mathbf{0}$ can be obtained. To this end, they approximate the solutions for $t \approx 0$ by Puiseux series and use the fact that the exponents of the leading terms in the expansion of $\mathbf{x}(t)$ correspond to the components of the inner normal $\mathbf{v}$ to the mixed facet. For an introduction to Puiseux series, see [140] or [154]. See also [20] for the application of Puiseux series to the computation of the asymptotic solutions of systems of equations.

Before performing the polyhedral continuation, we need to transform the polyhedral homotopy, according to the inner normal of the mixed cell we are considering.

Definition 5.3.20 Consider a polyhedral homotopy $\widehat{F}(\mathbf{x},t) = \mathbf{0}$ and a mixed cell $\mathcal{C}_{\mathbf{v}}$ with inner normal $\mathbf{v}$. Then, *the polyhedral homotopy* $\widehat{F}_{\mathbf{v}}(\mathbf{z},s) = \mathbf{0}$ *for* $\mathbf{v}$ is derived from $\widehat{F}$ by scaling the equations in

$$\widehat{F}(\mathbf{z} = \mathbf{x}t^{\frac{\mathbf{v}}{v_{n+1}}}, s = t^{-dv_{n+1}}) = \mathbf{0} \quad \text{with} \quad z_i = x_i t^{\frac{v_i}{v_{n+1}}}, \quad d = \max_{i,j \in \{1,2,\ldots,n\}} \deg(f_i, z_j), \quad (5.51)$$

so that $\widehat{F}_{\mathbf{v}}(\mathbf{z},0) = F_{\mathcal{C}_{\mathbf{v}}}(\mathbf{z})$.

Example 5.3.21 (Example 5.3.14 continued.) For the system $F(\mathbf{x}) = \mathbf{0}$ displayed in (5.4) and with the lifting given in (5.42), the induced polyhedral homotopy is

$$\widehat{F}(\mathbf{x},t) = \left\{ \begin{array}{r} c_{12}x_1^2t^0 + c_{11}x_2^2t^1 + c_{10}t^1 = 0 \\ c_{23}x_1^2x_2t^0 + c_{22}x_1x_2^2t^0 + c_{21}x_1t^0 + c_{20}x_2t^0 = 0 \end{array} \right. , \quad c_{ij} \in \mathbb{C}_0. \quad (5.52)$$

Consider the first mixed cell $\mathcal{C}_{\mathbf{v}^{(1)}}$, displayed in (5.43), which is characterized by $\mathbf{v}^{(1)} = [1,-1,4]^T$. The corresponding transformation is then defined by $x_1 \leftarrow z_1 t^{\frac{1}{4}}$ and $x_2 \leftarrow z_2 t^{\frac{-1}{4}}$. This yields

$$\widehat{F}(\mathbf{z},t) = \left\{ \begin{array}{r} c_{12}z_1^2t^{\frac{1}{2}} + c_{11}z_2^2t^{\frac{1}{2}} + c_{10}t^1 = 0 \\ c_{23}z_1^2z_2t^{\frac{1}{4}} + c_{22}z_1z_2^2t^{\frac{-1}{4}} + c_{21}z_1t^{\frac{1}{4}} + c_{20}z_2t^{\frac{-1}{4}} = 0 \end{array} \right. . \quad (5.53)$$

In (5.52) we substitute $s = t^{\frac{1}{4}}$. After dividing the first equation by s^2 and multiplying the second equation by s, we obtain the homotopy

$$\widehat{F}_{\mathbf{v}^{(1)}}(\mathbf{z},s) = \left\{ \begin{array}{r} c_{12}z_1^2 + c_{11}z_2^2 + c_{10}s^2 = 0 \\ c_{23}z_1^2z_2s^2 + c_{22}z_1z_2^2 + c_{21}z_1s^2 + c_{20}z_2 = 0 \end{array} \right. . \quad (5.54)$$

For $s = 0$, we have a binomial system that has exactly four regular solutions. By following the solution curves of $\widehat{F}_{\mathbf{v}^{(1)}}(\mathbf{z},s) = \mathbf{0}$ from $s = 0$ to 1, four solutions of $F(\mathbf{x}) = \mathbf{0}$ can be reached. To compute the other two solutions, it suffices to construct $\widehat{F}_{\mathbf{v}^{(2)}}(\mathbf{z},s) = \mathbf{0}$ and to trace the two solution curves starting at $s = 0$ up to 1. ∎

In view of easy evaluation of $\widehat{F}_{\mathbf{v}}(\mathbf{x},t)$ and good numeric properties in the beginning of the path tracking, the lifting function ω should be chosen as simple as possible.

The polyhedral continuation has to be applied on a random coefficient start system $F^{(0)}(\mathbf{x}) = \mathbf{0}$, that can then be used in a convex-linear homotopy to obtain all solutions of a target system with the same supports but with a particular choice of the coefficients. In order to avoid this intermediate stage of solving $F^{(0)}$, Huber and Sturmfels proposed in [72] to apply the polyhedral and the convex-linear homotopy at the same time. This yields

$$\widehat{f}_i(\mathbf{x},t) = \sum_{\mathbf{a} \in \operatorname{supp}(f_i)} \left(c_{\mathbf{a}}^{(0)} + (c_{\mathbf{a}} - c_{\mathbf{a}}^{(0)}) \ t\right) \mathbf{x}^{\mathbf{a}} t^{\omega_i(\mathbf{a})}, \quad i = 1,2,\ldots,n, \quad (5.55)$$

which defines another homotopy H with $H(\mathbf{x},1) = F(\mathbf{x})$ such that the Puiseux series leads to initial form systems with the coefficients $c_{\mathbf{a}}^{(0)}$ of $F^{(0)}$.

However, it can often be desirable to know all solutions of $F^{(0)}(\mathbf{x}) = \mathbf{0}$, as it can then be used as start system for a whole family of problems. Secondly, the numerical experience shows that it is better to proceed in two steps. Solving $F^{(0)}(\mathbf{x}) = \mathbf{0}$ can be done with any standard path tracker, as the generic choice of the coefficients implies the polyhedral homotopy to be well-conditioned, whereas several numerical difficulties can occur when it comes to solving $F(\mathbf{x}) = \mathbf{0}$. Such as paths leading to singularities or diverging to infinity, which force the use of special path trackers. Additionally, $F^{(0)}$ may often possess more symmetry properties than F which can be exploited.

5.3.4 Recursively constructed mixed subdivisions

For randomly chosen lifting functions, the subdivision is fine mixed. However, we are restricted to integer lifting functions and to obtain well-conditioned polyhedral homotopies we have to keep the exponents of the continuation parameter moderate. Moreover, this random choice is often prohibitive for the exploitation of symmetry relations as will become clear in Chapter 7. Therefore, Algorithm 5.3.15 will be applied recursively.

As pointed out by Bernd Sturmfels, we can interpret the recursive formula for computing mixed volumes, elaborated in Algorithm 5.2.5, as a lifting method. Indeed, when we consider the mixed homotopy (5.17), we see that it corresponds to giving the origin of the first polytope the lifting value -1, while leaving all the other points unlifted. This explains why we speak now of implicit lifting when we refer to the method used in Algorithm 5.2.5. Example 5.3.22 makes the implicit lifting explicit.

Example 5.3.22 (Example 5.3.14 continued.) Consider again the tuple of points $\mathcal{A}$, but now lifted implicitly, i.e., only $\mathbf{0} \in A_1$ gets a lifting value -1, the other points remain unlifted. The projection of the lifted polytopes in the tuple $\widehat{\mathcal{P}} = (\widehat{P}_1, \widehat{P}_2)$ and the sum $P_1 + P_2$ are pictured in Figure 5.10. The induced subdivision contains two fine mixed cells and one cell that has a nonzero mixed volume but that is spanned by more than four points. In order to compute its mixed volume, it has to be refined further. ■

For a mixed subdivision $S_\omega = \{\mathcal{C}_{\mathbf{v}^{(1)}}, \mathcal{C}_{\mathbf{v}^{(2)}}, \ldots, \mathcal{C}_{\mathbf{v}^{(m)}}\}$, we can compute the mixed volume according to (5.40) as

$$V_n(\mathcal{P}, \mathbf{k}) = \sum_{j=1}^{m} k_1! \cdots k_r! \cdot \mathrm{vol}_n(C_{\mathbf{v}^{(j)}}, \mathbf{k}). \tag{5.56}$$

Only when the subdivision is mixed, we can compute the mixed volume by (5.56), whereas the following formula

$$V_n(\mathcal{P}, \mathbf{k}) = \sum_{j=1}^{m} V_n(\mathcal{C}_{\mathbf{v}^{(j)}}, \mathbf{k}) \tag{5.57}$$

holds for any subdivision S_ω. We can refine S_ω by constructing for each cell $\mathcal{C}_{\mathbf{v}^{(j)}}$ a subdivision $S_\omega^{(j)}$, which is now assumed to be simple mixed (otherwise, apply the recursion again). Then

$$V_n(\mathcal{P}, \mathbf{k}) = \sum_{j=1}^{m} \sum_{\substack{\mathcal{C} \in S_\omega^{(j)} \\ \mathrm{type}(\mathcal{C}) = \mathbf{k}}} k_1! \cdots k_r! \cdot \mathrm{vol}_n(C, \mathbf{k}) \tag{5.58}$$

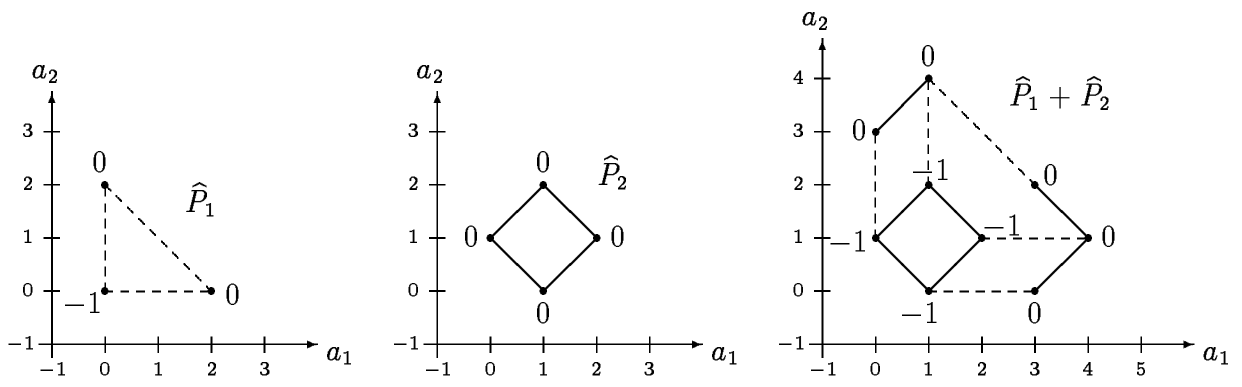


Figure 5.10: Implicit lifting applied to the polytopes P_1 and P_2. The big numbers indicate the values of the lifting function. The lower hull of $\widehat{P}_1 + \widehat{P}_2$ induces a subdivision of $P_1 + P_2$. There are three cells with a nonzero mixed volume.

can be used to compute the mixed volume. By polyhedral homotopy continuation, the initial form systems $F_{\mathcal{C}_{\mathbf{v}(j)}}$ corresponding to the mixed cells $\mathcal{C}_{\mathbf{v}(j)}$ in the subdivision S_ω can be solved, by the use of the simple mixed subdivision $S_\omega^{(j)}$. The solutions of $F_{\mathcal{C}_{\mathbf{v}(j)}}(\mathbf{x}) = \mathbf{0}$ will serve as start solutions in the polyhedral homotopy induced by the subdivision S_ω to solve the original system.

Note that (5.57) generalizes both the recursion formula used in Algorithm 5.2.5 and the nonrecursive approach of Algorithm 5.3.15.

Finally we indicate how to combine product and polyhedral homotopies. If the system contains a lot of terms, then a combination of this method with the lifting method is promising. Apply the product homotopy first on the system as a whole, and then apply the lifting method to the subsystems of the constructed start systems, which are in general much sparser than the original problem. This extends the use of linear-product start systems to more general product start systems, which can yield more performant product homotopies, i.e., with fewer paths to follow.

5.3.5 Normal fans and mixed cells

Powerful tools to investigate the combinatorial structure of polytopes and subdivisions are *fans*. Based on this abstraction, we present a major algorithmic cornerstone of the static lifting algorithm. General definitions can be found in [185, Lecture 7]. Here we reformulate the definitions using our notations. Note that we here work with *outer* normals.

Definition 5.3.23 Let $A \subset \mathbb{Z}^n$, then

$$K(A) = \{\ \mathbf{v} \in \mathbb{Z}^n \mid \forall \mathbf{x}, \mathbf{y} \in A : \langle \mathbf{x}, \mathbf{v} \rangle = \langle \mathbf{y}, \mathbf{v} \rangle\ \} \tag{5.59}$$

is *the normal cone on A*. Let $A \subset B$, then

$$K(A, B) = \{\ \mathbf{v} \in K(A) \mid \forall \mathbf{y} \in B \setminus A, \forall \mathbf{x} \in A : \langle \mathbf{y}, \mathbf{v} \rangle < \langle \mathbf{x}, \mathbf{v} \rangle\ \} \tag{5.60}$$

is *the normal cone on A w.r.t. B*.

Definition 5.3.24 Let $A \subset \mathbb{Z}^n$, then

$$\mathcal{N}^j(A) = \{ \ K(\partial_{\mathbf{v}} A, A) \mid \dim(\partial_{\mathbf{v}} A) = j \ \} \tag{5.61}$$

is *the normal cone complex of the j-faces $\partial_{\mathbf{v}} A$ of* conv(A). If A spans a lifted polytope, then the normal cone complex of the j-faces of the lower hull of conv(A) will be denoted by $\mathcal{N}^j_{\vee}(A)$. The *normal fan* $\mathcal{N}(A)$ of A is the set of all normal cone complexes $\mathcal{N}^j(A)$, for $j = 0, 1, \ldots, n$.

As the definition is given for point sets, it extends naturally to polytopes. We refer to [185, page 193] for an alternative definition of normal fans. An important property of fans is given in Proposition 5.3.25: fans remain invariant when we scale the polytope with a positive factor. In this sense, they characterize the combinatorial structure of the polytope, which is also invariant under scaling.

Proposition 5.3.25 $\mathcal{N}(\lambda A) = \mathcal{N}(A)$, $\forall \lambda > 0$.

Recall that, in order to compute the mixed volume, we need to know the inner normals to the mixed cells in a subdivision. When we scale the polytopes each with a (not necessarily the same) positive factor, these inner normals do not change. As we will see next, we can view the inner normals as defined by the intersection of normal cone complexes.

Definition 5.3.26 Given two normal cone complexes $\mathcal{K}$ and $\mathcal{K}'$. Their *common refinement* is defined as

$$\mathcal{K} \wedge \mathcal{K}' := \{ \ K \cap K' \mid K \in \mathcal{K}, \ K' \in \mathcal{K}' \ \}. \tag{5.62}$$

The *root of a normal cone complex* $\mathcal{K}$ is defined by $\displaystyle\bigcap_{K \in \mathcal{K}} K$.

In [10], Betke proposed a new idea for the computation of mixed volumes based on lifting and normal fans. See also [63, Lemma 2.1.5]. In [136], this idea has been generalized to more than two polytopes. Note that the idea of intersecting normal cones also appeared in [20]. Here we reformulate the idea in the following theorem.

Theorem 5.3.27 (Betke, Schneider) *Given $\widehat{\mathcal{A}} = (\widehat{A}_1, \widehat{A}_2, \ldots, \widehat{A}_r)$ a tuple of lifted point sets, with $\displaystyle\sum_{i=1}^{r} k_i = n$. The normals in the common refinement $\displaystyle\bigwedge_{i=1}^{r} \mathcal{N}^{k_i}_{\vee}(\widehat{A}_i)$ of the normal cone complexes of the k_i-faces of the lower hull of* conv$(\widehat{A}_i)$ *are the outer normals of the facets of the lower hull of* conv$(\widehat{A}_1 + \widehat{A}_2 + \cdots + \widehat{A}_r)$. *These facets determine the mixed cells in the subdivision of $\mathcal{A}$.*

Example 5.3.28 (Example 5.3.16 continued.) Theorem 5.3.27 is illustrated in Figure 5.11, on the same polytopes pictured in Figure 5.9. For the first polytope $\widehat{P}_1$ there are three normal cones on the edges of the lower hull of $\widehat{P}_1$. They are 2-dimensional and have one common 1-dimensional normal cone $K(\widehat{P}_1)$, which is drawn as a point. The square $\widehat{P}_2$ has four edges on its lower hull. The corresponding normal cones are 2-dimensional and share the 1-dimensional normal cone $K(\widehat{P}_2)$ which defines the root, drawn as a point. The intersection points in Figure 5.11 represent the normals that span the cones of $\mathcal{N}^1_{\vee}(\widehat{P}_1) \wedge \mathcal{N}^1_{\vee}(\widehat{P}_2)$. These normals define the mixed cells in the subdivision. ∎

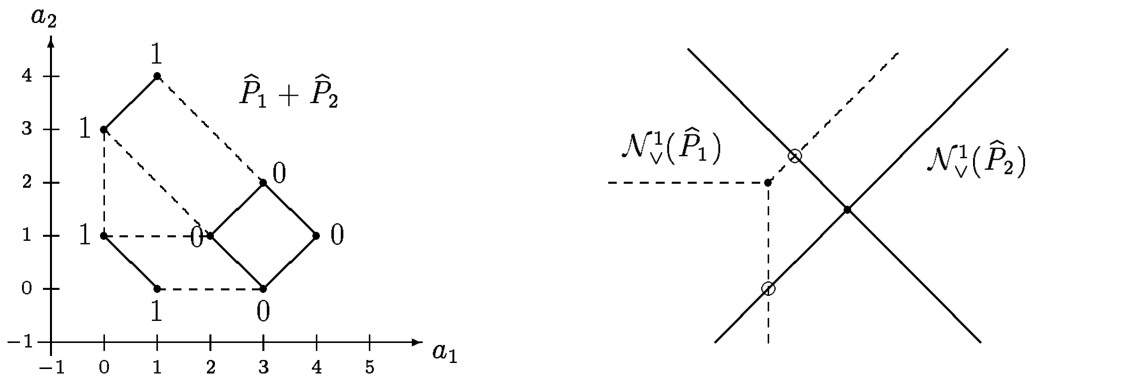


Figure 5.11: On the left, we see the projection of the lower hull of $\widehat{P}_1 + \widehat{P}_2$. On the right, the normal complexes $\mathcal{N}_V^1(\widehat{P}_1)$ and $\mathcal{N}_V^1(\widehat{P}_2)$ are drawn, also projected to the plane. The circles around the intersection points indicate the mixed facets.

In [26, 42, 72], the idea of Betke has been used to compute all mixed cells. Here the algorithm suggested by Theorem 5.3.27, and in [26, 42] described as the lift-and-prune algorithm, will be elaborated.

Given a tuple of lifted point sets $\widehat{\mathcal{A}} = (\widehat{A}_1, \widehat{A}_2, \ldots, \widehat{A}_r)$, any lifted cell $\widehat{\mathcal{C}}_{\mathbf{v}}$ of a regular subdivision can be characterized by its inner normal as

$$\widehat{\mathcal{C}}_{\mathbf{v}} = (\partial_{\mathbf{v}} \widehat{A}_1, \partial_{\mathbf{v}} \widehat{A}_2, \ldots, \partial_{\mathbf{v}} \widehat{A}_r). \tag{5.63}$$

Since $\mathrm{conv}(\widehat{\mathcal{C}}_{\mathbf{v}}) = \mathrm{conv}(\sum_{i=1}^r \partial_{\mathbf{v}} \widehat{A}_i)$ is a facet of the lower hull, the functional $\langle ., \mathbf{v} \rangle$ attains its minimum over $\widehat{A}_i$ at $\partial_{\mathbf{v}} \widehat{A}_i$, i.e.,

$$\forall \widehat{\mathbf{a}}, \widehat{\mathbf{b}} \in \partial_{\mathbf{v}} \widehat{A}_i : \ \langle \widehat{\mathbf{a}}, \mathbf{v} \rangle = \langle \widehat{\mathbf{b}}, \mathbf{v} \rangle, \quad i = 1, 2, \ldots, r, \tag{5.64}$$

$$\forall \widehat{\mathbf{a}} \in \widehat{A}_i \setminus \partial_{\mathbf{v}} \widehat{A}_i, \ \forall \widehat{\mathbf{b}} \in \partial_{\mathbf{v}} \widehat{A}_i : \ \langle \widehat{\mathbf{a}}, \mathbf{v} \rangle > \langle \widehat{\mathbf{b}}, \mathbf{v} \rangle, \quad i = 1, 2, \ldots, r. \tag{5.65}$$

Algorithm 5.3.29 presents a way to compute all mixed cells by searching for feasible solutions to the constraints (5.64) and (5.65). The algorithm enumerates all possible combinations of k_i-faces, with proper feasibility tests to limit the search space. The order of enumeration is organized so that mixed cells which share some faces also share a part of the factorization work to be done to solve the system defined by (5.64).

Algorithm 5.3.29 Shared factorizations subject to inequality constraints:

Input: $(\widehat{A}_1, \widehat{A}_2, \ldots, \widehat{A}_r)$, *lifted point sets*
$\mathbf{k} = (k_1, k_2, \ldots, k_r)$, $n = \sum_{i=1}^r k_i$, *type of mixture*
$(\widehat{\mathcal{F}}_1, \widehat{\mathcal{F}}_2, \ldots, \widehat{\mathcal{F}}_r)$. *$k_i$-faces of lower hull of* $\mathrm{conv}(\widehat{A}_i)$
Output: $\widehat{\mathfrak{S}}_\omega = \{ \, \widehat{C} \in \widehat{S}_\omega \mid V_n(\mathcal{C}, \mathbf{k}) > 0 \, \}$. *collection of lifted mixed cells*

At level i, $1 \leq i < r$:
DATA and INVARIANT CONDITIONS:
(M_1, κ): $M_1 \mathbf{v} = \mathbf{0} \not\Rightarrow v_{n+1} = 0$, $\kappa = \sum_{j=1}^{i-1} k_j$ *equalities* (5.64) *upper triangular up to row κ*
(M_2, κ): $M_2 \mathbf{v} \geq \mathbf{0} \not\Rightarrow -v_{n+1} \geq 0$ *inequalities* (5.65)
$\dim(M_2) = n - \kappa$ *still feasible and reduced*
ALGORITHM:
for each $\widehat{C}_i \in \widehat{\mathcal{F}}_i$ loop *enumerate over all k_i-faces*
Triangulate$(M_1, \kappa, \widehat{C}_i)$; *ensure invariant conditions*
if $M_1 \mathbf{v} = \mathbf{0} \not\Rightarrow v_{n+1} = 0$ *test for feasibility w.r.t.* (5.64)
then *Eliminate*$(M_1, M_2, \kappa, \widehat{C}_i, \widehat{A}_i)$; *eliminate unknowns*
if $M_2 \mathbf{v} \geq \mathbf{0} \not\Rightarrow -v_{n+1} \geq 0$ *test for feasibility w.r.t.* (5.65)
then proceed to next level $i+1$;
end if;
end if;
end for.

At level $i = r$:
Compute $\mathbf{v}$: $M_1 \mathbf{v} = \mathbf{0}$;
Merge the new cell $\mathcal{C}_\mathbf{v}$ with the list $\widehat{\mathfrak{S}}_\omega$.

Note that (5.65) had to be weakened to $\geq$ type inequalities in order to be able to compute also subdivisions that are not mixed. This also explains the merge operation at the end. The feasibility tests in the algorithm allow an efficient computation of the mixed cells.

The conditions (5.64) and (5.65) are verified incrementally. After choosing a k_i-face $\widehat{C}_i = \{\widehat{\mathbf{c}}_{0i}, \widehat{\mathbf{c}}_{1i}, \ldots, \widehat{\mathbf{c}}_{k_i i}\}$ of $\widehat{A}_i$, it is checked whether $(\widehat{C}_1, \ldots, \widehat{C}_i)$ can lead to a mixed cell in the induced subdivision. In the ith step, we already have the upper triangular matrix M_1, with κ rows, originating from the vectors $(\widehat{\mathbf{c}}_{j\nu} - \widehat{\mathbf{c}}_{0\nu})$, $\nu = 1, 2, \ldots, i-1$, $j = 1, 2, \ldots, k_\nu$, with $\kappa = \sum_{\nu=1}^{i-1} k_\nu$. Then this matrix M_1 is extended with new rows defined by the vectors $(\widehat{\mathbf{c}}_{ji} - \widehat{\mathbf{c}}_{0i})$, $j = 1, 2, \ldots, k_i$. By making row combinations and pivoting, we obtain $M_1 := L \cdot M_1$ with entries $M_1 = (m_{kl})$, such that $m_{\mu l} = 0$, $\mu = \kappa+1, \ldots, \kappa+k_i$, $l = 1, 2, \ldots, \kappa$. A unimodular transformation U yields an upper triangular form: $M_1 := U \cdot M_1$. Condition (5.64), with $\nu = 1, 2, \ldots, i$, is then equivalent to $M_1 \mathbf{v} = \mathbf{0}$. If $M_1 \mathbf{v} = \mathbf{0}$ implies $v_{n+1} = 0$, then $\mathrm{conv}(\widehat{C}_1) + \ldots + \mathrm{conv}(\widehat{C}_i)$ lies in a hyperplane perpendicular to the lifting axis and hence $(\widehat{C}_1, \ldots, \widehat{C}_i)$ cannot be part of any mixed cell. This concludes the first feasibility test.

For the second feasibility test, the rows of the matrix M_2 are extended with the vectors $(\widehat{\mathbf{a}}_i - \widehat{\mathbf{c}}_{0i})$, $\widehat{\mathbf{a}}_i \in \widehat{A}_i \setminus \widehat{C}_i$, since in (5.65), it is sufficient to consider only $\widehat{\mathbf{b}} = \widehat{\mathbf{c}}_{0i}$. In the ith step, we use $M_1\mathbf{v} = 0$ to eliminate $\kappa + k_i$ unknowns, which reduces the dimension of the space of inequalities to $n-\kappa-k_i$. If the system of inequalities $M_2\mathbf{v} \geq \mathbf{0}$ implies $-v_{n+1} \geq 0$, then $\mathrm{conv}(\widehat{C}_1) + \cdots + \mathrm{conv}(\widehat{C}_i)$ is not a lower facet of $\mathrm{conv}(\widehat{A}_1 + \cdots + \widehat{A}_i)$ and hence $(\widehat{C}_1, \ldots, \widehat{C}_i)$ cannot be part of any mixed cell. The test whether $M_2\mathbf{v} \geq \mathbf{0} \Rightarrow -v_{n+1} \geq 0$ is equivalent to determining whether the vector $(0, \ldots, 0, -1)$ belongs to the cone spanned by the vectors in the rows of M_2. The Farkas Lemma (see e.g. [185]) deals with this problem and can be worked out by linear programming (see e.g. [46]).

The advantage of this linear optimization model is that we can scale the rows of M_2 independently, so that we deal with unit vectors, which yields us a well-conditioned problem. When all angles of the vectors defined by the rows of M_2 with the vector $(0, \ldots, 0, -1)$ are larger than ninety degrees, then there exists no feasible solution. Otherwise, we construct by using Givens rotations an upper triangular basis for M_2 and lexicographically enumerate all candidate vectors for decomposing $(0, \ldots, 0, -1)$ until a decomposition with positive coefficients has been found. Note that this model can also be used to enumerate all k-faces of a polytope.

For an alternative to these feasibility tests we refer to similar criteria which can be verified by means of linear programming, as presented by Emiris and Canny in [42].

5.3.6 Complexity and cost estimates

Like volume computation (see Section 6.1.6), the problem of mixed-volume computation is known to be $\#P$-hard, see [38, 63]. The analogue result as in the unmixed case from computational convexity is given by the following theorem, proven in [63].

Theorem 5.3.30 (Gritzmann and Klee) *When the dimension n is fixed, the mixed volume of a tuple of polytopes can be computed in polynomial time in the input size.*

The cost of the static lifting method, based on the idea of Betke, has been investigated in great detail in [38] and in [42, 43]. Following these approaches, a bound on the complexity of linear programming is needed. Therefore, we use the fact that Karmarkar's projective algorithm runs in polynomial system, see [62].

Theorem 5.3.31 (Gritzmann and Klee) *Karmarkar's algorithm can be adapted to solve the general linear programming problem in $O(nL)$ steps, where the average step-complexity is $O(n^{\frac{5}{2}}L)$. The required precision is $O(L)$.*

Fixed-precision arithmetic is assumed, so the factor L is left out in the sequel. Recall the following notations: $\widehat{\mathcal{F}}_i$ denotes the k_i-faces of the lower hull of $\widehat{P}_i$ and the collection of mixed cells in the subdivision S_ω is denoted by $\mathfrak{S}_\omega$.

Theorem 5.3.32 *Let $\epsilon_i \in [0,1]$ be the probability of success that a k_i-face in $\widehat{\mathcal{F}}_i$ passes the feasibility test. On average, computing the collection of mixed cells $\mathfrak{S}_\omega$ requires $O(\prod_{i=1}^r (\#\widehat{\mathcal{F}}_i)\epsilon_i[in + (n-i)^{\frac{7}{2}}])$ arithmetical operations.*

Proof. In order to compute $\mathfrak{S}_\omega$ all face-face combinations need to be considered. This explains the product of the cardinalities of the k_i-faces with their respective probabilities.

At level i, it takes time $O(in)$ to perform the elimination step in Algorithm 5.3.29. Applying Theorem 5.3.31, the average time needed to solve a linear programming problem in dimension $n-i$ is given by $O((n-i)^{\frac{7}{2}})$. □

Note the practical importance of this elimination step in Algorithm 5.3.29, as it allows to reduce the dimension of the second feasibility test. This illustrates the importance of sharing the factorizations.

5.4 Applications

The use of polyhedral homotopy continuation has a considerably large range of applications. The polynomial systems considered here are all coming out of the literature. The fact that the systems often have a lower number of finite isolated solutions than the BKK bound is due to specific choices of the coefficients which are in practice never randomly chosen.

5.4.1 The polynomial systems

The first five examples presented can be found in the literature [98, 102, 104, 106, 109, 152, 155]. For each of the examples, the BKK bound has been computed followed by the calculation of the solutions by means of a coefficient homotopy.

A combustion chemistry example

The first application is the Model A combustion chemistry example from Chapter 9 of [104], see also [109], for a temperature of 3000°. Such sparse systems seem to be typical of chemical equilibrium systems.

$$F(\mathbf{x}) = \left\{ \begin{array}{r} x_2 + 2x_6 + x_9 + 2x_{10} - 10^{-5} = 0 \\ x_3 + x_8 - 3.0\ 10^{-5} = 0 \\ x_1 + x_3 + 2x_5 + 2x_8 + x_9 + x_{10} - 5.0\ 10^{-5} = 0 \\ x_4 + 2x_7 - 10^{-5} = 0 \\ 0.5140437\ 10^{-7}\ x_5 - x_1^2 = 0 \\ 0.1006932\ 10^{-6}\ x_6 - x_2^2 = 0 \\ 0.7816278\ 10^{-15}\ x_7 - x_4^2 = 0 \\ 0.1496236\ 10^{-6}\ x_8 - x_1x_3 = 0 \\ 0.6194411\ 10^{-7}\ x_9 - x_1x_2 = 0 \\ 0.2089296\ 10^{-14}\ x_{10} - x_1x_2^2 = 0 \end{array} \right. . \tag{5.66}$$

The total degree equals 96, the BKK bound 16. Before solving this system, first the scaling routine as described in [104, Chapter 5] has to be applied. For this choice of the coefficients, 8 finite solutions can be found.

A chemical equilibrium problem

The following chemical equilibrium problem has been stated in [98]:

$$F(\mathbf{x}) = \begin{cases} x_1x_2 + x_1 - 3x_5 & = & 0 \\ 2x_1x_2 + x_1 + 2R_{10}x_2^2 + x_2x_3^2 + R_7x_2x_3 & & \\ +R_9x_2x_4 + R_8x_2 - Rx_5 & = & 0 \\ 2x_2x_3^2 + R_7x_2x_3 + 2R_5x_3^2 + R_6x_3 - 8x_5 & = & 0 \\ R_9x_2x_4 + 2x_4^2 + 4Rx_5 & = & 0 \\ x_1x_2 + x_1 + R_{10}x_2^2 + x_2x_3^2 + R_7x_2x_3 + R_9x_2x_4 & & \\ +R_8x_2 + R_5x_3^2 + R_6x_3 + x_4^2 - 1 & = & 0 \end{cases} \quad . \tag{5.67}$$

The total degree equals 108, but there are only 4 real and 12 complex solutions. The constants R and R_j can be found in [98]. The BKK bound equals 16.

A system from economics modelling

The following example [104, page 148] came up in economics modelling. The structure of the system enables to formulate the problem for general dimensions n.

$$F(\mathbf{x}) = \begin{cases} \left(x_k + \sum_{l=1}^{n-k-1} x_l x_{l+k} \right) x_n - c_k \; = \; 0, & \\ & k = 1, 2, \ldots, n-1 \\ \sum_{l=1}^{n-1} x_l + 1 = 0 & \end{cases} \quad . \tag{5.68}$$

The constants c_k can be chosen at random. This problem has been solved for dimensions 4,5,6 and 7. See Table 5.2 for characteristics of this family of systems. As the dimension increases with one, the total degree is multiplied with three, while the BKK bound only doubles. The actual number of solutions matches the BKK bound. Note that by the simple transformation $u = \frac{1}{x_n}$, the total degree of the transformed polynomial system equals the BKK bound.

An inverse position problem

The following polynomial system is the inverse position problem for a six-revolute-joint problem in mechanics.

$$F(\mathbf{x}) = \begin{cases} x_l^2 + x_{l+1}^2 - 1 & = & 0, \quad l = 1, \ldots, 4 \\ c_{l1}x_1x_3 + c_{l2}x_1x_4 + c_{l3}x_2x_3 + c_{l4}x_2x_4 & & \\ + c_{l5}x_5x_7 + c_{l6}x_5x_8 + c_{l7}x_6x_7 + c_{l8}x_6x_8 & & \\ + c_{l9}x_1 + c_{l10}x_2 + c_{l11}x_3 + c_{l12}x_4 + c_{l13}x_5 & & \\ + c_{l14}x_6 + c_{l15}x_7 + c_{l16}x_8 + c_{l17} & = & 0, \quad l = 1, \ldots, 4. \end{cases} \tag{5.69}$$

See [152] for a technical paper concerning this problem and [104, Chapter 10], [113, 177] for more introductory publications. See also [91] for the application of a nonlinear homotopy to the parametric description of this problem. The coefficients for this system can be found in [106]. The total degree equals 256, the 2-homogeneous Bézout number 96 and the BKK bound 64. To compute the BKK bound efficiently, random product start polynomials have been constructed for the first 4 polynomials. Note that the Newton polytopes for the last 4 equations are the same. By application of the newly developed algorithms, the computation of 4 singular solutions with multiplicity 8 at infinity has been avoided. For the choice of the coefficients as in [106], the system has 32 finite solutions.

Benchmark i1 from the Interval Arithmetics Benchmarks

The following system is known as Benchmark i1 from the standard benchmarks in interval arithmetic papers, see [155].

$$F(\mathbf{x}) = \left\{ \begin{array}{lllll} x_1 & - & 0.25428722 & - & 0.18324757 x_4 x_3 x_9 & = & 0 \\ x_2 & - & 0.37842197 & - & 0.16275449 x_1 x_{10} x_6 & = & 0 \\ x_3 & - & 0.27162577 & - & 0.16955071 x_1 x_2 x_{10} & = & 0 \\ x_4 & - & 0.19807914 & - & 0.15585316 x_7 x_1 x_6 & = & 0 \\ x_5 & - & 0.44166728 & - & 0.19950920 x_7 x_6 x_3 & = & 0 \\ x_6 & - & 0.14654113 & - & 0.18922793 x_8 x_5 x_{10} & = & 0 \\ x_7 & - & 0.42937161 & - & 0.21180484 x_2 x_5 x_8 & = & 0 \\ x_8 & - & 0.07056438 & - & 0.17081208 x_1 x_7 x_6 & = & 0 \\ x_9 & - & 0.34504906 & - & 0.19612740 x_{10} x_6 x_8 & = & 0 \\ x_{10} & - & 0.42651102 & - & 0.21466544 x_4 x_8 x_1 & = & 0 \end{array} \right. \qquad (5.70)$$

Because the system is very sparse, the mixed volume yields a dramatic improvement in counting the roots. The total degree equals $3^{10} = 59,049$, while the best generalized Bézout bound we could find equals 437. The BKK bound equals 66. There are 50 solutions in $\mathbb{C}_0^{10}$.

Derivation of an optimal multi-dimensional integration formula

The following system has been given by Rabinowitz, see [102], and arose in the derivation of optimal multi-dimensional integration formulae.

$$F(\mathbf{x}) = \left\{ \begin{array}{r} x_1 + x_3 + x_5 + 2x_7 - 1 = 0 \\ x_1 x_2 + x_3 x_4 + 2x_5 x_6 + 2x_7 x_8 + 2x_7 x_9 - \frac{2}{3} = 0 \\ x_1 x_2^2 + x_3 x_4^2 + 2x_5 x_6^2 + 2x_7 x_8^2 + 2x_7 x_9^2 - \frac{2}{5} = 0 \\ x_1 x_2^3 + x_3 x_4^3 + 2x_5 x_6^3 + 2x_7 x_8^3 + 2x_7 x_9^3 - \frac{2}{7} = 0 \\ x_1 x_2^4 + x_3 x_4^4 + 2x_5 x_6^4 + 2x_7 x_8^4 + 2x_7 x_9^4 - \frac{2}{9} = 0 \\ x_5 x_6^2 + 2x_7 x_8 x_9 - \frac{1}{9} = 0 \\ x_5 x_6^4 + 2x_7 x_8^2 x_9^2 - \frac{1}{25} = 0 \\ x_5 x_6^3 + x_7 x_8 x_9^2 + x_7 x_8^2 x_9 - \frac{1}{15} = 0 \\ x_5 x_6^4 + x_7 x_8 x_9^3 + x_7 x_8^3 x_9 - \frac{1}{21} = 0 \end{array} \right. \qquad (5.71)$$

The total degree of this system equals 36,000. A generalized Bézout bound equal to 7090 could be found. The BKK bound equals 136. This system has 16 finite solutions.

5.4.2 Performance

Table 5.2 summarizes all characteristic figures for the applications.

system	description	n	D	B	$V_n(\mathcal{P})$	$N(F)$
(5.66)	combustion 3000°	10	96	44	16	8
(5.67)	chemical equilibrium	5	108	44	16	16
(5.68)	economics modelling	4	18	7	4	4
		5	54	16	8	8
		6	162	36	16	16
		7	486	80	32	32
(5.69)	six-revolute-joint	8	256	96	64	32
(5.70)	benchmark i1	10	59,049	437	66	50
(5.71)	integration formula	9	36,000	7,090	136	16

Table 5.2: Characteristics for the applications. For each system, the dimension n, total degree D, the best Bézout bound B and the mixed volume $V_n(\mathcal{P})$ is listed. The last column contains the number of isolated solutions $N(F)$ of the system F.

We see that in a lot of cases, the BKK bound improves the Bézout bound considerably. The sparser the system, the greater the gap between Bézout and BKK bound. Still, we observe that the mixed volume does not yield an exact root count for all applications.

5.5 Conclusions and future research

The development of polyhedral homotopy continuation has led to a computational breakthrough in solving sparse polynomial systems. We have described in this chapter two different lifting methods: implicit and static lifting. Implicit lifting does not require a random lifting and is inherently recursive. A recursive mechanism can be built into the static lifting algorithm, which is then suitable to exploit the symmetry relations of the application, as we will show in the chapter on symmetric lifting.

A topic for future research is the systematic combination of product with polyhedral homotopies. This should include the development of a priori criteria for deciding when a given bound based on a product structure equals the BKK bound. An algorithmic approach for decomposing Newton polytopes is therefore needed. The conditions whether a given decomposition is possible can be translated into linear optimization problems, that can be handled with linear programming methods.

Efficient linear programming methods for determining whether a partial tuple of supports can be part of a mixed cell will lead to an efficient construction of a mixed subdivision. The development of customized linear optimization problems that can be dealt with efficiently is of practical interest. In particular for symmetric polynomial systems, it can be worthwhile to adapt the standard methods to the symmetric case.

Although for many applications, the mixed volume yields a much sharper bound than the bounds based on Bézout's theorem, still a lot of applications are deficient w.r.t. the BKK bound, i.e., have fewer finite solutions than predicted by the mixed volume. It is necessary to construct special subdivisions that allow to take the deficiency into account.

Chapter 6

Dynamic Lifting

This chapter presents a method we developed for constructing regular subdivisions in order to compute mixed volumes incrementally. Our paper [166] contains most of the material covered by this chapter.

The lifting methods presented in the previous chapter were all static in the sense that all points are lifted first, after which the faces of the lower hull that induce the regular subdivision need to be computed. In the dynamic lifting algorithm, we consider one point at a time and compute the new cells in the subdivision before considering other points. By conservative lifting functions the algorithm encounters no degeneracies that require special care in incremental convex hull constructing algorithms, see [29, 39, 63, 66, 130]. Another main advantage of the new approach is that it provides us with a deterministic manipulation of the lifting for computing mixed volumes, which rules out randomness conditions, assumed in the static lifting method. Furthermore, we obtain an algorithm that performs in polynomial time in the output size, i.e., in the number of cells of the subdivision.

Although the motivation for this method stems from the polyhedral homotopy methods for sparse polynomial systems, our approach is of independent interest. We refer to [63, Section 9] for other applications of volume and mixed-volume computation. The algorithm will be developed from a geometric viewpoint. To have a general notation, we assume the points to be vectors in Euclidean space $\mathbb{E}^n$, equipped with the standard inner product $\langle .,. \rangle$.

The approach presented in this chapter offers a flexible computational tool to investigate which coefficients can have an influence on the number of solutions of the system, see [170]. Another result of dynamic lifting is an algorithm for incrementally solving polynomial systems, which tends to be more stable than the static polyhedral homotopy continuation.

This chapter consists of four parts. The first part is devoted to the case of computing a regular triangulation for one polytope. In the second part, the dynamic lifting algorithm will be generalized to the computation of the mixed volume. The impact on polynomial system solving will be discussed in the third part. In the fourth part, we describe the performance of the algorithm on a list of applications. This chapter concludes with a section summarizing the main properties of the investigated algorithms.

6.1 Dynamic construction of regular triangulations

This section describes the dynamic lifting algorithm applied to one polytope. First, the basic version of the algorithm will be sketched. The following subsections describe the key steps in this algorithm. Some cost estimates are given in the final subsection.

6.1.1 The dynamic lifting algorithm

The basic version of an incremental construction of a regular triangulation is described in Algorithm 6.1.1. There we write $\widehat{\triangle_\omega^{\mathbf{x}}}$ for the set of lifted cells which contain $\mathbf{x}$ in the triangulation for the points that are already processed. The general notation is the following. Let S_ω be a regular subdivision of A and consider $\mathbf{x} \in A$. Denote by $S_\omega^{\mathbf{x}} = \{C \in S_\omega \mid \mathbf{x} \in C\}$. Analogously, $\widehat{S_\omega^{\mathbf{x}}} = \{\widehat{C} \in \widehat{S}_\omega \mid \widehat{\mathbf{x}} \in \widehat{C}\}$.

Algorithm 6.1.1 The dynamic lifting algorithm for one polytope.

Input: A.	*a set of points*
Output: $\triangle_\omega$, ω.	*a regular triangulation of A*
$\triangle_\omega := \{C_0\}$;	*compute an initial cell*
$\omega := 0$;	*with lifting value = 0*
$E := C_0$;	*the points already processed*
$B := \emptyset$;	*the set of interior points*
while $E \neq A$ do	*invariant: $\triangle_\omega$ is a triangulation of E*
$\mathbf{x} \in A \setminus E$;	*choose next point*
$E := E \cup \{\mathbf{x}\}$;	*to be processed*
if $\mathbf{x} \in \triangle_\omega$	$\exists ?\ C \in \triangle_\omega$: $\mathbf{x} \in \mathrm{conv}(C)$
then $B := B \cup \{\mathbf{x}\}$;	*update the set of interior points*
else $\omega := \omega(\widehat{\triangle}_\omega, \mathbf{x})$;	*determine next value*
$\widehat{x}_{n+1} := \omega$;	*of the lifting function*
$\triangle_\omega^{\mathbf{x}} := New\_Facets(\triangle_\omega, \mathbf{x})$;	*compute new cells*
$\triangle_\omega := \triangle_\omega \cup \triangle_\omega^{\mathbf{x}}$;	*update the triangulation*
end if;	
end while;	
$\forall \mathbf{x} \in B$: $\widehat{x}_{n+1} := \omega + 1$.	*lift out the interior points*

There are 5 subalgorithms in Algorithm 6.1.1:

1. Compute an initial cell C_0;
2. Choose the next point $\mathbf{x}$ to be processed $\mathbf{x} \in A \setminus E$;
3. Check whether a point $\mathbf{x}$ already belongs to the triangulation $\mathbf{x} \in \triangle_\omega$, i.e., whether there exists a $C \in \triangle_\omega$ so that $\mathbf{x} \in \mathrm{conv}(C)$;
4. Give a point the lifting value $\omega(\widehat{\triangle}_\omega, \mathbf{x})$;
5. Compute the set of new cells in the triangulation $\triangle_\omega^{\mathbf{x}}$.

The initialization steps for Algorithm 6.1.1 are described in the next section. In order to achieve a simple update of the triangulation, i.e., like $\triangle_\omega := \triangle_\omega \cup \triangle_\omega^{\mathbf{x}}$, it is necessary to apply special lifting functions, defined in Section 6.1.3. In Section 6.1.4, a pivoting mechanism and an efficient data structure will be presented to compute the new cells. To control the height of the lifting, the regular triangulation will be made flat, as described in Section 6.1.5. In the last section, we prove that this algorithm runs in polynomial time in $\#\triangle_\omega$ and in n.

6.1.2 The computation of an initial cell and vertices

At this early stage of computation degeneracy, i.e., $\dim(A) < n$, should be detected. By characterizing the degenerate case, an initial cell can be computed by the greedy algorithm, see [65, page 212]. Algorithm 6.1.2 formulates this algorithm.

Algorithm 6.1.2 The computation of an initial cell.

Input: $A \subset \mathbb{E}^n$.
Output: $C = \{\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_i\}$, a collection of linearly independent points.
If $\dim(A) = n$, then $i = n$, otherwise $i < n$.

1. Let $\mathbf{v} \neq \mathbf{0}$. If $\partial_\mathbf{v} A = \partial_{-\mathbf{v}} A$, then $\dim(A) < n$.
   Otherwise, take $\mathbf{c}_0 \in \partial_\mathbf{v} A$ and $\mathbf{c}_1 \in \partial_{-\mathbf{v}} A$, $\mathbf{c}_1 \neq \mathbf{c}_0$. This yields $C := \{\mathbf{c}_0, \mathbf{c}_1\}$.
2. For i from 2 to n do the following. Let $\mathbf{v} \neq \mathbf{0}$: $\langle \mathbf{c}_j - \mathbf{c}_0, \mathbf{v} \rangle = 0$, for $j = 1, 2, \ldots, i-1$. If $\partial_\mathbf{v} A = \partial_{-\mathbf{v}} A$, then $\dim(A) < n$. Otherwise, if $\langle \partial_\mathbf{v} A, \mathbf{v} \rangle \neq \langle \mathbf{c}_0, \mathbf{v} \rangle$, then take $\mathbf{c}_i \in \partial_\mathbf{v} A$, else take $\mathbf{c}_i \in \partial_{-\mathbf{v}} A$. This yields $C := C \cup \{\mathbf{c}_i\}$.

Except for nongeneric choices of $\mathbf{v}$ the points in C are vertices.

An implementation of Algorithm 6.1.2 will now be described. If $\mathbf{0} \in A$, then we can take $\mathbf{c}_0 := \mathbf{0}$. Then the procedure for computing an initial cell C consists in finding n linearly independent vertices of the point set A. The determination of the directions $\mathbf{v}$'s goes as follows. It is natural to take the first direction along the first coordinate axis. The ith direction $\mathbf{v}$ is not explicitly chosen as in Algorithm 6.1.2, but equivalently a change of coordinates is performed so that the ith unit vector can be chosen as $\mathbf{v}$. The change of coordinates is done by a unimodular transformation U that preserves the first $i-1$ unit vectors and maps $\mathbf{c}_{i-1}$ to the $(i-1)$th unit vector (modulo a scalar multiple). By these orthogonality assumptions, the next point $\mathbf{c}_i$ can be chosen by taking the point with the largest ith component.

Example 6.1.3 Consider the matrix M whose columns contain the points of A. Because $\mathbf{0} \notin A$, let $\mathbf{c}_0 = (1, 0)$ and perform an affine shift with $\mathbf{c}_0$, so that $\mathbf{0} \in A'$. Let $\mathbf{v} = [1, 0]^T$, then $\mathbf{c}_1$ corresponds to the third column in M', which has the largest (underlined) first component. This yields $\mathbf{c}_1 = (4, 2)$.

$$M = \begin{bmatrix} 1 & 3 & 4 & 2 & 0 \\ 0 & 0 & 2 & 4 & 2 \end{bmatrix} \quad \rightarrow \quad M' = \begin{bmatrix} 0 & 2 & \underline{3} & 1 & -1 \\ 0 & 0 & 2 & 4 & 2 \end{bmatrix}. \tag{6.1}$$

The unimodular transformation

$$U = \begin{bmatrix} 1 & -1 \\ -2 & 3 \end{bmatrix} \quad \text{gives} \quad UM' = \begin{bmatrix} 0 & 2 & 1 & -3 & -3 \\ 0 & -4 & 0 & \underline{10} & 8 \end{bmatrix}. \tag{6.2}$$

In Step 2, the maximum w.r.t. $\mathbf{v} = [0,1]^T$ equals 10. Since we maintained the positions of the points in the matrix, the point $\mathbf{c}_2$ of the initial cell is the fourth column of the original matrix M. Because the unimodular transformation is volume preserving and at each step the maximum value along the coordinate axis has been taken (see the underlined entries in the matrices), the cell will have a volume as large as possible. ∎

Note that the steps in Algorithm 6.1.2 can be used for searching an optimal point to be added next. Optimal means that this point has a large contribution to the volume. To achieve this goal approximately, we propose to choose randomly a certain direction and then compute a vertex w.r.t. that direction. This corresponds to the principle of randomized incremental constructions, see [29] and [66].

Once Algorithm 6.1.2 terminates with an initial cell C_0, the volume computation problem is nondegenerate. For an efficient computation of the volume, the nonvertex points of the remainder point set $A \setminus C_0$ can be omitted in advance, as those points will have no influence on the volume. A vertex of a polytope can be considered as the solution of a linear optimization problem, see [138], [65, page 184]. In [64], a feasibility problem is proposed for computing an irredundant representation of a polytope. Let $A = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m\}$, to solve the membership question $\mathbf{x} \in \mathrm{conv}(A)$ we use the following

$$\begin{array}{rl} \min & \mu_0 + \mu_1 + \cdots + \mu_n \\ \text{subject to} & \\ & \left\{ \begin{array}{rcll} \displaystyle\sum_{i=1}^{m} \lambda_i a_{ij} + \mu_j x_j & = & x_j, & j = 1, 2, \ldots, n \\ \displaystyle\sum_{i=1}^{m} \lambda_i + \mu_0 & = & 1 & \\ \lambda_i & \geq & 0, & i = 1, 2, \ldots, m \\ \mu_j & \geq & 0, & j = 0, 1, \ldots, n. \end{array} \right. \end{array} \tag{6.3}$$

It has as trivial feasible but not optimal solution: all $\mu_j = 1$ and all $\lambda_i = 0$. If the problem has an optimal feasible solution, with all $\mu_j = 0$, for $j = 0, 1, \ldots, n$, then $\mathbf{x}$ can be written as a convex combination of the other points in A, the coefficients in this combination are given by the λ_i's and thus $\mathbf{x} \in \mathrm{conv}(A)$.

The vertex set of A can be computed by repetitive application of (6.3), as described in Algorithm 6.1.4.

Algorithm 6.1.4 Computation of the vertex set.

Input: A.	*a set of points*
Output: V.	*the vertex set*
$V := A$;	*initialization*
while $A \neq \emptyset$ do	*enumerate points in A*
choose $\mathbf{x} \in A$;	*next point to be checked*
$A := A \setminus \{\mathbf{x}\}$;	*update A*
if $\mathbf{x} \in \mathrm{conv}(V \setminus \{\mathbf{x}\})$	*to decide, solve* (6.3)
then $V := V \setminus \{\mathbf{x}\}$;	*update V*
end if;	
end while.	

So, an option has been added to Algorithm 6.1.1. We can start with all points or compute the vertices first. In the latter case, the test $\exists ?\ C \in \triangle_\omega$: $\mathbf{x} \in \mathrm{conv}(C)$ in Algorithm 6.1.1 is always false. For the choice of the next point to be processed in the latter case, we observe the following. For $n = 2$ it can be guaranteed that the number of cells in the subdivision will be minimal. For $n > 2$ it is still necessary to choose the point along a random direction to achieve this goal approximately.

6.1.3 Conservative lifting functions

The purpose of this section is to introduce special lifting functions which allow the dynamic construction of a regular triangulation.

Lemma 6.1.5 *Given $\widehat{S}_\omega = \{\widehat{C}_{\mathbf{v}}\}$, consider $\mathbf{x} \not\in \mathrm{conv}(C_{\mathbf{v}})$. Choose a lifting value $\omega(\mathbf{x})$ so that $\langle \widehat{\mathbf{x}}, \mathbf{v} \rangle > \langle \widehat{C}_{\mathbf{v}}, \mathbf{v} \rangle$. Let S'_ω be the induced subdivision of $C_{\mathbf{v}} \cup \{\mathbf{x}\}$, then $\widehat{S}_\omega \subset \widehat{S'_\omega}$. For each new cell $C \in S'_\omega \setminus S_\omega$: $\mathbf{x} \in C$.*

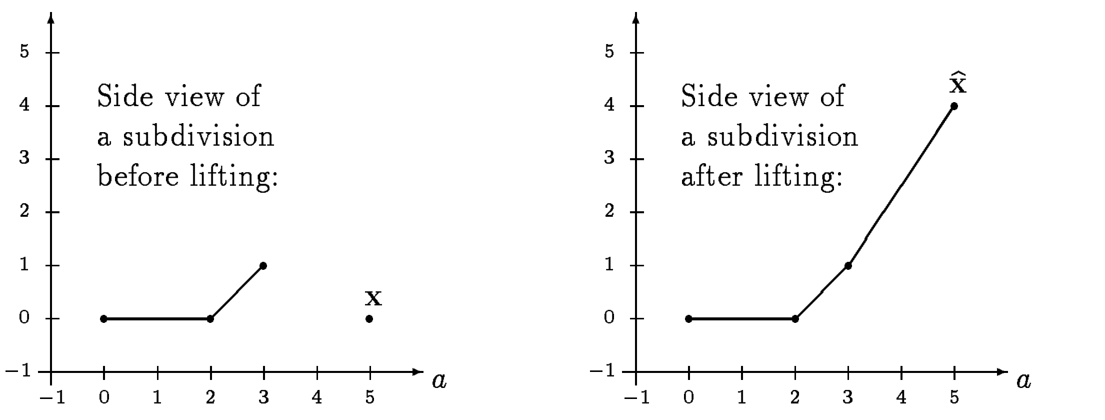


Figure 6.1: A conservative lifting of $\mathbf{x}$: the cells in the subdivision are preserved.

Proof. As $\langle \widehat{\mathbf{x}}, \mathbf{v} \rangle > \langle \widehat{\mathbf{c}}, \mathbf{v} \rangle$, $\forall \widehat{\mathbf{c}} \in \widehat{C}_{\mathbf{v}}$, $\mathbf{v}$ still attains its minimum at $\widehat{C}_{\mathbf{v}}$, so $\widehat{C}_{\mathbf{v}} \in \widehat{S'_\omega}$. Hence $\widehat{S}_\omega \subset \widehat{S'_\omega}$. The second statement is trivial to prove. □

Figure 6.1 illustrates the application of Lemma 6.1.5 and introduces the concept of *conservative lifting.*

Definition 6.1.6 Let $\widehat{C}_{\mathbf{v}}$ be a cell. Consider a point $\mathbf{x}$. Let $\omega(\mathbf{x}) = \widehat{x}_{n+1}$ so that $\langle \widehat{\mathbf{x}}, \mathbf{v} \rangle > \langle \widehat{\mathbf{c}}, \mathbf{v} \rangle$, $\forall \widehat{\mathbf{c}} \in \widehat{C}_{\mathbf{v}}$, then ω is called a *conservative lifting w.r.t.* $\widehat{C}_{\mathbf{v}}$.
Let $\widehat{S}_\omega$ be a regular subdivision. If ω is a conservative lifting w.r.t. each cell $\widehat{C}_{\mathbf{v}} \in \widehat{S}_\omega$, then ω is called a *conservative lifting w.r.t.* $\widehat{S}_\omega$.

This kind of lifting preserves the cells in the subdivision, so it is *conservative.* The advantage of using conservative lifting functions lies in the simplicity of *placing the points* (see [29, 39, 79]) in the triangulation: no deletion or modification of existing cells is required. By using different orders of placing the points, any *placeable triangulation* can be obtained. Note however that not every regular triangulation is placeable, see [79] for an illustration. In [150], triangulations obtained by placing are called *lexicographic triangulations.* Theorem 6.1.7 implies that a regular triangulation can always be maintained by successive applications of a conservative lifting function w.r.t. a regular triangulation, on a point $\mathbf{x}$.

Theorem 6.1.7 *Let S_ω be a regular subdivision of A induced by ω. Consider a point $\mathbf{x}$, lifted conservatively w.r.t. $\widehat{S}_\omega$. Let S'_ω be the regular subdivision of $A \cup \{\mathbf{x}\}$, then $\widehat{S}_\omega \subseteq \widehat{S'_\omega}$. For each new cell $C \in S'_\omega \setminus S_\omega$: $\mathbf{x} \in C$. If $\exists C_{\mathbf{v}} \in S_\omega$: $\mathbf{x} \in \mathrm{conv}(C_{\mathbf{v}})$, then $\widehat{S}_\omega = \widehat{S'_\omega}$.*

Proof. By applying Lemma 6.1.5 successively on all individual cells in the subdivision, the theorem is proven. It is sufficient to see that, for any $\widehat{C}_{\mathbf{v}}$, $\langle \widehat{\mathbf{x}}, \mathbf{v} \rangle > \langle \widehat{C}_{\mathbf{v}}, \mathbf{v} \rangle$. □

The efficient computation of the new cells $\widehat{S'_\omega} \setminus \widehat{S}_\omega$ is the topic of Section 6.1.4.
The last statement of Theorem 6.1.7 shows that a nonvertex point inside the convex hull of one of the cells can be *lifted out.*

An explicit formula for constructing an optimal conservative lifting function is given by the following

Proposition 6.1.8 *Let $\widehat{C}_{\mathbf{v}}$ be a cell. Then the lowest lifting value obtained by a discrete conservative lifting function is given by*

$$\omega(\widehat{C}_{\mathbf{v}}, \mathbf{x}) := \max\left(1\ ,\ \left(\frac{\langle \widehat{C}_{\mathbf{v}}, \mathbf{v} \rangle - \sum_{k=1}^{n} x_k v_k}{v_{n+1}} \right) + 1 \right). \tag{6.4}$$

Given a regular subdivision $\widehat{S}_\omega$. The lowest lifting value obtained by a conservative lifting function is given by

$$\omega(\widehat{S}_\omega, \mathbf{x}) := \max_{\widehat{C}_{\mathbf{v}} \in \widehat{S}_\omega} \omega(\widehat{C}_{\mathbf{v}}, \mathbf{x}). \tag{6.5}$$

The lifting described in Proposition 6.1.8 is the one applied in Figure 6.1.

6.1.4 The computation of new cells by pivoting

The aim of this section is the efficient computation of new cells and to show how a triangulation can always be obtained. It leads to the implementation of the subalgorithm *New_Facets* of Algorithm 6.1.1. We need the following fundamental lemma (Carathéodory, see, e.g., [185]).

Lemma 6.1.9 (Carathéodory) *Let $C = \{\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_n\}$ be a cell, spanning the n-dimensional simplex* $\mathrm{conv}(C)$*. Consider a point* $\mathbf{x}$*. Then*

$$\mathbf{x} \in \mathrm{conv}(C) \;\Leftrightarrow\; \exists \lambda_k : \mathbf{x} = \sum_{k=0}^{n} \lambda_k \mathbf{c}_k, \; \sum_{k=0}^{n} \lambda_k = 1, \; \lambda_k \geq 0, \; k = 0, 1, \ldots, n, \tag{6.6}$$

i.e., $\mathbf{x}$ *can be written as a convex combination of the points in C. Moreover, this representation is independent of the order of the points inside C.*

The computation of the λ_k's can be done by solving a linear system. Consider the set of shifted points $\{\mathbf{c}_1 - \mathbf{c}_0, \ldots, \mathbf{c}_n - \mathbf{c}_0, \mathbf{x} - \mathbf{c}_0\}$ to be the columns of the matrix M. Solve the homogeneous linear system defined by $M\Lambda = \mathbf{0}$. The existence of a solution is guaranteed by $\dim(C) = n$ and the uniqueness is guaranteed by $\#C = n + 1$.

To decide whether $\mathbf{x} \in P$, apply the following:

Theorem 6.1.10 (Carathéodory) *Let $\triangle$ be a triangulation of the polytope P, then $\mathbf{x} \in P \;\Leftrightarrow\; \exists C \in \triangle : \; \mathbf{x} \in \mathrm{conv}(C)$.*

Lemma 6.1.9 gives a simple computable criterion to implement the membership test $\mathbf{x} \in \mathrm{conv}(C)$. As illustrated in Figure 6.2, computing the decomposition of $\mathbf{x}$ w.r.t. the points in C provides additional information indicating which new cells arise. So, it also illustrates the conditions (6.7) in Lemma 6.1.11.

Lemma 6.1.11 *Let $S_\omega = \{C_\mathbf{v}\}$ and $\widehat{C}_\mathbf{v} = \{\widehat{\mathbf{c}}_0, \widehat{\mathbf{c}}_1, \ldots, \widehat{\mathbf{c}}_n\}$. Consider $\widehat{\mathbf{x}}$, $\widehat{\mathbf{x}} = \sum_{k=0}^{n} \lambda_k \widehat{\mathbf{c}}_k$, with $\sum_{k=0}^{n} \lambda_k = 1$ and $\widehat{x}_{n+1}$ so that $\langle \widehat{\mathbf{x}}, \mathbf{v} \rangle > \langle \widehat{C}_\mathbf{v}, \mathbf{v} \rangle$. Let S'_ω be the induced subdivision of $C_\mathbf{v} \cup \{\mathbf{x}\}$. Then the new cells $\widehat{C} \in \widehat{S'_\omega} \setminus \widehat{S}_\omega$ are given by*

$$\{\widehat{\mathbf{c}}_0, \ldots, \widehat{\mathbf{c}}_{k-1}, \widehat{\mathbf{x}}, \widehat{\mathbf{c}}_{k+1}, \ldots, \widehat{\mathbf{c}}_n\} \qquad \textit{where } \lambda_k < 0. \tag{6.7}$$

Proof. First bring the cell $\widehat{C}_\mathbf{v}$ in a diagonal form $\widehat{C}'_\mathbf{v}$, by a shift and a transformation: $\widehat{\mathbf{c}}'_k = T\,(\widehat{\mathbf{c}}_k - \widehat{\mathbf{c}}_0)$, $\widehat{\mathbf{c}}'_k \in \widehat{C}'_\mathbf{v}$: $\widehat{\mathbf{c}}'_0 = \mathbf{0}$ and $\widehat{\mathbf{c}}'_k$ is the kth unit vector, $k = 1, 2, \ldots, n$. Then

$$\begin{aligned} \widehat{\mathbf{x}} = \sum_{k=0}^{n} \lambda_k \widehat{\mathbf{c}}_k \;&\Leftrightarrow\; \widehat{\mathbf{x}} - \widehat{\mathbf{c}}_0 = \sum_{k=0}^{n} \lambda_k \widehat{\mathbf{c}}_k - (\sum_{k=0}^{n} \lambda_k)\widehat{\mathbf{c}}_0, \text{ because } \sum_{k=0}^{n} \lambda_k = 1 \\ &\Leftrightarrow\; \widehat{\mathbf{x}}' = T\,(\widehat{\mathbf{x}} - \widehat{\mathbf{c}}_0) = \sum_{k=0}^{n} \lambda_k T\,(\widehat{\mathbf{c}}_k - \widehat{\mathbf{c}}_0) \\ &\Leftrightarrow\; \lambda_k = \widehat{x}'_k, \; k = 1, 2, \ldots, n, \;\; \lambda_0 = 1 - \sum_{k=1}^{n} \widehat{x}'_k. \end{aligned} \tag{6.8}$$

Due to Lemma 6.1.5 all new cells contain $\mathbf{x}$. Since a cell has $n + 1$ points we need only consider the case that $\mathbf{x}$ replaces one $\mathbf{c}_k$. There are two cases.

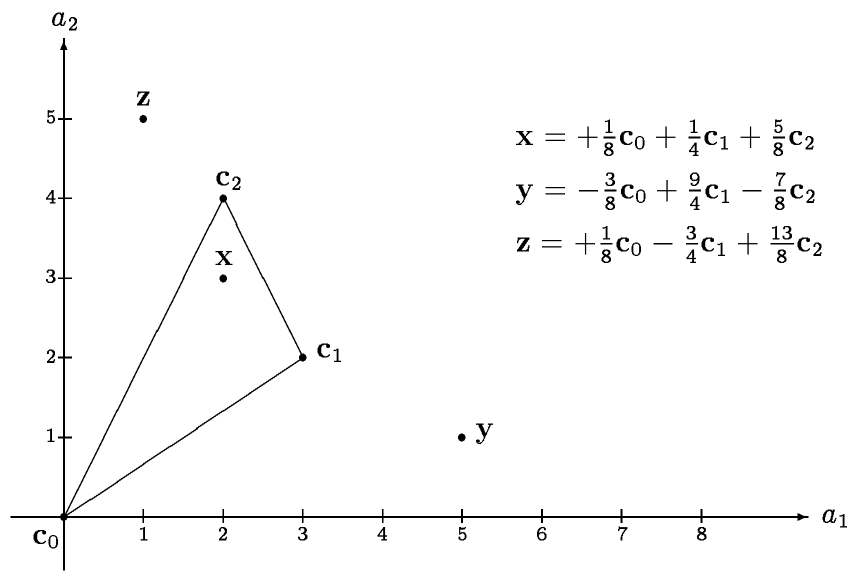


Figure 6.2: Given a simplex defined by $C = \{\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2\}$ and 3 possible point configurations: $\mathbf{x} \in \mathrm{conv}(C)$, adding $\mathbf{y}$ to $\mathrm{conv}(C)$ leads to two new simplices, while for $\mathbf{z}$ there is only one new simplex.

Case 1. $\widehat{\mathbf{x}}'$ *replaces* $\widehat{\mathbf{c}}'_0$:
The inner normal δ of this facet C_δ satisfies

$$
\begin{array}{rcl}
k \neq 0,\ l \neq 0: \ \langle \widehat{\mathbf{c}}'_k, \delta \rangle = \langle \widehat{\mathbf{c}}'_l, \delta \rangle & \Leftrightarrow & \delta_l = \delta_k \\
\langle \widehat{\mathbf{x}}', \delta \rangle = \langle \widehat{\mathbf{c}}'_1, \delta \rangle & \Leftrightarrow & \delta_1 (\sum_{k=1}^{n} \widehat{x}'_k) + \widehat{x}'_{n+1}\delta_{n+1} = \delta_1 \\
& \Leftrightarrow & \delta_1 (1 - \sum_{k=1}^{n} \widehat{x}'_k) = \widehat{x}'_{n+1}\delta_{n+1} \\
& \Leftrightarrow & \lambda_0 = \frac{\widehat{x}'_{n+1}\delta_{n+1}}{\delta_1}.
\end{array}
\tag{6.9}
$$

C_δ is a cell of S'_ω if $\widehat{C}_\delta$ is in the lower hull. The inner normal of a facet in the lower hull satisfies $\delta_{n+1} > 0$ and $\langle \widehat{\mathbf{c}}'_k, \delta \rangle < \langle \widehat{\mathbf{c}}'_0, \delta \rangle \Leftrightarrow \delta_k < 0$, for all $k \neq 0$.
Thus $\{\mathbf{x}, \mathbf{c}_1, \ldots, \mathbf{c}_n\} \in S'_\omega \setminus S_\omega$, if and only if $\lambda_0 < 0$.

Case 2. $\widehat{\mathbf{x}}'$ *replaces* $\widehat{\mathbf{c}}'_k$, $k > 0$:
For the inner normal δ we compute

$$
\begin{array}{rcl}
l \neq k: \ \langle \widehat{\mathbf{c}}'_l, \delta \rangle = \langle \widehat{\mathbf{c}}'_0, \delta \rangle & \Leftrightarrow & \delta_l = \delta_0 \\
\langle \widehat{\mathbf{x}}', \delta \rangle = \langle \widehat{\mathbf{c}}'_0, \delta \rangle & \Leftrightarrow & \widehat{x}'_k \delta_k + \widehat{x}'_{n+1}\delta_{n+1} = 0 \\
& \Leftrightarrow & \lambda_k = \frac{\widehat{x}'_{n+1}\delta_{n+1}}{\delta_k}.
\end{array}
\tag{6.10}
$$

If $\widehat{C}_\delta$ is part of the lower hull, then $\langle \widehat{\mathbf{c}}'_k, \delta \rangle > \langle \widehat{\mathbf{c}}'_0, \delta \rangle \Leftrightarrow \delta_k > 0$.
Thus $\{\mathbf{c}_0, \ldots, \mathbf{c}_{k-1}, \mathbf{x}, \mathbf{c}_{k+1}, \ldots, \mathbf{c}_n\} \in S'_\omega \setminus S_\omega$, if and only if $\lambda_k < 0$. $\square$

Lemma 6.1.11 enables the computation of new cells if the triangulation consists of one simplex. If the triangulation consists of several simplices, then a direct and simple approach for computing $\widehat{\triangle^{\mathbf{x}}_{\omega}}$ is successively applying Lemma 6.1.11 to all cells in the subdivision. Afterwards, it must be checked whether each new cell corresponds to a facet of the lower hull of $\mathrm{conv}(\widehat{A})$. An obvious way to do this is by computing the inner products of their normal with all the other points.

Better ways to avoid the computation of spurious cells use a kind of minimalist data structure, like the one described in [40]. The triangulation is stored as a list of cells. Each cell $C_{\mathbf{v}}$ is characterized by $n+1$ vertices and an inner normal $\mathbf{v}$. For each vertex there is a pointer to the neighboring cell, that can be obtained by replacing that vertex by another one, see Figure 6.3.

Figure 6.3: Four connected cells.

Note that in Figure 6.3 each vertex appears only once and that no null pointers are drawn. The application of Lemma 6.1.9 and Lemma 6.1.11 requires the solution of a linear system $M\Lambda = \mathbf{0}$, defined after Lemma 6.1.9. As this has to be done for each new point to be added to $\triangle$, the factorization matrix for the first n columns of M can be stored. This reduces the solution of a linear system to a back substitution, which requires only $O(n^2)$ operations, whereas the factorization of an n-dimensional matrix requires $O(n^3)$ operations.

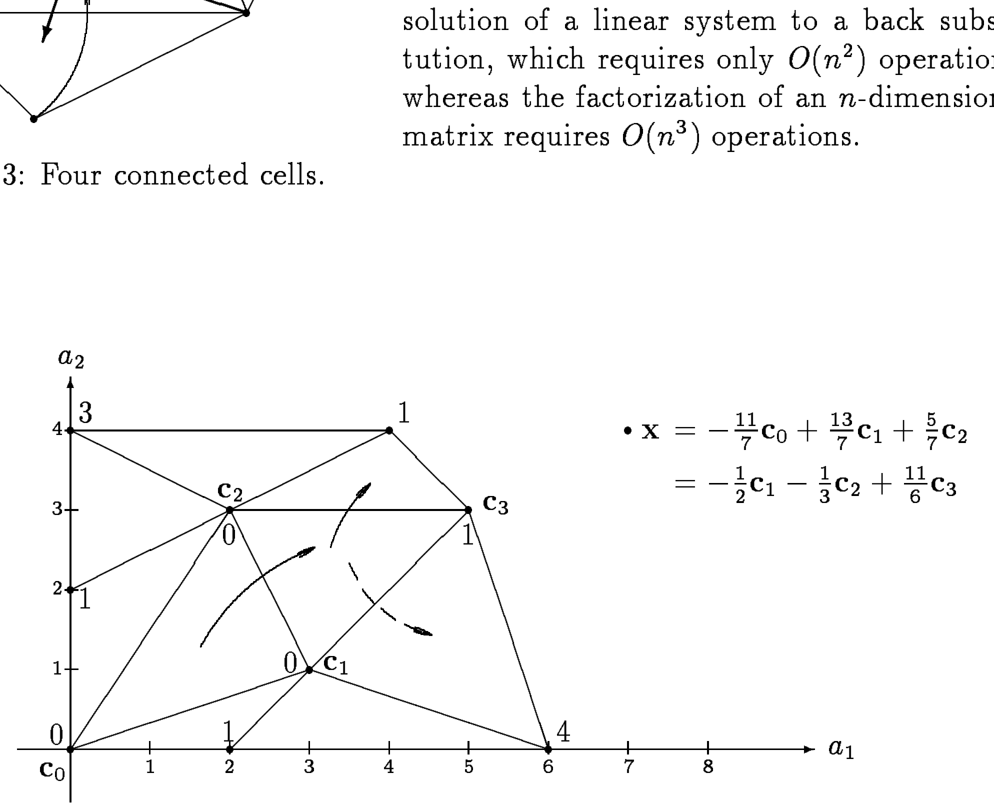


Figure 6.4: An efficient walk from the simplex defined by $C = \{\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2\}$ to the point $\mathbf{x} = (7,4)$. The lifting values are indicated by the big numbers. Only three instead of nine simplices need to be visited. The dashed arrow represents an alternative path.

To decide for a new point $\mathbf{x}$ whether $\mathbf{x} \in \triangle$, Theorem 6.1.10 can be implemented by enumerating all cells C and computing the decomposition of $\mathbf{x}$ w.r.t. the vertices of C. A better way is to exploit the neighborship relation. Starting with one cell C, e.g., the initial cell, we walk towards $\mathbf{x}$ along one chain of cells as pictured in Figure 6.4. We end either at a cell C with $\mathbf{x} \in \mathrm{conv}(C)$ or at a cell close to $\mathbf{x}$. For each cell in the chain we compute the representation $\mathbf{x} = \sum_k \lambda_k \mathbf{c}_k$. If $\mathbf{x} \notin \mathrm{conv}(C)$, then some $\lambda_k < 0$ is chosen and the walk continues at the cell to which the pointer associated to the vertex $\mathbf{x}_k$, corresponding to λ_k, points to. The walk stops when a null pointer associated to a vertex with $\lambda_k < 0$ is encountered.

The data structure can be used for the determination of the new cells in two ways. The walk described above finishes with one cell close to $\mathbf{x}$. With Lemma 6.1.11 one new cell is determined. The neighboring new cells can be determined by considering the neighboring points on the edge of the triangulation and by computing their decomposition w.r.t. this one new cell that contains $\mathbf{x}$. Alternatively, we can enumerate all outer cells, i.e., those with vertices with null pointer, and compute the decomposition $\mathbf{x} = \sum_k \lambda_k \mathbf{c}_k$. The vertices with $\lambda_k < 0$ and null pointer at vertex c_k yield the new cells due to Lemma 6.1.11.

To compare the cost of both alternatives we have to count the number of decompositions that have to be determined. For each cell in the walk such a decomposition is needed. As a conclusion, we can state that the walk will be fruitful when the triangulation contains inner points, like is the case in Figure 6.4. When all nonvertex points have been eliminated in advance, such a walk can be more expensive than the simple enumeration of all outer cells.

6.1.5 Flattening a regular subdivision

Section 6.1.3 provided lower bounds on the lifting for a new point to be added to the triangulation. This section deals with specifying operational upper bounds on the lifting, necessary to counter the following numerical problems. With fixed-point arithmetic, the lifting value might cause an overflow, see Figure 6.1 for how the lifting becomes steeper and steeper. With floating-point arithmetic, the rounding errors that occur while solving an ill-conditioned linear system might produce an erroneous result.

To control the condition of the lower hull $\widehat{A}$, we present the *flattening* of a regular subdivision. This involves a modification of the lifting function and provides an extension of the basic version of the dynamic lifting algorithm, as described in Algorithm 6.1.1.

Definition 6.1.12 A cell $\widehat{C}$ is called *flat*, if all lifting values are zero. A subdivision is called *flat*, if all its cells are flat.

The subdivision induced by a zero-lifting consists of one cell with normal $[0, \ldots, 0, 1]^T$. A subdivision S induced by a conservative lifting function is associated with the zero lifting and this lifting is used for the following steps of the dynamic lifting algorithm. This process is called *flattening* a subdivision.

As the flattening is performed several times we obtain a list of regular triangulations $S_0, S_1, S_2, \ldots$ where S_i is a regular subdivision of the set $A^{(i)}$ so that $\omega^i_{|A^{(i-1)}} \equiv 0$ and S_{i-1} is a subdivision of one cell ($= A^{(i-1)}$) of the subdivision S_i. This can be interpreted as reversing the order of refinement, as defined in [79], and worked out in the recursion applied in the static lifting methods, see Section 5.3.4.

All S_i together form the subdivision which is computed with the pivoting algorithm together with the variant of checking normals. Usually the other variants using the data structure do the same. However, in very special situations these variants compute an even finer subdivision. This detail will be discussed in Section 6.3.2 as well. Note that this final subdivision might not be regular anymore, but because we never need to compute the lifting function that induces this whole subdivision, this is not a problem.

From an implementational point of view the flattening only changes the lifting values, but all old cells remain stored. The pivoting algorithm, presented in the previous section, only requires the lifting to be conservative and has no other demands on the lifting function. So, flattening does not alter the algorithm for computing new cells.

Concerning the flattening mechanism, there is a bound on the maximal lifting value that needs to be set. Each time this bound is exceeded after determining the value of the conservative lifting function for the new point, the whole subdivision will be flattened. For reasons of simplicity, this bound can be set to one. The second alternative is to invoke the flattening automatically, when numerical problems occur during the update. With an exception handling mechanism like in Ada this can be implemented in a straightforward way. The third alternative is to flatten after each update, so that a lifting value equal to one can be taken for each new point.

6.1.6 Complexity and cost estimates

The complexity model used here is the binary Turing machine, see [52] for complexity theory. The complexity of volume and mixed-volume computation is known to be $\#P$-hard, see [37, 38, 63]. This is no surprise, because for a triangulation of the unit cube, we can have that

$$\#\triangle = n! = \mathrm{per}(M), \quad \text{with } M_{ij} = 1, \ \forall i, j. \tag{6.11}$$

So the problem of constructing a triangulation is at least as hard as computing a permanent, which is known to be $\#P$-hard, see Theorem 3.4.1. We refer to [33] for recent results on triangulations of the cube and $(0,1)$-polytopes.

In the theory of computational convexity, see [63], the following important result has been derived.

Theorem 6.1.13 (Gritzmann and Klee) *When the dimension n is fixed, the volume of a polytope can be computed in polynomial time in the input size.*

The idea of this result is based on linear programming, which can be used to derive, in polynomial time, a description of the polytope in terms of an inequality system, representing its facets. Once the facets are given in this representation, by a enumeration of the facets w.r.t. a common point of the polytope, the volume can be computed, straightforwardly by calculating determinants when the polytope is simplicial. Note that it is crucial here to consider n as a constant number.

In this section the cost for the optimal, average and worst case will be measured by counting the number of arithmetical operations in the important steps of the dynamic lifting algorithm. Fixed-point arithmetic is assumed throughout. Evidence is provided that the algorithm runs in polynomial time. The results are interpreted empirically, in Section 6.4.1.

Lemma 6.1.14 *Given a finite set of points $A \subset \mathbb{E}^n$. The computation of an initial cell requires $O(n^3)$ operations.*

Proof. Computing an initial cell is equivalent to computing n linearly independent vectors, which can be brought back to the triangulation of a matrix with n rows. □

For the following, it is assumed that the problem is nondegenerate. Hereby we denote $A' = A \setminus C$, where C is an initial cell.

Lemma 6.1.15 *The cost of checking whether a point belongs to a triangulation $\triangle$ requires at least $O(n^2)$, at most $O((\#\triangle)n^2)$ and on average $O((\#\triangle)n)$ arithmetical operations.*

Proof. As the factorization matrices are stored, only the solving of a triangular system is required for each cell. This requires $O(n^2)$ arithmetical operations for one cell. In the optimal case, already the decomposition (see Lemma 6.1.9) of one cell can suffice, while in the worst case, all cells need to be considered, which takes time $O((\#\triangle)n^2)$. In the average case, assume a uniform distribution of the cells in $\triangle$ and start the walk (see Figure 6.4) at the center of $\triangle$. Each path only goes through one facet of the starting cell. Hence, the path will on average only cross $\frac{1}{n}$th of the cells in $\triangle$, which takes time $O((\#\triangle)n)$. □

Lemma 6.1.16 *The cost of adding a point $\mathbf{x}$ to a triangulation $\triangle$ requires at least $O((\#\triangle^{\mathbf{x}})n^3 + n^2)$, at most $O((\#\triangle^{\mathbf{x}})n^3 + (\#\triangle)n^2)$ and on average $O((\#\triangle^{\mathbf{x}})n^3 + (\#\triangle)n)$ arithmetical operations, with $\triangle^{\mathbf{x}}$ the collection of new cells.*

Proof. Factorization matrices are computed only once, each time a new cell is constructed. So for all cases, time $O((\#\triangle^{\mathbf{x}})n^3)$ is needed. In the optimal case, only one decomposition has to be computed, which takes time $O(n^2)$, while for the worst case, all cells in $\triangle$ need to be considered, which takes time $O((\#\triangle)n^2)$. The average cost bound is derived by application of Lemma 6.1.15. □

Theorem 6.1.17 *The construction of a placeable triangulation $\triangle$ of a point set $A \subset \mathbb{E}^n$ takes at least $O((\#\triangle)n^3 + (\#A')n^2)$, at most $O((\#\triangle)n^3 + (\#A')(\#\triangle)n^2)$ and on average $O((\#\triangle)n^3 + (\#A')(\#\triangle)n)$ arithmetical operations.*

Proof. Factorization matrices are computed only once, each time a new cell is constructed. So for both cases, time $O((\#\triangle)n^3)$ is needed. In the optimal case, at each step, at least one decomposition needs to be computed. Because there are $\#A'$ steps in the algorithm, this takes time $O((\#A')n^2)$. Hence $O((\#\triangle)n^3 + (\#A')n^2)$ is obtained as the cost in the optimal case. In the worst case, at each step, for $j = 1, 2, \ldots, \#A'$, all cells in the corresponding triangulation $\triangle^{(j)}$ need to be considered. Applying Lemma 6.1.16 for each step, a total number of $O(\sum_{j=1}^{\#A'}(\#\triangle^{(j)})n^2)$ arithmetical operations is needed. As $\#\triangle^{(j)} \leq \#\triangle$, for $j = 1, 2, \ldots, \#A'$, $O((\#\triangle)n^3 + (\#A')(\#\triangle)n^2)$ is obtained as a bound for the cost in the worst case. In the average case application of Lemma 6.1.15 yields $O((\#\triangle)n^3 + (\#A')(\#\triangle)n)$. □

Theorem 6.1.17 indicates the bottleneck of the algorithm: determining the cells in $\triangle$ for which the pivoting yields new cells. All cost bounds contain three important factors which influence the general cost of the algorithm. On the input side we have the number of points $\#A'$ and the dimension n of the problem. On the output side the complexity of the facet structure of the lifted polytope plays a role, as the number of cells $\#\triangle$ is also taken into account. This number of cells can be influenced by the choice of the lifting function. It is natural to assume that a random lifting will induce a triangulation with an average number of cells. In the dynamic lifting algorithm a random placeable triangulation is obtained by adding the points in a random order.

Note that an explicit bound on the number of cells $\#\triangle$ is given by $O((\#A)^{\lfloor \frac{n}{2} \rfloor})$, see [130, page 92] for more precise bounds on the number of facets. Still, the bound is polynomial in the output size. The cost w.r.t. the space of the dynamic lifting algorithm is proportional to the number of cells, as the list of cells is maintained during the computations.

6.2 Placeable regular mixed subdivisions

This part deals with the application of the dynamic lifting algorithm to the general case. Algorithm 6.1.1 can be applied in a straightforward manner, by means of the Cayley trick, as explained next. The second subsection deals with the degeneracy check by the computation of an initial mixed cell. An efficient criterion to verify whether a vertex influences the mixed volume is elaborated in the third subsection. Conservative lifting functions are extended to the general case in the fourth subsection. In the fifth subsection, the connectivity of the mixed cells is conjectured and the main kernel of the dynamic lifting algorithm is presented. The computational complexity of the problem and cost estimates of the algorithm are treated in the last section of this part.

6.2.1 The Cayley trick

In this section, Algorithm 6.2.1 presents a geometric description of the so-called *Cayley trick*. This geometric description is due to Sturmfels, as mentioned in [69]. See also [56, Proposition 1.7, page 274].

Algorithm 6.2.1 The Cayley trick.

Input: $\mathcal{A} = (A_1, A_2, \ldots, A_r)$, *a tuple of point sets*
$\mathbf{k} \in \mathbb{N}^r$, $\sum_{i=1}^r k_i = n$. *type of mixture*
Output: $\triangle_\omega$. *a regular fine mixed subdivision of $\mathcal{A}$*

$\widetilde{A}_i := \left\{ \left(\begin{array}{c} \mathbf{a} \\ \mathbf{e}_{i-1} \end{array} \right) \middle| \ \mathbf{a} \in A_i \right\}$; *construct extended point sets with $\mathbf{e}_0 = \mathbf{0}$ and $\mathbf{e}_i = i$th unit vector in $\mathbb{E}^{r-1}$*

$\lhd_\omega := \mathrm{Triang}(\mathrm{conv}(\bigcup_{i=1}^r \widetilde{A}_i))$; *construct a regular triangulation*

$\triangle_\omega := \{ C \mid \exists \widetilde{C} \in \lhd_\omega : \pi(\widetilde{C}) = C \}$. *$\pi$ projects to original coordinates*

In Figure 6.5, an example for the first step of Algorithm 6.2.1 is given.

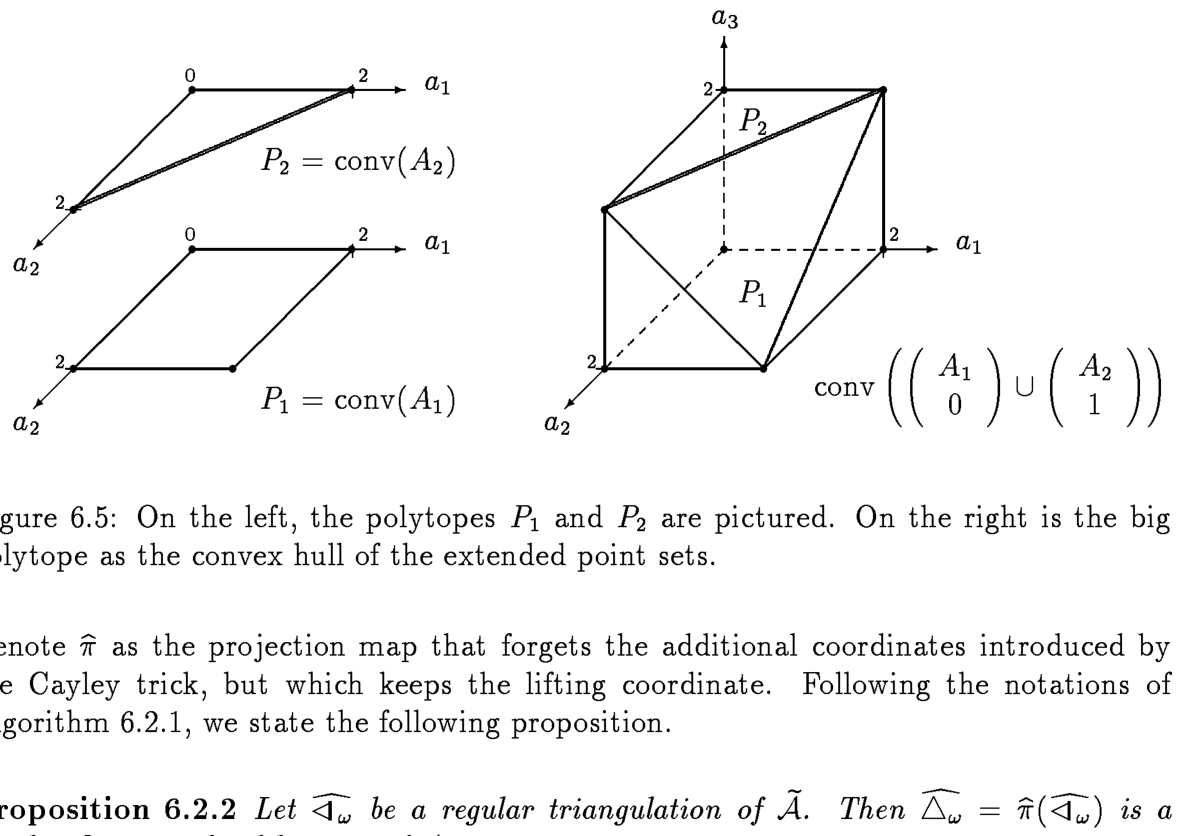


Figure 6.5: On the left, the polytopes P_1 and P_2 are pictured. On the right is the big polytope as the convex hull of the extended point sets.

Denote $\widehat{\pi}$ as the projection map that forgets the additional coordinates introduced by the Cayley trick, but which keeps the lifting coordinate. Following the notations of Algorithm 6.2.1, we state the following proposition.

Proposition 6.2.2 *Let $\widehat{\lhd_\omega}$ be a regular triangulation of $\widetilde{\mathcal{A}}$. Then $\widehat{\triangle_\omega} = \widehat{\pi}(\widehat{\lhd_\omega})$ is a regular fine mixed subdivision of $\mathcal{A}$.*

Proof. The projection $\widehat{\pi}$ copies the lifting ω on $\widetilde{\mathcal{A}}$ onto $\mathcal{A}$. As the extended coordinates are the same for all points which belong to the same component, there is a one-to-one correspondence between the cells $\widehat{C}_{\mathbf{v}} \in \widehat{\lhd_\omega}$ and $\widehat{\pi}(\widehat{C}_{\mathbf{v}}) \in \widehat{\triangle_\omega}$. The property of fine mixed is obtained because $\lhd_\omega$ is a triangulation. □

In the next sections, we will describe an approach to avoid the calculation of the cells that do not contribute to the mixed volume.

6.2.2 The computation of an initial mixed cell

The aim is to detect the degeneracy $V_n(\mathcal{P}) = 0$ by computing a mixed cell $\mathcal{C} = (C_1, C_2, \ldots, C_r)$, with $C_i = \{\mathbf{c}_{0i}, \mathbf{c}_{1i}, \ldots, \mathbf{c}_{k_i i}\} \subset A_i$. A straightforward approach could be to apply Algorithm 6.1.2 to compute from each set A_i in $\mathcal{A}$, k_i linearly independent points, w.r.t. a common origin. But this approach is only sure to work when for each A_i, $\dim(A_i) = n$, as illustrated in the following example.

Example 6.2.3 Consider the following tuple $\mathcal{A} = (A_1, A_2)$, $n = 2 = r$:

$$\mathcal{A} = (\ \{(0,0),(3,0),(1,2)\}\ ,\ \{(0,0),(1,0)\}\ ). \tag{6.12}$$

We should not take the first two points of A_1, because then there is no other linearly independent point left in A_2 to choose. Therefore we must first consider A_2. ■

Ordering the sets according to their dimension is only enough, whenever the A_i's lie in complementary affine spaces. But for the applications we have in mind, the sparsity of the vectors also plays an important role, as is illustrated in the following example.

Example 6.2.4 Consider the following tuple $\mathcal{A} = (A_1, A_2)$, $n = 3$, $r = 2$, with type of mixture given by (1,2):

$$\mathcal{A} = (\ \{(0,0,0),(1,0,0),(1,2,0)\}\ ,\ \{(0,0,0),(1,0,0),(0,0,1)\}\ ). \tag{6.13}$$

Both sets have the same dimension. We must choose the last point of the first set A_1, otherwise, no initial mixed cell can be found. ■

The instrument to deal with sparse vectors is provided in the following definition.

Definition 6.2.5 Given a set $A \subset \mathbb{E}^n$. Consider $i \in \{1, 2, \ldots, n\}$. The *occurrence of* i *w.r.t.* A is $\mathrm{occ}(i, A) = \#\{\mathbf{a} \in A \mid a_i \neq 0\}$. Consider $\mathbf{x} \in \mathbb{E}^n$. The *occurrence of* $\mathbf{x}$ *w.r.t.* A is $\mathrm{occ}(\mathbf{x}, A) = \min_{x_i \neq 0} \mathrm{occ}(i, A)$, if $\mathbf{x} \neq \mathbf{0}$ whereas $\mathrm{occ}(\mathbf{0}, A) = \#A + 1$. This extends to tuple of sets $\mathcal{A} = (A_1, A_2, \ldots, A_r)$: the *occurrence of* i *w.r.t.* $\mathcal{A}$ is $\mathrm{occ}(i, \mathcal{A}) = \sum_{j=1}^{r} \mathrm{occ}(i, A_j)$ and the *occurrence of* $\mathbf{x}$ *w.r.t.* $\mathcal{A}$ is $\mathrm{occ}(\mathbf{x}, \mathcal{A}) = \sum_{j=1}^{r} \mathrm{occ}(\mathbf{x}, A_j)$.

Example 6.2.6 Consider again the tuple $\mathcal{A}$ of Example 6.2.4. The vector $(8\ 2\ 1\ 8\ 2\ 1)$ gives the respective occurrences of the respective vectors w.r.t. $\mathcal{A}$, as listed in the order in which they appear in (6.13), i.e., $\mathrm{occ}((0\ 0\ 0), \mathcal{A}) = 8$, $\mathrm{occ}((1\ 0\ 0), \mathcal{A}) = 2$, etc... ■

The key step for an algorithm for computing an initial mixed cell is presented in Algorithm 6.2.7. Note that the occurrences must be adapted after every choice of a point. When occurrences are equal for two points, we have to choose that vector that is linearly independent from the remaining points.

Algorithm 6.2.7 Computation of the ith component of an initial mixed cell.

Input: i, k_i,	*current component*
$E \subseteq \cup_{j=1}^{i-1} A_j$,	*set of chosen points*
$\mathcal{R} = (A_i, \dots, A_r)$ with	*remainder of $\mathcal{A}$*
$\begin{cases} \mathbf{0} \in A_j, \ \#A_j = \dim(A_j) + 1, \\ \dim(A_j \cup E) \le \dim(A_{j+1} \cup E). \end{cases}$	*invariant conditions*
Output: C_i,	*ith component of a cell*
E, $\mathcal{R}$.	*updated sets*

$C_i := \{\mathbf{0}\}$; *initialization*
for j from 1 to k_i do
  choose $\mathbf{x} \in A_i$ so that
    (1) $\dim(E \cup \{\mathbf{x}\}) > \dim(E)$ *linearly independent*
    (2) $\mathbf{x}$ has minimal occurrence w.r.t. $\mathcal{R}$ *take sparsity into account*
      of all possible choices that satisfy (1);
  exit when no such $\mathbf{x}$ can be found;
  $C_i := C_i \cup \{\mathbf{x}\}$; $E := E \cup \{\mathbf{x}\}$; $R_i := R_i \setminus \{\mathbf{x}\}$; *update C_i, E and $\mathcal{R}$*
end for.

Algorithm 6.2.7 has to be applied r times, each time respecting the invariant conditions, given in the input specification. If the algorithm terminates with a component C_i, $\#C_i \le k_i$, then no linearly independent points could be found and the problem is degenerate. Proposition 6.2.8 provides a formal guarantee for Algorithm 6.2.7. The proof interpretes Algorithm 6.2.7 as the computation of one nonzero term in the expansion of the permanent of a $(0,1)$-matrix.

Proposition 6.2.8 *If Algorithm* 6.2.7 *yields a* C_i *with* $\#C_i \le k_i$, *then* $V_n(\mathcal{P}) = 0$.

Proof. Without loss of generality, the sparsest case can be focussed, i.e., all vectors in $\mathcal{A}$ are unit vectors, denoted by $\mathbf{e}_i$. For notational convenience, $r = n$ is assumed. With $\mathcal{A}$ the following $(0,1)$-matrix M can be associated: $M_{ij} = 1$, if $\mathbf{e}_j \in A_i$, otherwise $M_{ij} = 0$. So, the ith row of M represents A_i, for $i = 1, 2, \dots, n$. Then $V_n(\mathcal{P}) = 0 \Leftrightarrow \mathrm{per}(M) = 0$. When Algorithm 6.2.7 is iterated on each row, it picks of each row exactly one element on that column for which in the following rows there are a maximum number of zeros. Only when $\mathrm{per}(M) = 0$, the algorithm will not find enough nonzero elements. □

By computing the vertex set of each separate point set in the tuple, we can also eliminate those points that will certainly not influence the mixed volume, regardless of the subdivision used to compute it.

In the next chapter, we will present an alternative method for constructing an initial mixed cell, see Algorithm 7.3.10.

6.2.3 Influence of points on the mixed volume

A point of a polytope that is not a vertex has no influence on the volume and can be omitted. When we leave out a vertex of a polytope, then its volume decreases. For mixed volumes, we can also discard the nonvertex points, but there are vertices that can be omitted without decreasing the mixed volume. The aim of this section is to provide an efficient computational criterion to verify whether a vertex contributes to the mixed volume.

First we need an additional notation. By $K_\vee(\widehat{\mathbf{a}}, \widehat{A})$ we denote the normal cone to $\widehat{\mathbf{a}}$ in the lower hull of $\mathrm{conv}(\widehat{A})$. We introduce our criterion by means of a simple example.

Example 6.2.9 Consider the following polynomial systems

$$F(\mathbf{x}) = \left\{ \begin{array}{l} x_1 + x_2 - 1 = 0 \\ x_1^2 + x_2^2 - 1 = 0 \end{array} \right. \quad \text{and} \quad F^{(0)}(\mathbf{x}) = \left\{ \begin{array}{l} x_1 - 1 = 0 \\ x_2^2 - 1 = 0 \end{array} \right. . \tag{6.14}$$

We know that both systems have the same number of solutions. In fact, we can regard $F^{(0)}(\mathbf{x}) = \mathbf{0}$ as a start system (based on the total degree) to solve $F(\mathbf{x}) = \mathbf{0}$. This implies that the monomials x_2 and x_1^2 have no influence on the mixed volume of the system. The geometrical explanation becomes clear when we consider the intersection of the normal cone complexes associated with that subdivision that has as cell the support of $F^{(0)}$, as pictured in Figure 6.6.

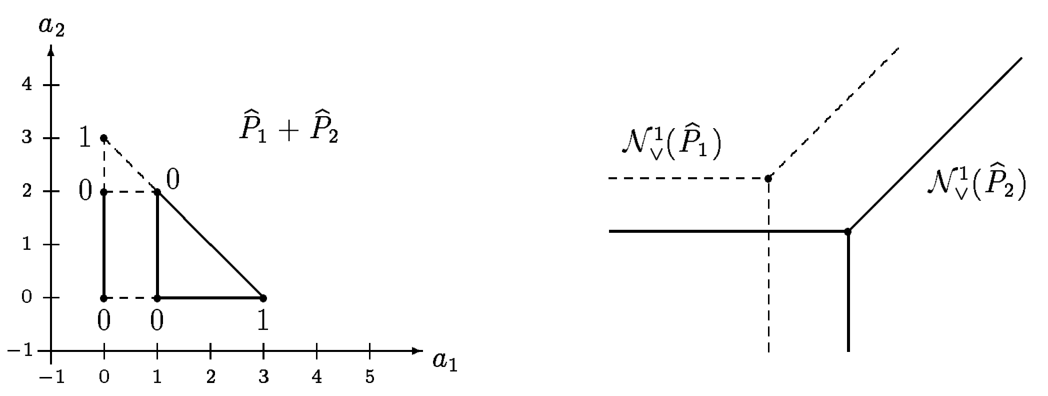


Figure 6.6: On the left, we see the projection of the lower hull of $\widehat{P}_1 + \widehat{P}_2$. On the right, the normal complexes $\mathcal{N}_\vee^1(\widehat{P}_1)$ and $\mathcal{N}_\vee^1(\widehat{P}_2)$ are drawn. Observe that $K_\vee((0,1,1), \widehat{P}_1) \subset K_\vee((0,2,0), \widehat{P}_2)$ and that $K_\vee((2,0,1), \widehat{P}_2) \subset K_\vee((1,0,0), \widehat{P}_1)$.

On the right of Figure 6.6, we see that the normal cone of the exponent vector of the monomial x_2 lies entirely in the normal cone of the exponent vector of x_2^2. Because the normal cones are disjunct, there can be no intersection between any normal cone, normal to a edge that contains $(0,1,1)$, and any other normal cone of the edges of $\widehat{P}_2$, so no mixed cell either. An analogue reasoning explains why x_1^2 is superfluous w.r.t. $F^{(0)}$.

Note that by the permutation $x_1 \leftrightarrow x_2$, the situation can be reversed. Therefore, we consider points always w.r.t. each other. ∎

Theorem 6.2.10 *Consider a tuple $\mathcal{A}$ of points sets, $\mathcal{A} = (A_1, A_2, \ldots, A_r)$, with $\sum_{i=1}^{r} k_i = n$, that span the tuple $\mathcal{P}$. Let $\mathbf{a} \in A_i$. Denote by $\mathcal{Q}$ the tuple of polytopes spanned by $(A_1, \ldots, A_i \setminus \{\mathbf{a}\}, \ldots, A_r)$. If $\exists \mathbf{b} \in A_j$, $j \neq i$: $K(\mathbf{a}, A_i) \subseteq K(\mathbf{b}, A_j)$, then $V_n(\mathcal{P}) = V_n(\mathcal{Q})$.*

Proof. We have to prove that there exists a lifting which induces a subdivision that contains no mixed cell with $\mathbf{a}$ in its ith component. Recall that $\bigwedge_{i=1}^{r} \mathcal{N}_{\vee}^{k_i}(\widehat{A}_i)$ provides the inner normals of facets of type $(k_1, k_2, \ldots, k_r)$ in the lower hull of $\mathrm{conv}(\widehat{A}_1 + \widehat{A}_2 + \cdots + \widehat{A}_r)$. We will construct a linear lifting because of $\mathcal{N}_{\vee}^{k_i}(\widehat{A}_i) \cong \mathcal{N}^{k_i}(A_i)$ and $K_{\vee}(\widehat{\mathbf{a}}, \widehat{A}_i) \cong K(\mathbf{a}, A_i)$.

The linear lifting we construct moves the apex of $K(\mathbf{a}, A_i)$ so that, after projection on the original coordinate space, the shifted cone $\widetilde{K}(\mathbf{a}, A_i) \subset K(\mathbf{b}, A_j)$, i.e., $\widetilde{K}(\mathbf{a}, A_i)$ has no intersection with the boundary of $K(\mathbf{b}, A_j)$. More precisely, we have that

$$K(\mathbf{b}, A_j) = \{\ \mathbf{v} \in \mathbb{Z}^n \mid \forall \mathbf{y} \in A_j : \langle \mathbf{y} - \mathbf{b}, \mathbf{v} \rangle \leq 0\ \} \tag{6.15}$$

and $\exists \mathbf{c} \in \mathbb{Z}_0^m$, $m = \# A_i$, so that

$$\widetilde{K}(\mathbf{a}, A_i) = \{\ \mathbf{v} \in \mathbb{Z}^n \mid \forall \mathbf{x}^{(k)} \in A_i : \langle \mathbf{x}^{(k)} - \mathbf{a}, \mathbf{v} \rangle \leq c_k,\ k = 1, 2, \ldots, m\ \} \tag{6.16}$$

lies entirely in $K(\mathbf{b}, A_j)$. This means that for the points in $\widetilde{K}(\mathbf{a}, A_i)$, the inequality ≤ 0 in (6.15) can be replaced by a strict inequality of type < 0. In general, we can say that $\widetilde{K}(\mathbf{a}, A_i)$ lies in the relative interior of $K(\mathbf{b}, A_j)$, denoted by $\widetilde{K}(\mathbf{a}, A_i) \subset \mathrm{relint}(K(\mathbf{b}, A_j))$.

For all normal cones $\mathcal{N}_{\vee}^{(k_i, \widehat{\mathbf{a}})}(\widehat{A}_i)$ on the k_i-faces that contain $\widehat{\mathbf{a}}$ we have that $\mathcal{N}_{\vee}^{(k_i, \widehat{\mathbf{a}})}(\widehat{A}_i) \subseteq K_{\vee}(\widehat{\mathbf{a}}, \widehat{A}_i)$. Because of the strict inclusion $K_{\vee}(\widehat{\mathbf{a}}, \widehat{A}_i) \subset K_{\vee}(\widehat{\mathbf{b}}, \widehat{A}_j)$, $\mathcal{N}_{\vee}^{(k_i, \widehat{\mathbf{a}})}(\widehat{A}_i) \wedge \mathcal{N}_{\vee}^{k_j}(\widehat{A}_j) = \emptyset$. Thus there is no inner normal to a facet of type $(k_1, k_2, \ldots, k_r)$ that contains $\widehat{\mathbf{a}}$ in its ith component and hence $\mathbf{a}$ does not contribute to the mixed volume. □

The interesting aspect of the criterion presented in Theorem 6.2.10 is that we can verify it rather cheaply, as illustrated by the following example.

Example 6.2.11 (Example 6.2.9 continued.) If we consider the generators of the normal cone on $\mathbf{a} = (0, 1) \in A_1$, then we can write

$$K(\mathbf{a}, A_1) = \left\{ \lambda_1 \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \lambda_2 \begin{bmatrix} -1 \\ 0 \end{bmatrix} + \lambda_3 \begin{bmatrix} 1 \\ 1 \end{bmatrix} \mid \lambda_1, \lambda_2, \lambda_2 \in \mathbb{N} \right\}. \tag{6.17}$$

The other normal cone on $\mathbf{b} = (0, 2) \in A_2$ can be represented by a system of linear inequalities:

$$K(\mathbf{b}, A_2) = \{\ \mathbf{v} \in \mathbb{Z}^2 \mid \forall \mathbf{x} \in A_2 : \langle \mathbf{x} - \mathbf{b}, \mathbf{v} \rangle \leq 0\ \} = \left\{ \begin{array}{r} -2v_2 \leq 0 \\ 2v_1 - 2v_2 \leq 0 \end{array} \right. . \tag{6.18}$$

So, to verify whether $K(\mathbf{a}, A_1) \subset K(\mathbf{b}, A_2)$, we simply have to check whether the generators of $K(\mathbf{a}, A_1)$ satisfy the system of linear inequalities of $K(\mathbf{b}, A_2)$. ■

In practice, to compute the generators of a normal cone to a point, it suffices to calculate the facet normals to all facets that contain that point, provided the polytope is of full dimension.

Proposition 6.2.12 *If* $\dim(A_i) = n$, *then the cost to verify the criterion presented in Theorem* 6.2.10 *equals* $O(n^3 \# \mathcal{F}^{\mathbf{a}}) + O(n \sum_{j \neq i} \# A_j)$, *given the set* $\mathcal{F}^{\mathbf{a}}$ *of all facets that contain* $\mathbf{a}$.

Proof. To compute one facet normal, we have to solve a linear system, which takes $O(n^3)$ arithmetical operations. We have to do this for all facets that contain $\mathbf{a}$. To complete the generators, we also have to include the vectors $\mathbf{x} - \mathbf{a}$ for all points $\mathbf{x}$ that span the facets. But this cost is bounded by $O(n^3 \# \mathcal{F}^{\mathbf{a}})$.

To verify the criterion, we have to see whether there exists a cone that contains all generators of $K(\mathbf{a}, A_i)$. Therefore we have to construct the system of inequalities for each point in the tuple of supports. The number of inequalities that determine a normal cone to a point is bounded by the number of points that span the polytope. So, to construct all these equalities, it takes $O(n \sum_{j \neq i} \# A_j)$ operations. The cost of verification is of the same order. Hence, the total cost of the criterion equals $O(n^3 \# \mathcal{F}^{\mathbf{a}}) + O(n \sum_{j \neq i} \# A_j)$. □

If the polytope A_i is not of full dimension, then we have to construct recursively normals to A_i and hereby transform the set A_i to lower dimensions until we are low enough to have a full-dimensional set. These transformations also require $O(n^3)$ arithmetical operations, which corresponds to the cost estimate given in Proposition 6.2.12.

Unfortunately, the criterion of Theorem 6.2.10 only works in one direction, at least when applied for $n > 2$. This means that it is not necessarily true that when the criterion fails that then also the point contributes to the mixed volume. We illustrate this with a simple three-dimensional example.

Example 6.2.13 Consider the following polynomial system

$$F(\mathbf{x}) = \left\{ \begin{array}{r} x_1^2 - 1 = 0 \\ x_2^2 + x_3^2 + 1 = 0 \\ x_1 x_2 + x_3 + 1 = 0 \end{array} \right. \quad \rightarrow \quad \begin{array}{l} x_1 = \pm 1 \\ \left\{ \begin{array}{r} x_2^2 + x_3^2 + 1 = 0 \\ \pm x_2 + x_3 + 1 = 0 \end{array} \right. . \end{array} \tag{6.19}$$

From the reduction on the left we can read off which monomials are superfluous, because we have transformed the system into a similar one as presented in Example 6.2.9. During this reduction, we have used the normal cone to the edge defined by the support of the first equation to reduce the normal cones of the rest. ■

Nevertheless we can use this criterion to eliminate those points that will certainly not contribute to the mixed volume. Either we can do this before we start the mixed-volume computation, or we can do it on line, while the dynamic lifting algorithm is running.

In [133], Rojas derived necessary and sufficient conditions to determine whether a point influences the mixed volume. In an e-mail discussion with me, he explained that his conditions can be refined to verifying whether the facial mixed volumes $V_{n-1}(\partial_{\mathbf{v}} \mathcal{P})$ (leaving out the ith component) are equal to zero, for all directions $\mathbf{v}$. It is doubtful whether the computational complexity of this iff-criterion is not harder than the complexity of the original problem, than the computation of the mixed volume of $\mathcal{P}$.

6.2.4 Conservative lifting functions applied to mixed cells

A linear lifting function ω is face structure preserving, i.e., there is a one-to-one correspondence between the faces $\partial_{\mathbf{v}} A_i$ of A_i and the faces of the lower hull of $\widehat{A}_i$. For a generic choice of linear ω a mixed subdivision $\mathcal{A}$ is induced. However, in the context of dynamic lifting a nonlinear lifting is more appropriate.

Analogously to Definition 6.1.6, we define conservative lifting functions. Denote $\widehat{\mathfrak{S}}_\omega$ as the collection of lifted mixed cells in the regular subdivision of $\mathcal{A} = (A_1, A_2, \ldots, A_r)$ and $\widehat{S}_\omega^{(i)}$ as the corresponding regular subdivision of A_i, $i = 1, 2, \ldots, r$. Note that $\widehat{S}_\omega^{(i)} \subset \widehat{S}_\omega$ as it contains only the cells of type $\mathbf{k}$, $k_i = n$ and $k_j = 0$, $\forall j \neq i$. Alternatively, we can interpret $\widehat{S}_\omega^{(i)}$ as the subdivision of A_i induced by the ith lifting function ω_i.

Definition 6.2.14 Let $\widehat{C}_{\mathbf{v}} = (\widehat{C}_1, \widehat{C}_2, \ldots, \widehat{C}_r)$ be a cell. Consider a point $\mathbf{x}$ w.r.t. $\widehat{C}_i$. Let $\omega(\mathbf{x}) = \widehat{x}_{n+1}$ so that $\langle \widehat{\mathbf{x}}, \mathbf{v} \rangle > \langle \widehat{C}_i, \mathbf{v} \rangle$, then ω is called a *conservative lifting w.r.t. the ith component of* $\widehat{C}_{\mathbf{v}}$. If ω is a conservative lifting w.r.t. all cells in $\widehat{S}_\omega^{(i)}$ and w.r.t. the ith component of all cells in $\widehat{\mathfrak{S}}_\omega$, then ω is called a *conservative lifting w.r.t.* $\widehat{S}_\omega^{(i)}$ *and w.r.t. the ith component of* $\widehat{\mathfrak{S}}_\omega$.

Analogously to Proposition 6.1.8, optimal discrete lifting functions can be defined.

Proposition 6.2.15 *Consider a point* $\mathbf{x}$ *w.r.t.* A_i. *Then the lowest possible value of a discrete conservative lifting function is given by*

$$\omega_i(\widehat{\mathfrak{S}}_\omega, \widehat{S}_\omega^{(i)}, \mathbf{x}) := \max\left(\omega_i(\widehat{S}_\omega^{(i)}, \mathbf{x}), \max_{\widehat{C}_{\mathbf{v}} \in \widehat{\mathfrak{S}}_\omega} \omega_i(\widehat{C}_{\mathbf{v}i}, \mathbf{x})\right), \quad i = 1, 2, \ldots, r, \tag{6.20}$$

where on the right-hand side, formulas (6.4) *and* (6.5) *are applied respectively.*

Analogously to the unmixed case, the mixed cells are preserved by conservative lifting. Denote the collection of mixed cells in the regular subdivision of $(A_1, \ldots, A_i \cup \{\mathbf{x}\}, \ldots, A_r)$ by $\widehat{\mathfrak{S}'_\omega}$ and the corresponding subdivision of $A_i \cup \{\mathbf{x}\}$ by $\widehat{S'}_\omega^{(i)}$.

Lemma 6.2.16 *Consider* $\mathbf{x}$ *w.r.t.* A_i. *Let* $\dim(A_i \cup \{\mathbf{x}\}) < n$. *Let* $\mathbf{x}$ *be lifted conservatively w.r.t. the ith component of* $\widehat{\mathfrak{S}}_\omega$. *Then* $\widehat{\mathfrak{S}}_\omega \subseteq \widehat{\mathfrak{S}'_\omega}$. *If* $\mathbf{x} \in \operatorname{conv}(A_i)$, *then* $\widehat{\mathfrak{S}}_\omega = \widehat{\mathfrak{S}'_\omega}$, *otherwise* $\forall \widehat{C}_{\mathbf{v}} \in \widehat{\mathfrak{S}'_\omega} \setminus \widehat{\mathfrak{S}}_\omega$, $\widehat{C}_{\mathbf{v}} = (\widehat{C}_1, \ldots, \widehat{C}_i, \ldots, \widehat{C}_r) : \widehat{\mathbf{x}} \in \widehat{C}_i$.

Proof. Due to the conservative lifting of $\mathbf{x}$, for each $\widehat{C}_{\mathbf{v}} \in \widehat{\mathfrak{S}}_\omega$, $\langle ., \mathbf{v} \rangle$ still attains its minimum at $\widehat{C}_{\mathbf{v}}$, so $\widehat{C}_{\mathbf{v}} \in \widehat{\mathfrak{S}'_\omega}$. The proof of the second statement is trivial. □

Lemma 6.2.17 *Consider* $\mathbf{x}$ *w.r.t.* A_i. *Let* $\dim(A_i \cup \{\mathbf{x}\}) = n$. *Let* $\mathbf{x}$ *be lifted conservatively w.r.t. the ith component of* $\widehat{\mathfrak{S}}_\omega$ *and w.r.t.* $\widehat{S}_\omega^{(i)}$. *Then* $\widehat{\mathfrak{S}}_\omega \subseteq \widehat{\mathfrak{S}}'_\omega$ *and* $\widehat{S}_\omega^{(i)} \subseteq \widehat{S'}_\omega^{(i)}$. *If* $\mathbf{x} \in \operatorname{conv}(A_i)$, *then* $\widehat{\mathfrak{S}}_\omega = \widehat{\mathfrak{S}'_\omega}$, *otherwise* $\forall \widehat{C}_{\mathbf{v}} \in \widehat{\mathfrak{S}'_\omega} \setminus \widehat{\mathfrak{S}}_\omega$, $\widehat{C}_{\mathbf{v}} = (\widehat{C}_1, \ldots, \widehat{C}_i, \ldots, \widehat{C}_r) : \widehat{\mathbf{x}} \in \widehat{C}_i$ *and* $\widehat{C}_i$ *belongs to the lower hull of* $\operatorname{conv}(\widehat{A}_i \cup \widehat{\mathbf{x}})$.

Proof. The statements concerning the mixed cells can be derived from Lemma 6.2.16. Because $\widehat{C}_{\mathbf{v}}$ belongs to the lower hull of $\operatorname{conv}(\widehat{A}_1 + \widehat{A}_2 + \cdots + \widehat{A}_r)$, its components $\widehat{C}_i$ belong to the lower hull of $\operatorname{conv}(\widehat{A}_i)$. □

Note that in special cases it may happen that no new mixed cells are obtained ($\widehat{\mathfrak{S}}_\omega = \widehat{\mathfrak{S}'_\omega}$) although $\mathbf{x} \notin \mathrm{conv}(A_i)$.

Recall that the inner normals of the mixed cells are determined by the intersection of the normal cone complexes, see Chapter 5. Lemma 6.2.17 uses the change of the normal cone complex $\mathcal{N}_\vee^{k_i}(\widehat{A}_i)$ by adding a point $\mathbf{x}$. Observe that $\mathcal{N}_\vee^{k_j}(\widehat{A}_j), j \neq i$, do not change. If $\mathbf{x} \notin \mathrm{conv}(A_i)$, then $\widehat{A'}_i = \widehat{A}_i \cup \{\mathbf{x}\}$ has new facets in the lower hull in comparison to $\widehat{A}_i$, which give new cells in $S'^{(i)}_\omega$.

The new facets in the lower hull generate new k_i-faces. Some of them are k_i-faces of old facets. Others are new. The normal cone of one old k_i-face C is modified: $K(C, \widehat{A'}_i) \subset K(C, \widehat{A}_i)$. Instead of $K(C, \widehat{A}_i) \in \mathcal{N}_\vee^{k_i}(\widehat{A}_i)$ we have $K(C, \widehat{A'}_i) \in \mathcal{N}_\vee^{k_i}(\widehat{A'}_i)$. The normal cones of the new k_i-faces $K(C, \widehat{A'}_i)$ are new elements of $\mathcal{N}_\vee^{k_i}(\widehat{A'}_i)$.

The analogue definition for flattening corresponds to Definition 6.1.12.

Computing new mixed cells can be done by lifting the new point $\mathbf{x}$ conservatively and applying Algorithm 5.3.29, where the input for the ith component can be restricted to the new k_i-faces that contain $\widehat{\mathbf{x}}$. It is important to note that even a simple mixed subdivision cannot be guaranteed, because we are only dealing with the collection of mixed cells. See Section 6.3.2 for how to deal with this problem.

6.2.5 The incremental construction with connectivity

In this section, the analogue pivoting mechanism of the dynamic lifting algorithm for the general case is described. Here we conjecture that in a regular subdivision, all mixed cells are connected. Therefore, we first investigate the modification of the normal cone complexes by the addition of a new point more carefully.

Assume that $\dim(A_j) = n$ for all $j = 1, \ldots, r$. There is a subdivision $S^{(i)}_\omega$ of A_i induced by ω_i. In addition to the normal cone complex $\mathcal{N}_\vee^{k_i}(\widehat{A}_i)$ we consider for each facet $\partial_\mathbf{v} \widehat{A}_i$ in the lower hull of $\widehat{A}_i$ the normal cone complex $\mathcal{N}_\vee^{k_i}(\partial_\mathbf{v} \widehat{A}_i)$. For each k_i-face C of $\partial_\mathbf{v} \widehat{A}_i$ there is a cone $K := K(C, \partial_\mathbf{v} \widehat{A}_i) \in \mathcal{N}_\vee^{k_i}(\partial_\mathbf{v} \widehat{A}_i)$. If C is a common face of two facets $\partial_{\mathbf{v}^{(1)}} \widehat{A}_i$ and $\partial_{\mathbf{v}^{(2)}} \widehat{A}_i$, then $K(C, \partial_{\mathbf{v}^{(1)}} \widehat{A}_i) \cap K(C, \partial_{\mathbf{v}^{(2)}} \widehat{A}_i) \subset K(C, \widehat{A}_i) \in \mathcal{N}_\vee^{k_i}(\widehat{A}_i)$. We consider the root $\bigwedge_{K \in \mathcal{N}_\vee^{k_i}(\partial_\mathbf{v} \widehat{A}_i)} K$ of the normal cone complex. Since $\partial_\mathbf{v} \widehat{A}_i$ is a facet of the lower hull, the root is generated by the outer normal $-\mathbf{v}$ of $\partial_\mathbf{v} \widehat{A}_i$.

Adding a point $\mathbf{x} \notin \mathrm{conv}(A_i)$ to A_i gives raise to new facets $\partial_\mathbf{v} \widehat{A'}_i$ in the lower hull and thus gives raise to new normal cones $K(\partial_\mathbf{v} \widehat{A'}_i)$ and new normal cone complexes $\mathcal{N}_\vee^{k_i}(\partial_\mathbf{v} \widehat{A'}_i)$ and new roots. This process can be visualised nicely with an example.

Example 6.2.18 Consider again the same polytopes as in Figure 6.5. On the left in Figure 6.7 the normal cone complexes $\mathcal{N}_\vee^1(\widehat{P}_1)$ and $\mathcal{N}_\vee^1(\widehat{P}_2)$ are drawn in a way that the intersection points are put above the corresponding mixed cells of the induced subdivision of (P_1, P_2). Since the induced subdivisions $S^{(1)}_\omega$ and $S^{(2)}_\omega$ consist only of one cell, respectively, the complexes $\mathcal{N}_\vee^1(\widehat{P}_1)$ and $\mathcal{N}_\vee^2(\widehat{P}_2)$ are already the complexes $\mathcal{N}_\vee^1(\partial_\mathbf{v} \widehat{P}_i)$ for the cells $\partial_\mathbf{v} \widehat{P}_i$ in the subdivision of $\widehat{P}_i$. On the right the point $(3, 1)$ has been considered w.r.t. P_2 and amended after lifting it conservatively. A new cell C_2 is obtained, so $S'^{(2)}_\omega = \{C_1, C_2\}$. Thus two roots are drawn. Two cones $K_1 \in \mathcal{N}_\vee^1(\widehat{C}_1)$ and $K_2 \in \mathcal{N}_\vee^1(\widehat{C}_2)$ have a nonempty intersection which is drawn as the line between the two roots. The conservativeness of the lifting is reflected by the fact that the intersection points in the left picture exist as well in the right picture. ■

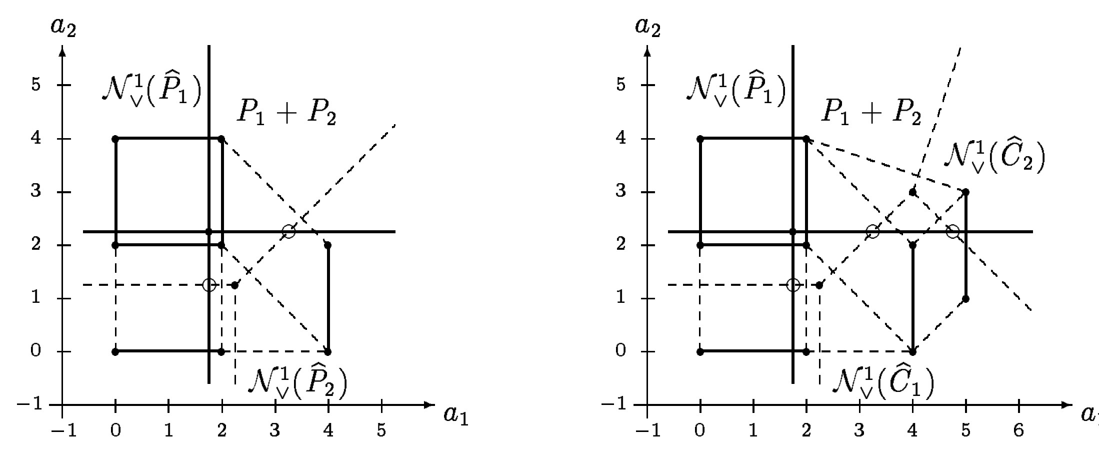


Figure 6.7: The projection of the normal cone complexes $\mathcal{N}^1_\vee(\widehat{P}_1)$ and $\mathcal{N}^1_\vee(\widehat{P}_2)$ are shown on the left. The roots of the normal cone complexes have been placed so that intersections between dashed and thick lines lie above the mixed cells. On the right, one point has been amended to P_2.

In Figure 6.7, we see that the three mixed cells are connected to each other. The argument presented in the following sentence is due to Paul Pedersen. By considering the corresponding construction with normal cone complexes, passing from one intersection point to another one becomes possible by going back to the roots.

For one polytope it is obvious that all faces are *connected to each other*, i.e., for each pair of k-faces $\mathcal{F}$, $\mathcal{F}^*$, there exists a path of k-faces $\mathcal{F}^{(j)}$, $j = 0, 1, \ldots, s$, so that $\mathcal{F}^{(0)} = \mathcal{F}$, $\mathcal{F}^{(s)} = \mathcal{F}^*$, $\dim(\mathcal{F}^{(j)} \cap \mathcal{F}^{(j+1)}) \geq k - 1$, $j = 0, 1, \ldots, s-1$. The notion of connectivity of mixed cells is more complicated.

Definition 6.2.19 Let S_ω be the induced subdivision of $\mathcal{A}$. Denote the collection of lifted mixed cells in $\widehat{S}_\omega$ by $\widehat{\mathfrak{S}}_\omega$. Two mixed cells $\widehat{C}, \widehat{C}^* \in \widehat{\mathfrak{S}}_\omega$ are *connected to each other* if there exists a path from $\widehat{C}$ to $\widehat{C}^*$: there exists a sequence of mixed cells $\widehat{C}^{(j)} \in \widehat{\mathfrak{S}}_\omega$, $j = 0, 1, \ldots, s$, so that $\widehat{C}^{(0)} = \widehat{C}$ and $\widehat{C}^{(s)} = \widehat{C}^*$, with

$$\dim(\widehat{C}_i^{(j)} \cap \widehat{C}_i^{(j+1)}) \geq k_i - 1 \quad \forall i = 1, \ldots, r, \quad j = 0, \ldots, s-1. \tag{6.21}$$

Ignoring the lifting values, the mixed cells C and C^* are said to be *connected to each other.* If all mixed cells in $\widehat{\mathfrak{S}}_\omega$ are connected to each other, then $\widehat{\mathfrak{S}}_\omega$ is *connected.* Ignoring the lifting values, $\mathfrak{S}_\omega$ is *connected* if all its mixed cells are connected to each other.

The following conjecture was first stated by Paul Pedersen [128]:

Conjecture 6.2.20 *Let S_ω be the induced subdivision of $\mathcal{A}$ and $\mathfrak{S}_\omega$ the set of mixed cells. Then $\mathfrak{S}_\omega$ is connected.*

For $n = 2$, Paul Pedersen has given a proof, based on normal fans. His idea has been pictured on the left of Figure 6.7.

Algorithm 6.2.21 allows the exploitation of the connectivity of the mixed cells. It is important to note that this only happens when $\widehat{\triangle}_\omega^{(i)} \neq \emptyset$, for full-dimensional polytopes. Also, as long as Conjecture 6.2.20 remains unproven, this part of the algorithm remains heuristic, but it can be directly switched off by omitting the test on $\widehat{\triangle}_\omega^{(i)} = \emptyset$ and applying Algorithm 5.3.29 with all lower k_i-faces instead of only the neighboring ones. Furthermore, note that for Algorithm 6.2.21, it is sufficient that this conjecture holds for any *placeable* regular mixed subdivision.

Algorithm 6.2.21 The dynamic lifting algorithm to compute new mixed cells.

Input: $\mathbf{k} = (k_1, k_2, \ldots, k_r)$, $n = \sum_{i=1}^r k_i$,	*type of mixture*
$\widehat{\mathcal{A}} = (\widehat{A}_1, \widehat{A}_2, \ldots, \widehat{A}_r)$,	*lifted point sets*
$\widehat{\mathcal{F}} = (\widehat{\mathcal{F}}_1, \widehat{\mathcal{F}}_2, \ldots, \widehat{\mathcal{F}}_r)$,	*k_i-faces of lower hull of* $\mathrm{conv}(\widehat{A}_i)$
$\widehat{\triangle}_\omega = (\widehat{\triangle}_\omega^{(1)}, \widehat{\triangle}_\omega^{(2)}, \ldots, \widehat{\triangle}_\omega^{(r)})$,	*regular triangulations of P_i*
$\widehat{\mathfrak{S}}_\omega = \{ \widehat{\mathcal{C}} \in \widehat{S}_\omega \mid V_n(\mathcal{C}, \mathbf{k}) > 0 \}$,	*collection of mixed cells*
$\mathbf{x}$, i.	*a point to add to A_i*
Output: $(\widehat{A}_1, \ldots, \widehat{A}_i \cup \{\widehat{\mathbf{x}}\}, \ldots, \widehat{A}_r)$,	*updated lifted point sets*
$(\widehat{\mathcal{F}}_1, \ldots, \widehat{\mathcal{F}}_i \cup \widehat{\mathcal{F}_i^{\mathbf{x}}}, \ldots, \widehat{\mathcal{F}}_r)$,	*updated k_i-faces*
$(\widehat{\triangle}_\omega^{(1)}, \ldots, \widehat{\triangle}_\omega^{(i)} \cup \widehat{\triangle}_\omega^{(i)\mathbf{x}}, \ldots, \widehat{\triangle}_\omega^{(r)})$,	*updated regular triangulations*
$\widehat{\mathfrak{S}}_\omega \cup \widehat{\mathfrak{S}_\omega^{\mathbf{x}}}$.	*updated collection of mixed cells*

COMPUTE $\widehat{\mathcal{F}_i^{\mathbf{x}}}$:	
$\widehat{x}_{n+1} := \omega(\widehat{\mathfrak{S}}_\omega, \widehat{\triangle}_\omega^{(i)}, \mathbf{x}, i)$;	*lift the point conservatively*
$\widehat{A}_i := \widehat{A}_i \cup \{\widehat{\mathbf{x}}\}$;	*update lifted points set*
if $\widehat{\triangle}_\omega^{(i)} = \emptyset$	*update triangulation of A_i*
then	
if $\dim(A_i) < n$	
then $\widehat{\mathcal{F}_i^{\mathbf{x}}} := Enumerate\_Faces(\widehat{\mathcal{A}}_i, k_i, \widehat{\mathbf{x}})$;	*new k_i-faces*
else $\widehat{\triangle}_\omega^{(i)\mathbf{x}} := Initial\_Facets(\widehat{\mathcal{A}}_i, \widehat{\mathbf{x}})$;	*initial facets of lower hull*
$\widehat{\mathcal{F}_i^{\mathbf{x}}} := Enumerate\_Faces(\widehat{\triangle}_\omega^{(i)\mathbf{x}}, k_i, \widehat{\mathbf{x}})$;	*new k_i-faces*
$\widehat{\triangle}_\omega^{(i)} := \widehat{\triangle}_\omega^{(i)} \cup \widehat{\triangle}_\omega^{(i)\mathbf{x}}$;	*update the lower hull*
end if;	
else $\widehat{\triangle}_\omega^{(i)\mathbf{x}} := New\_Facets(\widehat{\triangle}_\omega^{(i)}, \widehat{\mathbf{x}})$;	*apply Algorithm* 6.1.1
$\widehat{\mathcal{F}_i^{\mathbf{x}}} := Enumerate\_Faces(\widehat{\triangle}_\omega^{(i)\mathbf{x}}, k_i, \widehat{\mathbf{x}})$;	*new k_i-faces*
$\widehat{\triangle}_\omega^{(i)} := \widehat{\triangle}_\omega^{(i)} \cup \widehat{\triangle}_\omega^{(i)\mathbf{x}}$;	*update the lower hull*
end if.	

COMPUTE $\widehat{\mathfrak{S}}^{\mathbf{x}}_{\omega}$:
if $\widehat{\triangle}^{(i)}_{\omega} = \emptyset$
then $\widehat{\mathfrak{S}}^{\mathbf{x}} := Algorithm\_5.3.29(\widehat{\mathcal{F}}_1, \ldots, \widehat{\mathcal{F}}^{\mathbf{x}}_i, \ldots, \widehat{\mathcal{F}}_r)$; *new mixed cells*
else $\widehat{\mathfrak{S}}^{\mathbf{x}}_{\omega} := \emptyset$; *exploit neighborship*
for all cells $\mathcal{C} \in \mathfrak{S}_{\omega}$ do *apply the connectivity*
$\widehat{\mathcal{F}}'^{\mathbf{x}}_i := Neighboring\_Faces(\mathcal{C}, \widehat{\mathcal{F}}^{\mathbf{x}}_i)$; *neighbored new k_i-faces*
if $\widehat{\mathcal{F}}'^{\mathbf{x}}_i \neq \emptyset$ then
for all j in $\{1, \ldots, r\} \setminus \{i\}$ do
$\widehat{\mathcal{F}}'_j := Neighboring\_Faces(\mathcal{C}, \widehat{\mathcal{F}}_j)$; *apply the connectivity*
end for;
$\widehat{\mathfrak{S}}'^{\mathbf{x}}_{\omega} := Algorithm\_5.3.29(\widehat{\mathcal{F}}'_1, \ldots, \widehat{\mathcal{F}}'^{\mathbf{x}}_i, \ldots, \widehat{\mathcal{F}}'_r)$; *new mixed cells*
$\widehat{\mathfrak{S}}^{\mathbf{x}}_{\omega} := \widehat{\mathfrak{S}}^{\mathbf{x}}_{\omega} \cup \widehat{\mathfrak{S}}'^{\mathbf{x}}_{\omega}$; *update mixed cells*
end if;
end for;
end if.

The idea is to apply Algorithm 6.1.1 to compute the new k_i-faces of $\widehat{A}_i \cup \{\widehat{\mathbf{x}}\}$ after adding a point $\mathbf{x}$ to A_i. The new mixed cells will be computed by repeated application of Algorithm 5.3.29 which is called for each old mixed cell which is neighbored to a new k_i-face. Algorithm 5.3.29 is called with the following small input set. Let $\mathcal{C} = (C_1, \ldots, C_r)$ be the old mixed cell. $\widehat{\mathcal{F}}'^{\mathbf{x}}_i$ are those new k_i-faces which have a common $(k_i{-}1)$-face with $\widehat{C}_i$. $\widehat{\mathcal{F}}'_j, j \neq i$, are the k_j-faces which share a common $(k_j{-}1)$-face with $\widehat{C}_j$. In practice, it turns out to be efficient to put the new faces $\widehat{\mathcal{F}}'^{\mathbf{x}}_i$ in front of the argument list of Algorithm 5.3.29, in order to avoid to make face-face combinations which lead to other mixed cells, not containing any of the new faces.

In the beginning, when only the initial cell and some other cells exist then Algorithm 5.3.29 is called with all new k_i-faces. Note that, when $\dim(A_i)$ becomes n, due to the consideration of $\mathbf{x}$, *Initial_Facets* first computes one initial cell and then applies Algorithm 6.1.1 to all other points in A_i. Note that $\#A_i$ might be larger than $\dim(A_i)$. The new k_i-faces are computed by *Enumerate_Faces*, which applies either to the whole list $\widehat{A}_i$ or to the new cells in $\triangle^{(i)\mathbf{x}}_{\omega}$.

Algorithm 6.2.21 combines the advantages of both the Cayley trick (Algorithm 6.2.1) and the generalization of Betke's idea (Algorithm 5.3.29), because it allows to exploit the neighborship relations, so that many spurious face-face combinations can be prevented a priori, as opposed to the feasibility tests in Algorithm 5.3.29. This implies that Algorithm 6.2.21 provides an efficient solution to the unmixed, semimixed and fully mixed case.

6.2.6 Complexity and cost estimates

By application of the Cayley trick, Theorem 6.1.17 can be extended in a straightforward way. Again it is assumed that the problem is nondegenerate.

Theorem 6.2.22 *Given a tuple of points sets $\mathcal{A} = (A_1, A_2, \ldots, A_r)$. Denote $m = n + r - 1$. Let $\#\mathcal{A} = \sum_{i=1}^{r} \#A_i$ and $\mathcal{A}' = \mathcal{A} \setminus \mathcal{C}$, with $\mathcal{C}$ a cell. The construction of a* *placeable fine mixed subdivision S of $\mathcal{A}$ takes at least $O((\#S)m^3 + (\#\mathcal{A}')m^2)$, at most $O((\#S)m^3 + (\#S)(\#\mathcal{A}')m^2)$ and on average $O((\#S)m^3 + (\#S)(\#\mathcal{A}')m)$ arithmetical operations.*

Note that $\#S$ stands for the cardinality of the whole subdivision, with as well mixed as unmixed cells included. The Cayley trick becomes more expensive when r increases.

The asymptotic complexity for the dynamic lifting algorithm is the same as given in Theorem 5.3.32, although in practice, Algorithm 5.3.29 can take benefit from sharing factorizations, i.e., at level i, all face-face combinations of one branch can use the result of the same elimination step, while in Algorithm 6.2.21 these factorization has to be computed again, each time the algorithm is invoked with a new point.

6.3 Impact on polynomial system solving

In this section we describe the application of the dynamic lifting algorithm to polynomial system solving. We first introduce the idea of incremental polyhedral continuation. Then the necessity of recursive computations is illustrated by means of a simple example. Finally we show how an incremental application of the dynamic lifting algorithm can be useful to count in addition also the solutions with zero components.

6.3.1 Incremental polyhedral continuation

In the previous chapter, we discussed polyhedral continuation for computing all isolated solutions of $F(\mathbf{x}) = \mathbf{0}$. Here we illustrate the idea of *incremental polyhedral continuation* with a small example.

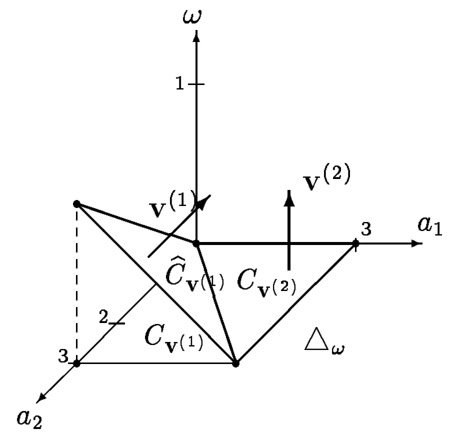
Example 6.3.1 In Figure 6.8, a regular triangulation is pictured of the support set of an unmixed polynomial system with randomly chosen complex coefficients.
The triangulation $\triangle_\omega$ also subdivides the system F in initial form systems $F_{\mathbf{v}^{(i)}}$, that are subsystems of F whose exponent vectors all belong to $C_{\mathbf{v}^{(i)}}$, $i = 1, 2, \ldots, \#\triangle_\omega$. The induced homotopy $\widehat{F}$ can be used directly to extend the solutions of the initial form system $F_{\mathbf{v}^{(2)}}$ to the system F. In order to extend the solutions corresponding to the subsystem $F_{\mathbf{v}^{(1)}}$, a transformation needs to be used, defined by the components of the inner normal $\mathbf{v}^{(1)}$:

$$\mathbf{v}^{(1)} = [1, -1, 3]^T = 3 \times \left[\frac{1}{3}, \frac{-1}{3}, 1\right]^T : \quad x_1 \leftarrow \tilde{x}_1 t^{1/3} \qquad x_2 \leftarrow \tilde{x}_2 t^{-1/3}, \tag{6.22}$$

which leads to the homotopy

$$\widehat{F}_{\mathbf{v}^{(1)}}(\tilde{x}_1, \tilde{x}_2, t) = \left\{ \begin{array}{l} \tilde{x}_1^3 \tilde{x}_2^3 + c_{11} \tilde{x}_1^3 t^1 + c_{12} \tilde{x}_2^3 + c_{13} = 0 \\ \tilde{x}_1^3 \tilde{x}_2^3 + c_{21} \tilde{x}_1^3 t^1 + c_{22} \tilde{x}_2^3 + c_{23} = 0 \end{array} \right. . \tag{6.23}$$

For $t = 0$: $\widehat{F}_{\mathbf{v}^{(1)}}(\tilde{\mathbf{x}}, 0) = F_{\mathbf{v}^{(1)}}(\tilde{\mathbf{x}})$ and for $t = 1$: $\widehat{F}_{\mathbf{v}^{(1)}}(\tilde{\mathbf{x}}, 1) = F(\tilde{\mathbf{x}})$. So by letting t vary from 0 to 1, the desired solutions can be computed. ■



$$F(x_1, x_2) = \left\{ \begin{array}{l} x_1^3 x_2^3 + c_{11} x_1^3 + c_{12} x_2^3 + c_{13} = 0 \\ x_1^3 x_2^3 + c_{21} x_1^3 + c_{22} x_2^3 + c_{23} = 0 \end{array} \right.$$

$$\widehat{F}(x_1, x_2, t) = \left\{ \begin{array}{l} x_1^3 x_2^3 t^0 + c_{11} x_1^3 t^0 + c_{12} x_2^3 t^1 + c_{13} t^0 = 0 \\ x_1^3 x_2^3 t^0 + c_{21} x_1^3 t^0 + c_{22} x_2^3 t^1 + c_{23} t^0 = 0 \end{array} \right.$$

$$\mathbf{v}^{(1)} = [1, -1, 3]^T \qquad \mathbf{v}^{(2)} = [0, 0, 1]^T$$

Figure 6.8: On the left the regular triangulation is pictured. The corresponding polynomial system and the induced homotopy are displayed at the right.

This example illustrates the advantages of the induced homotopy obtained by the dynamic lifting algorithm: all lifting values are as low as possible which leads to a well-conditioned homotopy whose solution paths are weakly nonlinear and hence less expensive to track numerically than with a homotopy induced by a random lifting function. Note that the computational effort to create the triangulation is negligible in comparison to the tracking of the solutions paths.

6.3.2 Unfolding cells with the same inner normal

For the unmixed case, we mentioned in Section 6.1.5 that due to flattening, different cells could get the same inner normal. Because the cells are stored as before, this fact does not influence the computation of new cells or the computation of the volume. However, for static polyhedral continuation, all cells need to have a different inner normal.

For dynamic polyhedral continuation, the following approach can be applied. The solutions which correspond to the existing cells have to be extended to the enlarged system, and therefore they can all use the *same* homotopy, generated by the *same* inner normal. In order to compute the solutions which correspond to the new cells, the points which belong to those new cells with the same inner normal can be relifted, by exploiting the data structure and by application of conservative lifting functions. In Figure 6.9 the *unfolding* of cells is pictured on a small example.

In general, we still have to rely on the recursion mechanism described in the previous chapter to deal with the case when some mixed cells are not exactly of type $(k_1, k_2, \ldots, k_r)$. So, there is a tradeoff on the bound to be set for the lifting value. If the bound is low, then flattening will occur frequently, which leads to relifting afterwards.

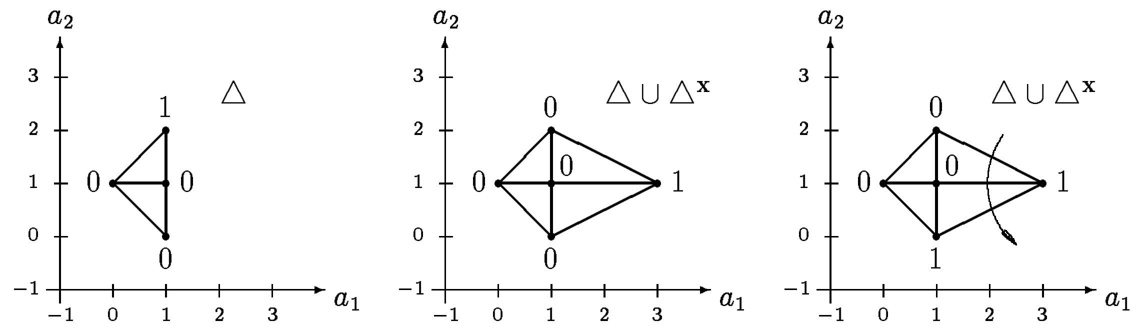


Figure 6.9: A triangulation $\triangle$ is shown on the left. The big numbers indicate the values of the lifting function. In the middle $\widehat{\triangle}$ has been flattened and new cells that contain $\mathbf{x} = (3, 1)$ have been added. The two new cells have the same inner normal $\mathbf{v} = [-1, 0, 2]^T$. The modification of the lifting is displayed on the right. Here the two new cells have a different inner normal. The arrow indicates the order of traversing the new cells while relifting.

6.3.3 Counting and computing in affine space

Bernshteĭn's theorem provides a root count for the solutions in $\mathbb{C}_0^n$, ignoring the solutions with zero components. Recently, a lot of research have been focussed on the development of a root count in affine space, to count all solutions in $\mathbb{C}^n$. In this section, we briefly summarize the main results of [71], [89], [133] and [134]. The last paragraph gives the importance of incremental mixed-volume computation and dynamic lifting in the practical realization of these root counts.

Rojas proved in [133] that the mixed volume of so-called shadowed Newton polytopes provides an upper bound for the number of solutions in affine space. These shadowed polytopes can be obtained by taking the Newton polytopes of $f_i(x_1 + c_1, x_2 + c_2, \ldots, x_n + c_n)$, where each c_i is a randomly chosen complex constant different from zero. Note that this transformation $x_i \mapsto x_i + c_i$ only needs to be performed for those components of which we want to know also the solutions with zero components. Unfortunately, the mixed volume of the shadowed Newton polytopes usually overestimates the number of solutions.

In [89], Li and Wang provided a sharper estimates for the number of isolated solutions in $\mathbb{C}^n$. They proved that by computing the mixed volume of the supports, each extended with the zero vector, an upper bound for the number of solutions in $\mathbb{C}^n$ can be obtained. Because the Newton polytopes considered by Li and Wang are smaller than the shadowed Newton polytopes, their bound yields an improvement of the construction presented in [133] by Rojas. We can state their main result (see [89] for a proof) as follows.

Theorem 6.3.2 (Li and Wang) *Let $F(\mathbf{x}) = \mathbf{0}$ be a polynomial system with supports $\mathcal{A} = (A_1, A_2, \ldots, A_n)$. The number of isolated solutions of $F(\mathbf{x}) = \mathbf{0}$ in $\mathbb{C}^n$ is bounded by $V_n(\mathrm{conv}(A_1 \cup \{\mathbf{0}\}), \mathrm{conv}(A_2 \cup \{\mathbf{0}\}), \ldots, \mathrm{conv}(A_n \cup \{\mathbf{0}\}))$.*

In [71], Huber and Sturmfels introduced the concept of stable mixed volumes. They give a bound on the number of solutions in affine space that is exact for generic choices of the coefficients. We briefly summarize their ideas.

Definition 6.3.3 Let $I \subset \{1, 2, \ldots, n\}$ and consider a regular subdivision S_ω of *the extended supports* $(A_1 \cup \{\mathbf{0}\}, A_2 \cup \{\mathbf{0}\}, \ldots, A_n \cup \{\mathbf{0}\})$, induced by a lifting ω that only lifts the zero constants and that gives the points in A_i lifting value 0. A cell $\mathcal{C}_\mathbf{v} \in S_\omega$ is called *I-stable* when all components of its inner normal $\mathbf{v}$ are nonnegative and when $v_i > 0$, $\forall i \in I$. The *I-stable mixed volume* is defined as the sum of the volumes of all I-stable mixed cells.

Note that, when $I = \emptyset$, the ordinary mixed volume is obtained. For $I = \{1, 2, \ldots, n\}$, the stable mixed volume is bounded by the mixed volume of the extended supports.

Theorem 6.3.4 (Huber and Sturmfels) *Let $I \subset \{1, 2, \ldots, n\}$. Denote by $\mathbb{C}^n_I = \{\mathbf{x} \in \mathbb{C}^n \mid x_i \neq 0,\ i \notin I\}$ Consider a polynomial system $F(\mathbf{x}) = \mathbf{0}$. The number of isolated solutions of $F(\mathbf{x}) = \mathbf{0}$ in $\mathbb{C}^n_I$, counted with multiplicity, is bounded by the stable mixed volume of the extended supports of F. The bound is exact for generic choices of the coefficients.*

In [134], Rojas and Wang provided conditions under which the stable mixed volume equals the mixed volume of the extended supports.

In practice, to count the additional roots with zero components we can start with a flattened regular subdivision of the original supports. By application of conservative lifting functions, the zero constants can be added one by one, each time followed by the application of Algorithm 6.2.21, to compute the new mixed cells. Eventually, as only the I-stable mixed cells are important, we could use the constraints $v_i > 0$, $\forall i \in I$, to speed up the pruning of Algorithm 5.3.29.

6.4 Applications

The dynamic lifting algorithm has been implemented to triangulate Newton polytopes, to apply the Cayley trick and to compute only the mixed cells in an incremental way. Except for the linear program solver, we have worked with integer calculus, because we are dealing with Newton polytopes.

6.4.1 The polynomial systems

The polynomials systems listed below are taken from the literature, see [24], [105], [117], [121], [178], [182], some come from the POSSO test suite. We have chosen a great variety of examples to illustrate the various aspects of the investigated algorithms.

The nine-point problem

A long standing problem in mechanism design has been the problem of finding all four-bar linkages whose coupler curve passes through nine prescribed points. Recently, in [178], for the first time a complete solution has been given. In [117] an efficient homotopy has been designed by exploiting its product-decomposition structure.

In [117] the unknowns of this system are grouped in two 6-tuples $\mathbf{z} = (n, \widehat{n}, x, \widehat{x}, a, \widehat{a})$ and $\mathbf{w} = (m, \widehat{m}, y, \widehat{y}, b, \widehat{b})$. The coordinates of the precision points are defined by the numbers δ_i and $\widehat{\delta}_i$, for $i = 1, 2, \ldots, 8$. The ninth precision point lies in the origin.
The first four equations are

$$n = a\widehat{x}, \quad \widehat{n} = \widehat{a}x, \quad m = b\widehat{y}, \quad \widehat{m} = \widehat{b}y \tag{6.24}$$

and the remaining eight equations have the following form

$$f_i(\mathbf{z}, \mathbf{w}) = \gamma_i(\mathbf{z}, \mathbf{w})\widehat{\gamma}_i(\mathbf{z}, \mathbf{w}) + \gamma_i(\mathbf{z}, \mathbf{w})\gamma_i^0(\mathbf{z}, \mathbf{w}) + \gamma_i^0(\mathbf{z}, \mathbf{w})\widehat{\gamma}_i(\mathbf{z}, \mathbf{w}) = 0, \quad i = 1, 2, \ldots, 8, \tag{6.25}$$

where the polynomials $\gamma_i(\mathbf{z}, \mathbf{w})$, $\widehat{\gamma}_i(\mathbf{z}, \mathbf{w})$ and $\gamma_i^0(\mathbf{z}, \mathbf{w})$ can be written in terms of linear ones:

$$\gamma_i(\mathbf{z}, \mathbf{w}) = q_i(\mathbf{z})r_i(\mathbf{w}) - q_i(\mathbf{w})r_i(\mathbf{z}), \tag{6.26}$$

$$\widehat{\gamma}_i(\mathbf{z}, \mathbf{w}) = r_i(\mathbf{z})p_i(\mathbf{w}) - r_i(\mathbf{w})p_i(\mathbf{z}), \tag{6.27}$$

$$\gamma_i^0(\mathbf{z}, \mathbf{w}) = p_i(\mathbf{z})q_i(\mathbf{w}) - p_i(\mathbf{w})q_i(\mathbf{z}), \tag{6.28}$$

where p_i, q_i and r_i are defined as

$$p_i(\mathbf{z}) = \widehat{n} - \widehat{\delta}_i x, \quad q_i(\mathbf{z}) = n - \delta_i \widehat{x}, \quad r_i(\mathbf{z}) = \delta_i(\widehat{a} - \widehat{x}) + \widehat{\delta}_i(a - x) - \delta_i\widehat{\delta}_i, \tag{6.29}$$

$$p_i(\mathbf{w}) = \widehat{m} - \widehat{\delta}_i y, \quad q_i(\mathbf{w}) = m - \delta_i \widehat{y}, \quad r_i(\mathbf{w}) = \delta_i(\widehat{b} - \widehat{y}) + \widehat{\delta}_i(b - y) - \delta_i\widehat{\delta}_i. \tag{6.30}$$

By the first four equations (6.24), the unknowns n, $\widehat{n}$, m and $\widehat{m}$ can be replaced by their respective right-hand sides into the remaining eight equations (6.25). The resulting system is unmixed and consists of eight equations of degree seven. Hence the total degree equals $7^8 = 5,764,801$. This substitution blows up the total degree (originally equal to $2^4 4^8 = 1,048,576$) and the best m-homogeneous Bézout bound (from 286,720 to 645,120, $m = 4$), but leaves the BKK bound unchanged. The fact that all polytopes in the system are the same makes it easier to handle.

Now we give the results of our program on this eight-dimensional problem. The timings presented are obtained from a run on a DECstation 5240, they only have a relative meaning, to illustrate the cost estimates. In the system there are 259 terms, but only 158 of them lead to vertices. The time needed to verify this was less than 3 minutes (163 cpu sec.). For a random addition of the points, there are 13,339 simplices in the triangulation, computed in about 36 minutes (2162 cpu sec.). The volume equals 83,977 divided by 8!. The computation of the volume, given the triangulation, costed about 1.3 minutes (75 cpu sec.).

The cost bounds for the dynamic lifting algorithm, as derived in Section 6.1.6, will be checked on this example. The time for the computation of the volume, given the triangulation $\triangle$, corresponds to $O((\#\triangle)n^2)$. Multiplication by n gives $O((\#\triangle)n^3)$, yielding 600 cpu sec. (75 cpu sec. $\times$ 8) as the total cost for all factorizations. By dividing $O((\#\triangle)n^2)$ by $\#\triangle$ and multiplying by $\#A'$, the cost for computing the additional decompositions in the optimal case is given by 0.8 cpu sec. (75 cpu sec. / 13,339 $\times$ (158–9)). In the worst case, multiplying $O((\#\triangle)n^2)$ by $\#A'$, yields 11,175 cpu sec. (75 cpu sec. $\times$ (158–9)), as the total cost for computing all decompositions. In the average case, dividing $O((\#A')(\#\triangle)n^2)$ by n yields 1397 cpu sec. (11,175 / 8). After adding the factorization and decomposition costs the following inequality is obtained: $601 < 2,162 < 11,775$, while the average cost bound equals 1997 cpu sec., which is quite close to the actual computing time. Note that the latter time contains not only the arithmetic operations, but also the overhead caused by e.g. memory management.

The purpose of this example is to demonstrate the complexity of the volume computation, and not to claim that this leads to an efficient approach for solving this system. Though it is worth noting that the BKK bound 83,977 is less than 286,720, which is the bound for the number of the solutions where the brute force technique in [178] was based on. As explained in [117] the product-decomposition structure of the system should be exploited in order to solve it more efficiently.

The system of A.H. Wright

In [182], the following polynomial system has been presented:

$$F(\mathbf{x}) = \begin{cases} x_1^2 - \underline{x_1} + \underline{x_2} + \underline{x_3} + \underline{x_4} + \underline{x_5} - 10 = 0 \\ x_2^2 + \underline{x_1} - \underline{x_2} + \underline{x_3} + \underline{x_4} + \underline{x_5} - 10 = 0 \\ x_3^2 + \underline{x_1} + \underline{x_2} - \underline{x_3} + \underline{x_4} + \underline{x_5} - 10 = 0 \\ x_4^2 + \underline{x_1} + \underline{x_2} + \underline{x_3} - \underline{x_4} + \underline{x_5} - 10 = 0 \\ x_5^2 + \underline{x_1} + \underline{x_2} + \underline{x_3} + \underline{x_4} - \underline{x_5} - 10 = 0 \quad . \end{cases} \tag{6.31}$$

The mixed volume of the supports equals 32, that is also the total degree of this system. This fact follows after the application of our criterion. The underlined monomials in (6.31) are those monomials that could be omitted without modification of the mixed volume.

The system called Reimer5

The following system belongs to the POSSO test suite (available by anonymous ftp at gauss.dm.unipi.it):

$$F(x,y,z,t,u) = \begin{cases} \underline{2x^2} - 2y^2 + \underline{2z^2} - \underline{2t^2} + \underline{2u^2} - 1 = 0 \\ \underline{2x^3} - \underline{2y^3} + 2z^3 - \underline{2t^3} + \underline{2u^3} - 1 = 0 \\ \underline{2x^4} - \underline{2y^4} + \underline{2z^4} - 2t^4 + \underline{2u^4} - 1 = 0 \\ \underline{2x^5} - \underline{2y^5} + \underline{2z^5} - \underline{2t^5} + 2u^5 - 1 = 0 \\ 2x^6 - \underline{2y^6} + \underline{2z^6} - \underline{2t^6} + \underline{2u^6} - 1 = 0 \quad . \end{cases} \tag{6.32}$$

The mixed volume equals the total degree $D = 6! = 720$. There can be only one mixed cell. After application of our criterion, all equations have only two monomials.

A PUMA robot

The hand position and orientation of a PUMA robot can be modeled [105] by the following

$$\left\{\begin{array}{r} \underline{x_1^2} + x_2^2 - 1 = 0 \\ x_3^2 + x_4^2 - 1 = 0 \\ x_5^2 + x_6^2 - 1 = 0 \\ x_7^2 + x_8^2 - 1 = 0 \\ \underline{0.004731 x_1 x_3} - \underline{0.3578 x_2 x_3} - \underline{0.1238 x_1} - \underline{0.001637 x_2} - 0.9338 x_4 + x_7 - 0.3571 = 0 \\ \underline{0.2238 x_1 x_3} + 0.7623 x_2 x_3 + \underline{0.2638 x_1} - \underline{0.07745 x_2} - 0.6734 x_4 - 0.6022 = 0 \\ x_6 x_8 + 0.3578 x_1 + 0.004731 x_2 = 0 \\ -0.7623 x_1 + \underline{0.2238 x_2} + 0.3461 = 0 \end{array}\right. . \tag{6.33}$$

The total degree of this system equals 128. By partitioning the set of unknowns in $Z = \{\{x_1, x_2\}, \{x_3, x_4, x_7, x_8\}, \{x_5, x_6\}\}$, the 3-homogeneous Bézout bound equals 16. The BKK bound equals 16, which is the exact number of isolated roots of this system. The remarkable fact of this system is that *any* initial mixed cell has volume 16. By our criterion, nine monomials can be omitted. In (6.33) these monomials are underlined.

Camera motion from point matches

The following system models the displacement of a camera between two positions in a static environment, with the coordinates of the matched points as given in [43]. The coordinates of the frames have been scaled, i.e., all components have been divided by 1000. In [47], this problem has been formulated using epipolar geometry.

$$\left\{\begin{array}{rcl} -d_1 q_1 - d_2 q_2 - d_3 q_3 + 1 & = & 0 \\ -3.6 d_1 q_1 + 4.1 d_1 q_2 + 2.0 d_1 q_3 + 0.1 d_1 + 4.1 d_2 q_1 + 1.8 d_2 q_2 + 3.7 d_2 q_3 - 0.2 d_2 & & \\ +2.0 d_3 q_1 + 3.7 d_3 q_2 - 4.0 d_3 q_3 + 0.3 d_3 + 0.1 q_1 - 0.2 q_2 + 0.3 q_3 + 5.8 & = & 0 \\ 0.3464 d_1 q_1 + 0.1732 d_1 q_2 - 5.999648 d_1 q_3 - 0.1732 d_1 + 0.1732 d_2 q_1 - 5.999648 d_2 q_2 & & \\ -0.1732 d_2 q_3 + 0.3464 d_2 - 5.999648 d_3 q_1 - 0.1732 d_3 q_2 - 0.3464 d_3 q_3 - 0.1732 d_3 & & \\ -0.1732 q_1 + 0.3464 q_2 - 0.1732 q_3 + 5.999648 & = & 0 \\ -5701.3 d_1 q_1 - 2.9 d_1 q_2 + 3796.7 d_1 q_3 - 1902.7 d_1 - 2.9 d_2 q_1 - 5698.7 d_2 q_2 & & \\ +1897.3 d_2 q_3 + 3803.3 d_2 + 3796.7 d_3 q_1 + 1897.3 d_3 q_2 + 5703.1 d_3 q_3 + 0.7 d_3 & & \\ -1902.7 q_1 + 3803.3 q_2 + 0.7 q_3 + 5696.9 & = & 0 \\ -6.8 d_1 q_1 - 3.2 d_1 q_2 + 1.3 d_1 q_3 + 5.1 d_1 - 3.2 d_2 q_1 - 4.8 d_2 q_2 - 0.7 d_2 q_3 - 7.1 d_2 & & \\ +1.3 d_3 q_1 - 0.7 d_3 q_2 + 9.0 d_3 q_3 - d_3 + 5.1 q_1 - 7.1 q_2 - q_3 + 2.6 & = & 0 \\ -2.140796 d_1 q_1 - 3.998792 d_1 q_2 + 3.715992 d_1 q_3 - 0.2828 d_1 - 3.998792 d_2 q_1 & & \\ -1.575196 d_2 q_2 - 3.998792 d_2 q_3 + 3.715992 d_3 q_1 - 3.998792 d_3 q_2 - 2.140796 d_3 q_3 & & \\ +0.2828 d_3 - 0.2828 q_1 + 0.2828 q_3 + 5.856788 & = & 0 \end{array}\right. \tag{6.34}$$

The total degree of this system equals 64. By partitioning the set of unknowns as $\{\{d_1, d_2, d_3\}, \{q_1, q_2, q_3\}\}$, the 2-homogeneous Bézout bound 20 is obtained, which equals the mixed volume and the exact number of isolated solutions. The system is semimixed, i.e., there are only three different support sets. More precisely, the type of mixture equals $(1, 4, 1)$. The Minkowski polynomial is displayed in (6.35), with the term of interest underlined.

$$
\begin{aligned}
&6!\mathrm{vol}_6(\lambda_1 P_1 + \lambda_2 P_2 + \lambda_3 P_3) \\
&= 20\lambda_1^3\lambda_2^3 + 20\lambda_1^3\lambda_2^2\lambda_3 + 20\lambda_1^3\lambda_2\lambda_3^2 + 16\lambda_1^3\lambda_3^3 + 20\lambda_1^2\lambda_2^4 + 20\lambda_1^2\lambda_2^3\lambda_3 + 20\lambda_1^2\lambda_2^2\lambda_3^2 \\
&+ 20\lambda_1^2\lambda_2\lambda_3^3 + 18\lambda_1^2\lambda_3^4 + 20\lambda_1\lambda_2^5 + \underline{20\lambda_1\lambda_2^4\lambda_3} + 20\lambda_1\lambda_2^3\lambda_3^2 + 20\lambda_1\lambda_2^2\lambda_3^3 + 20\lambda_1\lambda_2\lambda_3^4 \\
&+ 18\lambda_1\lambda_3^5 + 20\lambda_2^6 + 20\lambda_2^5\lambda_3 + 20\lambda_2^4\lambda_3^2 + 20\lambda_2^3\lambda_3^3 + 20\lambda_2^2\lambda_3^4 + 20\lambda_2\lambda_3^5 + 18\lambda_3^6
\end{aligned}
\tag{6.35}
$$

This application can be handled efficiently with the Cayley trick, and by exploiting the connectivity conjecture, the dynamic lifting algorithm computes the mixed volume more efficiently than the static lifting algorithm.

An inverse position problem

This system occurs as Example 3.3 in [175] and has been described in [174]. It represents an inverse position problem for 6-jointed robot arms.

$$
\left\{
\begin{array}{rcl}
c_1^2 + z_{21}^2 + z_{22}^2 - 1 & = & 0 \\
z_{31}^2 + z_{32}^2 + z_{33}^2 - 1 & = & 0 \\
z_{41}^2 + z_{42}^2 + z_{43}^2 - 1 & = & 0 \\
z_{51}^2 + z_{52}^2 + z_{53}^2 - 1 & = & 0 \\
c_1 z_{33} - c_2 + z_{21} z_{31} + z_{22} z_{32} & = & 0 \\
-c_3 + z_{31} z_{41} + z_{32} z_{42} + z_{33} z_{43} & = & 0 \\
-c_4 + z_{41} z_{51} + z_{42} z_{52} + z_{43} z_{53} & = & 0 \\
-c_1 + z_{51} z_{61} + z_{52} z_{62} + z_{53} z_{63} & = & 0 \\
-c_1 e_2 z_{32} + d_2 z_{21} + d_3 z_{31} + d_4 z_{41} + d_5 z_{51} - e_1 z_{22} + e_2 z_{22} z_{33} + e_3 z_{32} z_{43} & & \\
-e_3 z_{33} z_{42} + e_4 z_{42} z_{53} - e_4 z_{43} z_{52} + e_5 z_{52} z_{63} - e_5 z_{53} z_{62} - p_{61} & = & 0 \\
c_1 e_2 z_{31} + d_2 z_{22} + d_3 z_{32} + d_4 z_{42} + d_5 z_{52} + e_1 z_{21} - e_2 z_{21} z_{33} - e_3 z_{31} z_{43} & & \\
+e_3 z_{33} z_{41} - e_4 z_{41} z_{53} + e_4 z_{43} z_{51} - e_5 z_{51} z_{63} + e_5 z_{53} z_{61} - p_{62} & = & 0 \\
c_1 d_2 + d_3 z_{33} + d_4 z_{43} + d_5 z_{53} + e_2 z_{21} z_{32} - e_2 z_{22} z_{31} + e_3 z_{31} z_{42} & & \\
-e_3 z_{32} z_{41} + e_4 z_{41} z_{52} - e_4 z_{42} z_{51} + e_5 z_{51} z_{62} - e_5 z_{52} z_{61} - p_{63} & = & 0
\end{array}
\right.
\tag{6.36}
$$

The total degree of this system equals 1024. The 2-homogeneous Bézout number equals 320, with partition of the unknowns, computed in [175], $Z = \{\{z_{21}, z_{23}, z_{41}, z_{42}, z_{43}\}, \{z_{31}, z_{32}, z_{33}, z_{51}, z_{52}, z_{53}\}\}$. A better root count is provided by the BKK bound which equals 288. For any random complex choice of the parameters, there are only 16 finite regular solutions. This example illustrates the difficulty the dynamic lifting algorithm has when the factorizations cannot be shared (see the last paragraph of Section 6.2.6).

A heart-dipole problem

The following problem has been presented as a heart-dipole problem, see [110, 111]. The original problem description can be found in [121]. The four monomials that can be eliminated by the criterion are underlined.

$$F(a,b,c,d,t,u,v,w) = \left\{ \begin{array}{r} a + b = 0.6325 \\ c + d = 0.8465 \\ ta + ub - vc - wd = 0.1245 \\ va + wb + tc + ud = 5.3452 \\ at^2 - av^2 - 2ctv + bu^2 - bw^2 - 2duw = 1.4352 \\ ct^2 - cv^2 + 2atv + du^2 - dw^2 + 2buw = 0.9896 \\ \underline{at^3} - 3atv^2 + cv^3 - 3cvt^2 + \underline{bu^3} - 3buw^2 + dw^3 - 3dwu^2 = 0.3464 \\ ct^3 - 3ctv^2 - \underline{av^3} + 3avt^2 + du^3 - 3duw^2 - \underline{bw^3} + 3bwu^2 = 3.1345 \end{array} \right. \tag{6.37}$$

The right-hand sides of the equations are the parameters of the system and have been chosen at random. The total degree of this system equals 576. When partitioning the set of unknowns into $Z = \{\{a,b,c,d\},\{t,u,v,w\}\}$, the 2-homogeneous Bézout bound equals 193. The BKK bound equals 121. In [111] the number of solutions with a generic choice of the parameters, the so-called coefficient-parameter bound, is reported to equal 32. However, there is a type error in the formulation of the system, as presented in [111], so that for the original problem, presented in [121] and in [110], there can be only 4 regular solutions, for random right-hand sides. There are only two real symmetrical solutions. Note that in [110], this system has been reduced to a quadratic univariate equation.

Butcher's problem

The next system belongs to the POSSO test suite, available at the site gauss.dm.unipi.it by anonymous ftp.

$$\left\{ \begin{array}{r} zu + yv + tw - w^2 - 1/2w - 1/2 = 0 \\ zu^2 + yv^2 - tw^2 + w^3 + w^2 - 1/3t + 4/3w = 0 \\ xzv - tw^2 + w^3 - 1/2tw + w^2 - 1/6t + 2/3w = 0 \\ zu^3 + yv^3 + tw^3 - w^4 - 3/2w^3 + tw - 5/2w^2 - 1/4w - 1/4 = 0 \\ xzuv + tw^3 - w^4 + 1/2tw^2 - 3/2w^3 + 1/2tw - 7/4w^2 - 3/8w - 1/8 = 0 \\ xzv^2 + tw^3 - w^4 + tw^2 - 3/2w^3 + 2/3tw - 7/6w^2 - 1/12w - 1/12 = 0 \\ -tw^3 + w^4 - tw^2 + 3/2w^3 - 1/3tw + 13/12w^2 + 7/24w + 1/24 = 0 \end{array} \right. \tag{6.38}$$

This example clearly illustrates the efficiency of the mixed volume as root count. The total degree equals 4608. The 4-homogeneous Bézout number equals 1361. With the algorithm presented in [163] a set structure, which yields the generalized Bézout number 605, can be obtained. The BKK bound equals 24 and can be computed in less than 1 minute (on a DECStation 5240). There are 5 isolated solutions and a component of solutions: $t = -1 = w$, $z = 0 = y$ and $u, v \in \mathbb{C}$.

The cyclic n-roots problem

The following application belongs to a family of systems which have been presented in [6, 13, 14]. The general formulation goes as follows:

$$\left\{\begin{array}{ll} f_k(\mathbf{x}) = \displaystyle\sum_{i=1}^{n} \prod_{j=1}^{k} x_{(i+j) \bmod n}, & k = 1, 2, \ldots, n-1, \\ f_n(\mathbf{x}) = \displaystyle\prod_{j=1}^{n} x_j - 1. & \end{array}\right. \tag{6.39}$$

In Table 6.1, we can see the performance of the mixed volume as root count, compared to the Bézout bounds. This application also demonstrates the $\#P$-hardness of the problem of mixed-volume computation. Augmenting the dimension n leads to a significantly harder problem.

In (6.40) the Minkowski polynomial for the cyclic 5-roots problem is displayed. The term of interest is underlined.

$$\begin{aligned} & 5!\mathrm{vol}_5(\lambda_1 P_1 + \lambda_2 P_2 + \lambda_3 P_3 + \lambda_4 P_4 + \lambda_5 P_5) \\ & = 5\lambda_1^4\lambda_5 + 10\lambda_1^3\lambda_2\lambda_5 + 15\lambda_1^3\lambda_3\lambda_5 + 20\lambda_1^3\lambda_4\lambda_5 + 20\lambda_1^2\lambda_2^2\lambda_5 + 30\lambda_1^2\lambda_2\lambda_3\lambda_5 \\ & + 40\lambda_1^2\lambda_2\lambda_4\lambda_5 + 20\lambda_1^2\lambda_3^2\lambda_5 + 35\lambda_1^2\lambda_3\lambda_4\lambda_5 + 30\lambda_1^2\lambda_4^2\lambda_5 + 15\lambda_1\lambda_2^3\lambda_5 \\ & + 35\lambda_1\lambda_2^2\lambda_3\lambda_5 + 30\lambda_1\lambda_2^2\lambda_4\lambda_5 + 40\lambda_1\lambda_2\lambda_3^2\lambda_5 + \underline{70\lambda_1\lambda_2\lambda_3\lambda_4\lambda_5} + 35\lambda_1\lambda_2\lambda_4^2\lambda_5 \\ & + 10\lambda_1\lambda_3^3\lambda_5 + 30\lambda_1\lambda_3^2\lambda_4\lambda_5 + 40\lambda_1\lambda_3\lambda_4^2\lambda_5 + 20\lambda_1\lambda_4^3\lambda_5 + 5\lambda_2^4\lambda_5 \\ & + 20\lambda_2^3\lambda_3\lambda_5 + 10\lambda_2^3\lambda_4\lambda_5 + 30\lambda_2^2\lambda_3^2\lambda_5 + 40\lambda_2^2\lambda_3\lambda_4\lambda_5 + 20\lambda_2^2\lambda_4^2\lambda_5 \\ & + 20\lambda_2\lambda_3^3\lambda_5 + 35\lambda_2\lambda_3^2\lambda_4\lambda_5 + 30\lambda_2\lambda_3\lambda_4^2\lambda_5 + 15\lambda_2\lambda_4^3\lambda_5 + 5\lambda_3^4\lambda_5 \\ & + 15\lambda_3^3\lambda_4\lambda_5 + 20\lambda_3^2\lambda_4^2\lambda_5 + 10\lambda_3\lambda_4^3\lambda_5 + 5\lambda_4^4\lambda_5 \end{aligned} \tag{6.40}$$

For increasing n the Minkowski polynomial grows enormously. We see that the Cayley trick is not suited for this type of applications.

Note that when the system has to be solved, it is better to apply the following transformation, proposed in [43]: $x_i = y_i/y_n$, $i = 1, 2, \ldots, n-1$, and multiply the kth equation by y_n^k, for $k = 1, 2, \ldots, n$.

$$\left\{\begin{array}{ll} f_k(x_i = \dfrac{y_i}{y_n}) y_n^k = y_n^k \displaystyle\sum_{i=1}^{n} \prod_{j=1}^{k} \frac{y_{(i+j) \bmod n}}{y_n}, & k = 1, 2, \ldots, n-1, \\ f_n(x_i = \dfrac{y_i}{y_n}) y_n^n = \displaystyle\prod_{j=1}^{n-1} y_j - y_n^n. & \end{array}\right. \tag{6.41}$$

After elaboration of (6.41), we see that the last unknown y_n only appears in the last equation. So, we can first concentrate on the first $n-1$ equations, which yields a system with a mixed volume that is n times less than the mixed volume of the original problem. However, the original formulation has been used here for solving the system.

A geometric problem

The following system has been communicated to me by Bernd Sturmfels, who received it from Neil Sloane. The application field deals with the construction of nice sets of points in Euclidean space, see [24].

$$F(a,b,c,d,e,f,g) = \left\{ \begin{array}{r} \underline{1} - 2a^5 - 2b^5 - 2c^5 - 2d^5 - \underline{4e^5} - 4f^5 - 4g^5 = 0 \\ \underline{1} - 18a^4b - 18c^4d - 36e^4f - 36g^4e - 36f^4g = 0 \\ \underline{1} - 18b^4a - 18d^4c - 36e^4g - 36g^4f - 36f^4e = 0 \\ \underline{1} - 42a^3b^2 - 42c^3d^2 - 84e^3f^2 - 84g^3e^2 - 84f^3g^2 = 0 \\ \underline{1} - 42b^3a^2 - 42d^3c^2 - 84e^3g^2 - 84g^3f^2 - 84f^3e^2 = 0 \\ \underline{1} - 252e^3fg - 252g^3ef - 252f^3ge = 0 \\ 1 - 420e^2f^2g - 420g^2e^2f - 420f^2g^2e = 0 \end{array} \right. \qquad (6.42)$$

The total degree of this system equals $5^7 = 78,125$, while an improved Bézout bound based on a set structure yields 64,855. The BKK bound of this system is 23,750. Seven monomials can be omitted. This provides a significant speed up in computing the mixed volume.

6.4.2 Performance

In Table 6.1, the characteristics of each application are summarized. The meaning of the columns is as follows. The first and second column provide a label and a short description of the application. The following columns respectively list the dimension n, the number of different supports r in the tuple, the total degree D, the best known generalized Bézout bound B, the mixed volume $V_n(\mathcal{P})$ and the number of isolated solutions $N(F)$ in $\mathbb{C}_0^n$.

	description of	characteristics					
system	the applications	n	r	D	B	$V_n(\mathcal{P})$	$N(F)$
(6.25)	nine point	8	1	5,764,801	645,120	83,977	8,652
(6.31)	system of Wright	5	5	32	32	32	32
(6.32)	system called Reimer5	5	5	720	720	720	144
(6.33)	PUMA robot	8	8	256	16	16	16
(6.34)	camera motion	6	3	64	20	20	20
(6.36)	inverse position	11	11	1,024	320	288	16
(6.37)	heart-dipole	8	8	576	193	121	4
(6.38)	Butcher's problem	7	7	4,608	605	24	5
(6.39)	cyclic n-roots	5	5	120	108	70	70
		6	6	720	504	156	156
		7	7	5,040	3,960	924	924
		8	8	40,320	20,352	2,560	1,152
(6.42)	geometric problem	7	7	78,125	64,855	23,750	3,000

Table 6.1: Characteristics of the polynomial systems.

Furthermore, Table 6.2 lists the number of mixed cells $\#\triangle$ and the cardinality $\#\triangleleft$ of the triangulation of the polytope used in the Cayley trick. A '–' in the table indicates that either the Cayley trick was inappropriate to use (as is the case for the first application) or that the Cayley was too expensive to perform (as is the case for the other applications).

system	description of the applications	cardinalities	
		$\#\mathfrak{S}_\omega$	$\#\triangleleft$
(6.25)	nine point	13,339	–
(6.31)	system of Wright	2	331
(6.32)	system called Reimer5	1	126
(6.33)	PUMA robot	1	580
(6.34)	camera motion	12	392
(6.36)	inverse position	17	–
(6.37)	heart-dipole	36	–
(6.38)	Butcher's problem	4	1,867
(6.39)	cyclic n-roots	14	166
		25	1,109
		124	13,180
		268	–
(6.42)	geometric problem	29	–

Table 6.2: Cardinalities of the triangulations: the number of mixed cells versus the number of total cells.

In Table 6.3, we show the performance for those systems where we applied the criterion on to eliminate monomials that do not influence the mixed volume. We list the cardinalities of the mixed cells in the subdivision, once computed with all points and once after the application of the criterion.

system	description of the applications	Cardinalities		
		$\#S_\omega$	$\#S'_\omega$	#omit
(6.31)	system of Wright	3	1	20
(6.32)	system called Reimer5	1	1	20
(6.33)	model of PUMA robot	1	1	9
(6.37)	heart-dipole problem	14	27	4
(6.42)	geometric problem	29	10	7

Table 6.3: Performance of the criterion on some applications: the number of mixed cells $\#S_\omega$ and $\#S'_\omega$, in the respective subdivisions computed once with all points and once after application of the criterion. The number of points that could be omitted is given in the last column.

6.5 Conclusions and future research

Three different algorithms have been investigated for computing mixed volumes by means of mixed subdivisions: the Cayley trick, static and dynamic lifting. The key idea presented in this chapter is that of conservative lifting functions which allow to construct regular triangulations without the randomness assumption, generally required by all other approaches. This has led to the construction of well-conditioned polyhedral homotopies for computing all isolated solutions to polynomial systems, which provides an important elaboration of the ideas presented in [72].

The Cayley trick is efficient when either we want to compute all cells, i.e., also the cells that do not contribute to the mixed volume, or when the system is semimixed and the total number of cells compared to the number of mixed cells is not exponentially large. Actually, the Cayley trick is very effective in computing the Minkowski polynomial. The fact that the lifting function is again ruled out can be considered as an advantage. The disadvantage of this approach is that, for fully mixed systems, the number of mixed cells is much less than the total number of cells. Note that the space complexity of the Cayley trick can be overcome by using reverse enumeration methods, developed by Avis and Fukuda, see [5]. But these techniques cannot be applied to remove the randomness assumption on the lifting function and hence offer less control on the growth of the lifting values than the dynamic lifting algorithm.

For fully mixed systems, the lifting method, based on Betke's idea, allows to compute only the mixed cells and has to be prefered. The static lifting algorithm, with a randomized lifting and with properly worked out feasibility tests provides a very efficient way to compute mixed subdivisions and mixed volumes. It is used in the dynamic lifting algorithm which turns out to be less efficient, due to the fact that factorizations cannot be shared any longer. Nevertheless, the extra work done by the dynamic lifting algorithm pays off when it comes to constructing polyhedral homotopies for solving polynomial systems. The efficiency of both the static and dynamic lifting algorithm depends largely on the efficiency by which the feasibility tests can be worked out, which is determined by the efficiency of the linear programming solver. So, linear programming forms the computational bottleneck of both algorithms.

Another important conclusion of this project is that computing the mixed volume is in practice not harder than solving the system by tracking all solution paths. The incremental aspect of solving polynomial systems, in [169] applied for computing the solutions inside a bounded domain, also provides more insight in the complexity of homotopy continuation for this problem, see [141] for the complexity analysis of Bézout's theorem.

Finally, we can mention some important open problems. First of all, there is the proof of the conjecture on the connectivity of the mixed cells. Furthermore, from an algorithmic point of view, it would be interesting to develop algorithms, analogue to the flipping mechanisms proposed in [40], which transform placeable (mixed) subdivisions into any desired regular (mixed) subdivisions, with e.g. either a minimum or a maximum number of cells. The elaboration of a generalized version of the criterion to check which points influence the mixed volume also deserves attention. Last but not least, the BKK bound provides for many applications not an exact root count. It would be worthwhile to develop a systematic approach to reformulate problems to an equivalent formulation with a lower mixed volume, like e.g. done with the cyclic n-roots problem.

Chapter 7

Symmetric Lifting

This chapter describes the application of polyhedral homotopy methods to symmetric polynomial systems.

There are at least two reasons to consider the symmetry of the given system of equations. On the one hand the lifting method works generically for almost all choices of coefficients, but symmetry may force the exceptional situation. On the other hand the use of symmetry makes the algorithm much more efficient.

To construct symmetric polyhedral homotopies, we developed in [165] *symmetric lifting*. This lifting method consists in two steps. First the points in the support sets are classified into orbits. During the second step we give all points in the same orbit the same random lifting value. To compute the generators of the solution set of a symmetric random coefficient start system, it suffices to follow the paths starting at the generating initial form systems defined by the polyhedral homotopy induced by a symmetric lifting.

In addition to the techniques described in [165], this chapter discusses the degeneracy problem further. The aim is to construct nondegenerate and proper subdivisions by an appropriate assignment of the lifting values to the orbits. For this purpose we extend the dynamic lifting algorithm so that we can construct symmetric subdivisions with it.

This chapter is organized as follows. First we define what symmetric Laurent polynomial systems are. The second part of this chapter contains the description of the symmetric static lifting algorithm. The symmetric dynamic lifting is explained in the third part. Finally, we demonstrate the usefulness of the developed techniques by presenting some practical applications. We conclude by stating our conclusions and suggestions for future research.

7.1 Symmetric Laurent polynomial systems

In this section we establish the notation concerning symmetry groups applied to systems of Laurent polynomials and their supports.

Recall that we describe symmetry by means of a finite group G, represented by matrix representations $V, W : G \to GL(n, \mathbb{C})$, with

$$V(g_1)V(g_2) = V(g_1g_2) \quad \text{and} \quad W(g_1)W(g_2) = W(g_1g_2), \quad \forall\, g_1, g_2 \in G. \tag{7.1}$$

Of course we assume that V and G are isomorphic as groups. Otherwise we would consider a subgroup of G.

We allow different groups to act on the polynomials in F and on the supports $\mathcal{A}$, because the exploitation of the symmetry in the supports can make the construction of a regular subdivision more efficient even when the system is not symmetric. We restrict to cases where the group operation leaves the supports $\mathcal{A}$ invariant and can be expressed by two permutation representations that are not necessarily the same. The group actions on $\mathcal{A}$ are represented by permutation matrices belonging to $Z_2^{r\times r}$, with $Z_2 = \{0, 1\}$.

Definition 7.1.1 Consider a tuple F of Laurent polynomials with supports $\mathcal{A}$. If

$$W^F(g)F(\mathbf{x}) = F(V^F(g)\mathbf{x}), \quad \forall g \in G^F, \forall \mathbf{x} \in \mathbb{C}^n, \tag{7.2}$$

then the system F is said to be (G^F, V^F, W^F)*-symmetric*. A tuple $\mathcal{A}$ of supports is called $(G^{\mathcal{A}}, V^{\mathcal{A}}, W^{\mathcal{A}})$*-symmetric*, if

$$W^{\mathcal{A}}(g)\mathcal{A} = \mathcal{A} \circ V^{\mathcal{A}}(g), \quad \forall g \in G^{\mathcal{A}}. \tag{7.3}$$

The operation on the left of (7.3) corresponds to a permutation of the supports A_i whereas the operation on the right of (7.3) represents a permutation of the components of all vectors $\mathbf{a} \in A_i$, for $i = 1, 2, \ldots, r$.

Example 7.1.2 We give a simple example where $G^F = Z_2(s_1) \times Z_2(s_2) = \{id, s_1, s_2, s_1s_2\}$ and V^F is a matrix representation defined by

$$V^F(id) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad V^F(s_1) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad V^F(s_2) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{7.4}$$

$V^F(s_1s_2) = V^F(s_1)V^F(s_2)$ and $W^F(s_1) = V^F(s_1)$, $W^F(s_2) = V^F(id)$. So, we consider an equivariant system

$$F(\mathbf{x}) = \begin{cases} x_1^2x_2^2 + 3x_1^2 + x_2^2 + x_1x_2 + 5 = 0 \\ x_1^2x_2^2 + x_1^2 + 3x_2^2 + x_1x_2 + 5 = 0 \end{cases} \tag{7.5}$$

that is $(Z_2(s_1) \times Z_2(s_2), V^F, W^F)$-symmetric. Since $Z_2(s_2)$ does not transport to the Newton polytopes (drawn in Figure 7.1), we consider F also as $(Z_2(s_1), V^F, V^F)$-symmetric.

The support $\mathcal{A}$ is unmixed, i.e.: $A_1 = A_2$, though, in order to exploit the symmetry we consider the system as fully mixed. Then $Z_2(s_1)$ is acting on the support with $V^{\mathcal{A}}(s_1) = V^F(s_1)$, $V^{\mathcal{A}}(s_2) = id$, $W^{\mathcal{A}}(s_1) = W^F(s_1)$, $W^{\mathcal{A}}(s_2) = id$. The group action s_1 transports $(2, 0) \in A_1$ to $(0, 2) \in A_2$. The element s_2 does not have any effect. Note that G^F and $G^{\mathcal{A}}$ are not isomorphic as groups, because $G^{\mathcal{A}} = Z_2(s_1)$. ■

Example 7.1.3 Let $G^F = Z_2(s_1) \times Z_2(s_2)$ and V^F be like in Example 7.1.2. Assume W^F is given by $W^F(s_1) = W^F(s_2) = V^F(id)$. Then we consider

$$F(\mathbf{x}) = \begin{cases} x_1^2x_2^2 + x_1^2 + x_2^2 + x_1x_2 + 3 = 0 \\ x_1^2x_2^2 + 3x_1^2 + 3x_2^2 + x_1x_2 + 5 = 0 \end{cases} \tag{7.6}$$

is $(Z_2(s_1) \times Z_2(s_2), V^F, W^F)$-symmetric and consists of two invariant polynomials. Because the system in (7.6) has the same support as the system presented in (7.5), the supports have the same $(G^{\mathcal{A}}, V^{\mathcal{A}}, W^{\mathcal{A}})$-symmetric structure. ■

Next we give an example which demonstrates that symmetry may cause the BKK bound not to be exact as root count.

Example 7.1.4 The construction of a quadrature formula of high accuracy degree leads to a system of algebraic equations. In order that the formula $w_1f(y_1) + w_2f(y_2)$ approximates the integral $\int_{-1}^{1} f(y)dy$ precisely for all polynomials f up to degree three, the unknowns $\mathbf{x} = (w_1, w_2, y_1, y_2)$ have to fulfil

$$F(w_1, w_2, y_1, y_2) = \begin{cases} w_1 & + & w_2 & = & 2 \\ w_1y_1 & + & w_2y_2 & = & 0 \\ w_1y_1^2 & + & w_2y_2^2 & = & \frac{2}{3} \\ w_1y_1^3 & + & w_2y_2^3 & = & 0 \end{cases} . \tag{7.7}$$

System (7.7) is $(Z_2(s_1) \times Z_2(s_2), V^F, W^F)$-symmetric, where $V^F(s_1)\mathbf{x} = (w_2, w_1, y_2, y_1)^T$, $V^F(s_2)\mathbf{x} = (w_1, w_2, -y_1, -y_2)^T$ and $W^F(s_1) = V^F(id)$, $W^F(s_2)(f_1, f_2, f_3, f_4) = (f_1, -f_2, f_3, -f_4)$.

The BKK bound of this system equals four and is not exact. Indeed it is well-known that the system (7.7) has two solutions that are conjugate to each other by $V^F(s_1)$. Note that if all right-hand side coefficients are different from zero, then the BKK bound equals five instead of four. ■

7.2 Symmetric lifting functions

7.2.1 Subdivisions induced by symmetric lifting

In this section the consequences of the symmetry of systems to the Newton polytopes are shown. We demonstrate how to exploit the symmetry by using symmetric lifting functions. The efficiency of using symmetry is demonstrated with an example.

Assume that $\mathcal{A}$ is $(G^{\mathcal{A}}, V^{\mathcal{A}}, W^{\mathcal{A}})$-symmetric. For convenience we introduce

$$\widehat{V^{\mathcal{A}}}(g) = \begin{bmatrix} V^{\mathcal{A}}(g) & 0 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad \widehat{W^{\mathcal{A}}}(g) = \begin{bmatrix} W^{\mathcal{A}}(g) & 0 \\ 0 & 1 \end{bmatrix}, \quad \forall g \in G. \tag{7.8}$$

Definition 7.2.1 An r-tuple of lifting functions ω is called $(G^{\mathcal{A}}, V^{\mathcal{A}}, W^{\mathcal{A}})$-*symmetric* if

$$\widehat{W^{\mathcal{A}}}(g)\widehat{\mathcal{A}} = \widehat{\mathcal{A}} \circ \widehat{V^{\mathcal{A}}}(g), \quad \forall g \in G, \quad \text{where} \quad \widehat{\mathcal{A}} = \omega(\mathcal{A}). \tag{7.9}$$

Example 7.2.2 (Example 7.1.2 continued.) We discuss three different symmetric lifting functions for the system displayed in (7.5) with corresponding induced subdivisions and initial form systems.

Since the two polynomials in (7.5) have the same support we have $r = 1$, $k_1 = 2$, $\omega = (\omega_1 = \omega_2)$ and $\mathcal{A} = (A_1 = A_2)$ with

$$A_1 = \{(2,2),(2,0),(0,2),(1,1),(0,0)\}\,. \tag{7.10}$$

We assume that the vectors in the support are sorted according to the graded lexicographic ordering, so we can represent the lifting function as a sequence of lifting values, whereby the ith lifting value defines the lifting of the ith vector in the support.

By choosing three different lifting functions, represented by $\omega^{(1)} = [1,7,7,8,1]$, $\omega^{(2)} = [7,1,1,8,7]$ and $\omega^{(3)} = [7,7,7,1,7]$, three different triangulations $\triangle^{(i)}$, $i = 1,2,3$, are induced, see Figure 7.1. They are

$$\begin{aligned} \triangle^{(1)} &= \{\{(0,0),(2,2),(2,0)\},\{(0,0),(2,2),(0,2)\}\}\,,\\ \triangle^{(2)} &= \{\{(0,0),(2,0),(0,2)\},\{(2,0),(0,2),(2,2)\}\}\,,\\ \triangle^{(3)} &= \{\{(0,0),(2,0),(1,1)\},\{(0,0),(0,2),(1,1)\}\,,\\ &\qquad \{(2,0),(2,2),(1,1)\},\{(0,2),(2,2),(1,1)\}\}\,. \end{aligned} \tag{7.11}$$

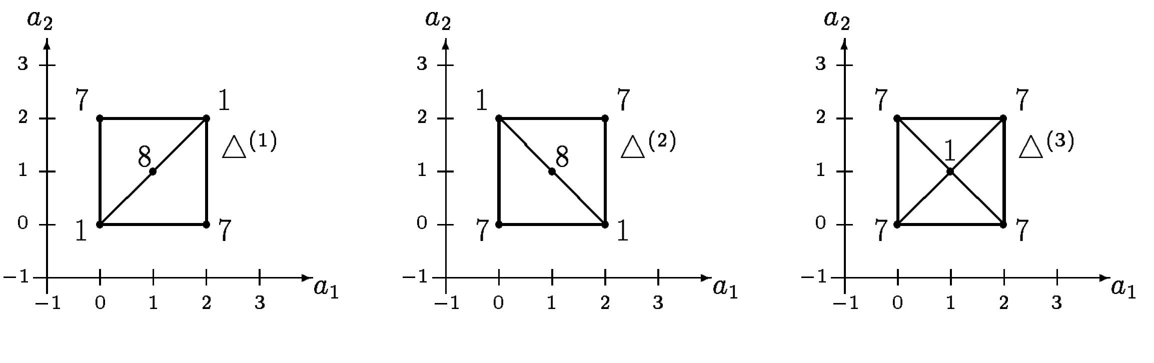


Figure 7.1: The Newton polytope P_1 of system (7.5) with three possible triangulations $\triangle^{(1)}$, $\triangle^{(2)}$ and $\triangle^{(3)}$ which are induced by different lifting functions $\omega^{(1)}, \omega^{(2)}, \omega^{(3)}$.

In $\triangle^{(1)}$ and $\triangle^{(2)}$ the point $(1,1)$ does not appear because the lifting value is greater than the others and it is in the middle. We say that this point was *lifted out*.

The two cells of subdivision $\triangle^{(2)}$ are both of type $(k_1,\ldots,k_r) = (2)$ and thus contribute solutions. The induced initial form systems are

$$F_{\mathbf{v}^{(1)}}(\mathbf{x}) = \left\{ \begin{array}{l} 3x_1^2 + x_2^2 + 5 = 0 \\ x_1^2 + 3x_2^2 + 5 = 0 \end{array} \right. \quad \text{and} \quad F_{\mathbf{v}^{(2)}}(\mathbf{x}) = \left\{ \begin{array}{l} x_1^2x_2^2 + 3x_1^2 + x_2^2 = 0 \\ x_1^2x_2^2 + x_1^2 + 3x_2^2 = 0 \end{array} \right. . \tag{7.12}$$

Each equation of the second system $F_{\mathbf{v}^{(2)}}$ will be divided by x_2^2. After Gaussian elimination the following systems are obtained

$$F'_{\mathbf{v}^{(1)}}(\mathbf{x}) = \begin{cases} 8x_1^2 + 10 = 0 \\ 8x_2^2 + 10 = 0 \end{cases} \quad \text{and} \quad F'_{\mathbf{v}^{(2)}}(\mathbf{x}) = \begin{cases} 2x_1^2 x_2^{-2} - 2 = 0 \\ 2x_1^2 + 8 = 0 \end{cases}. \tag{7.13}$$

They have four solutions each. These solutions are used as start solutions of a homotopy leading to eight solutions of the system F displayed in (7.5).

But the two other subdivisions contain cells leading to initial form systems without solutions. For example the first cell of $\triangle^{(3)}$ leads to a system without solutions:

$$F_{\mathbf{v}^{(1)}}(\mathbf{x}) = \begin{cases} x_1 x_2 + 3x_1^2 + 5 = 0 \\ x_1 x_2 + x_1^2 + 5 = 0 \end{cases}. \tag{7.14}$$

This is no contradiction to Theorem 5.1.9 because they work for almost all choices of coefficients. But system F in (7.5) is $(Z_2(s_1), V^F, V^F)$-symmetric putting restrictions on the coefficients and forcing them to be among the exceptions. This is also shown by observation of system (7.6) that has the same support as (7.5). Here the first cell of $\triangle^{(3)}$ gives the initial form system

$$F_{\mathbf{v}^{(1)}}(\mathbf{x}) = \begin{cases} x_1 x_2 + x_1^2 + 3 = 0 \\ x_1 x_2 + 3x_1^2 + 5 = 0 \end{cases} \tag{7.15}$$

which has two solutions. ■

Although (7.6) is $(Z_2(s_1) \times Z_2(s_2), V^F, W^F)$-symmetric this sort of symmetry does not affect Algorithm 5.3.15. It is the existence of solutions (x_1, x_1) with isotropy $Z_2(s_1)$ in the system given in (7.5) that needs special attention and a modification of Theorem 5.1.9 and a modification of Algorithm 5.3.15.

It is important to note that symmetric lifting functions are often no longer generic. This means that the induced subdivision may fail to be mixed. Even with nonlinear liftings this can occur.

Example 7.2.3 (Example 7.2.2 continued.) The supports $\mathcal{A}$ are $(Z_2(s_1) \times Z_2(s_2), V^{\mathcal{A}}, W^{\mathcal{A}})$-symmetric with $V^{\mathcal{A}}$ as defined in Example 7.1.2. To exploit the symmetry we choose a $(Z_2(s_1) \times Z_2(s_2), V^{\mathcal{A}}, W^{\mathcal{A}})$-symmetric lifting function $\omega = (\omega_1, \omega_2)$ with $\omega_1 = [7, 6, 5, 2, 4]$ and $\omega_2 = [7, 5, 6, 2, 4]$. But the induced subdivision is not mixed. The problem arises because points with a low lifting value have a nonunique representation as sum of points in the lower hull, see Figure 7.2. The nonuniqueness results from the $Z_2(s_1)$-symmetry of the lifting. The lower hull of $P_1 + P_2$ contains eight cells, four of them contribute to the mixed volume, the other four are not mixed. ■

In order to avoid nonmixed regular subdivisions induced by a $(G^{\mathcal{A}}, V^{\mathcal{A}}, W^{\mathcal{A}})$-symmetric lifting functions, the points with nonunique representation should get a sufficiently high lifting value such that they do not belong to the lower hull of the lifted polytope.

Proposition 7.2.4 together with the structure described in Definition 7.2.5 are very helpful for the computation of the subdivision and the computation of the mixed volume even when the support is symmetric but the system F is not.

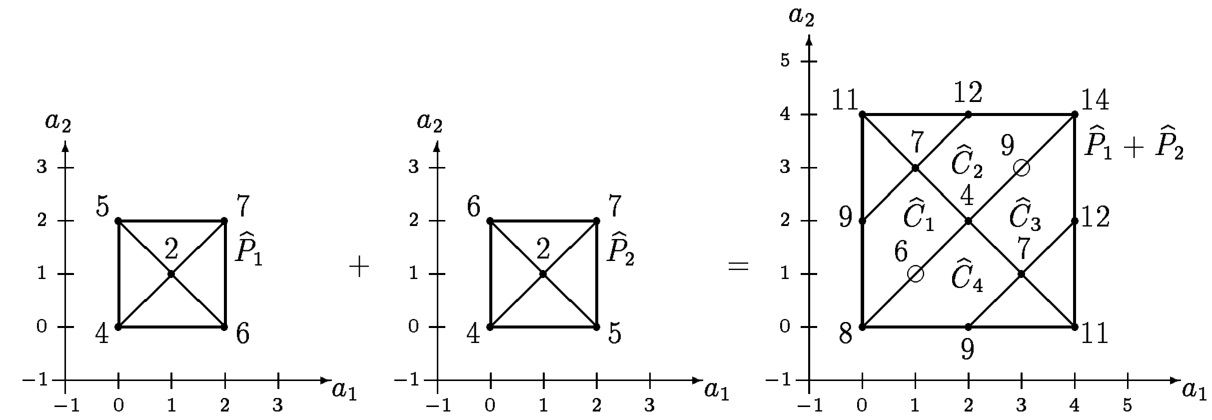


Figure 7.2: Minkowski sum of the lifted Newton polytopes $\widehat{P}_1, \widehat{P}_2$. The vertices with nonunique representation are marked with a circle. This nonuniqueness implies that the cells in the lower hull $\widehat{C}_1, \widehat{C}_2, \widehat{C}_3, \widehat{C}_4$ define a subdivision that is not mixed.

Proposition 7.2.4 *Let $\mathcal{A}$ be $(G, V^{\mathcal{A}}, W^{\mathcal{A}})$-symmetric and let ω be a $(G, V^{\mathcal{A}}, W^{\mathcal{A}})$-symmetric lifting function. Then the cells $\mathcal{C}$ in the subdivision S_ω induced by ω satisfy:*

$$W^{\mathcal{A}}(g^{-1})\mathcal{C} \circ V^{\mathcal{A}}(g) \in S_\omega, \quad \forall g \in G, \forall \mathcal{C} \in S_\omega \tag{7.16}$$

and $(V^{\mathcal{A}}(g)\mathbf{v}, 1)$ is its inner normal, where $[\mathbf{v}, 1]^T$ is the inner normal of $\widehat{\mathcal{C}}$.

Proof. Denote the cell $\mathcal{C}$ with inner normal $[\mathbf{v}, 1]^T$ by $\mathcal{C}^{(1)} = (C_1^{(1)}, C_2^{(1)}, \ldots, C_r^{(1)})$. Then $\langle \mathbf{v}, \mathbf{a}\rangle + \omega_i(\mathbf{a})$ attains its minimum over $\widehat{A}_i$ at $\widehat{C}_i^{(1)}$. Because $\mathcal{A}$ is $(G, V^{\mathcal{A}}, W^{\mathcal{A}})$-symmetric, A_i is permuted to some A_j for some $j \in \{1, 2, \ldots, r\}$ and $C_i^{(1)}$ is permuted to a subset $C_j^{(2)}$ of A_j. Since ω is a symmetric lifting function $\langle V^{\mathcal{A}}(g^{-1})\tilde{\mathbf{a}}, \mathbf{v}\rangle + \omega_j(\tilde{\mathbf{a}})$, where $\tilde{\mathbf{a}} = V^{\mathcal{A}}(g)\mathbf{a}$, $\mathbf{a} \in C_i^{(1)}$, attains its minimum over $\widehat{A}_j$ at $\widehat{C}_i^{(2)}$. The argumentation works for every $i \in \{1, 2, \ldots, r\}$ and the permutation does not change the type of a cell. □

Definition 7.2.5 Let $\mathcal{A}$ and ω be $(G, V^{\mathcal{A}}, W^{\mathcal{A}})$-symmetric, then

$$G_{\mathcal{C}} := \{\ g \in G \mid W^{\mathcal{A}}(g^{-1})\mathcal{C} \circ V^{\mathcal{A}}(g) = \mathcal{C}\ \} \tag{7.17}$$

is *the isotropy group of a cell $\mathcal{C} \in S_\omega$*. If $W^{\mathcal{A}}(g^{-1})\mathcal{C} \circ V^{\mathcal{A}}(g) \neq \mathcal{C}$, then it is called the *conjugate* to $\mathcal{C}$. The conjugates form the *orbit* $\mathcal{O}_{\mathcal{C}}$ and $\mathcal{C}$ is called *a generator.*

Observe that the orbit structure of the cells is very similar to the solution structure.

Proposition 7.2.6 *Assume F is (G, V^F, W^F)-symmetric with $W^F = V^F$ and assume its support $\mathcal{A}$ is $(G, V^{\mathcal{A}}, W^{\mathcal{A}})$-symmetric, where $V^{\mathcal{A}}, W^{\mathcal{A}}$ are induced by V^F. Let S_ω be a subdivision induced by a $(G, V^{\mathcal{A}}, W^{\mathcal{A}})$-symmetric lifting ω. Then the following statements hold.*

(a) *The initial form system $F_{\mathbf{v}}(x) = \mathbf{0}$ of a cell with isotropy G is (G, V^F, V^F)-symmetric.*

(b) *If $\mathcal{C}$ is a cell whose isotropy is not G, then for almost all choices of coefficients in F the corresponding initial form system $F_{\mathbf{v}}(\mathbf{x}) = 0$ is not (G, V^F, V^F)-symmetric and thus in general has no solutions with isotropy G.*

Proof. (a) The support of the initial form system $F_{\mathbf{v}}$ is given by $\mathcal{C}_{\mathbf{v}}$. Since $\mathcal{C}_{\mathbf{v}}$ is assumed to be invariant with respect to $V^{\mathcal{A}}$ and F is (G, V^F, V^F)-symmetric, $F_{\mathbf{v}}$ is also (G, V^F, V^F)-symmetric.
(b) If the isotropy of a cell $\mathcal{C}$ is not G, then there is a $g \in G$ with

$$W^{\mathcal{A}}(g^{-1})\mathcal{C} \circ V^{\mathcal{A}}(g) \neq \mathcal{C}. \tag{7.18}$$

Since $\mathcal{C}$ is the support of the corresponding initial form system, (7.18) implies that $V^F(g^{-1})F_{\mathbf{v}}(V^F(g)\mathbf{x})$ has a support different from $\mathcal{C}$. Thus the system $F_{\mathbf{v}}(\mathbf{x}) = \mathbf{0}$ is not (G, V^F, V^F)-symmetric. In general, systems without symmetry properties are not expected to have solutions with isotropy G or to have conjugate solutions. □

Proposition 7.2.7 *Assume $\mathcal{A}$ with $\mathbf{k}$ the vector of occurrences, $\sum_{i=1}^{r} k_i = n$. Let $V^{\mathcal{A}}$ and $W^{\mathcal{A}}$ be two permutation representations of G such that $\mathcal{A}$ is $(G, V^{\mathcal{A}}, W^{\mathcal{A}})$-symmetric. Let ω be a $(G, V^{\mathcal{A}}, W^{\mathcal{A}})$-symmetric lifting inducing a mixed subdivision S_ω. Then*

$$V_n(\mathcal{P}, \mathbf{k}) = \sum_{\substack{\mathcal{O}_{\mathcal{C}} \\ \mathcal{C} \in S_\omega \\ \mathrm{type}(\mathcal{C}) = \mathbf{k}}} k_1! \cdots k_r! \cdot \mathrm{vol}_n(C, \mathbf{k}) \cdot \#\mathcal{O}_C. \tag{7.19}$$

Example 7.2.8 This example describes the treatment of a family of systems. The general formula for generating the system $F^{(n)}$ is defined as

$$f_i^{(n)}(\mathbf{x}) = \prod_{j=1}^{n} x_j^2 + 2x_i^2 + \sum_{j=1}^{n} x_j^2 + \prod_{j=1}^{n} x_j + 5 = 0, \; i = 1, 2, \ldots, n. \tag{7.20}$$

n	D	B	$V_n(\mathcal{P})$	$=$	$\#C$	$\times$	$\mathrm{vol}_n(C)$
2	16	8	8	$=$	4	$\times$	2
3	216	48	24	$=$	6	$\times$	4
4	4,096	384	64	$=$	8	$\times$	8
5	10,000	3,840	160	$=$	10	$\times$	16
6	2,985,984	46,080	384	$=$	12	$\times$	32
7	105,413,504	645,120	896	$=$	14	$\times$	64
n	$(2n)^n$	$n! \times 2^n$	$n \times 2^n$	$=$	$2n$	$\times$	2^{n-1}

Table 7.1: The total degree D, the optimal n-homogeneous bound B and the BKK bound for the number of solutions of a family of systems.

The system F given in (7.5) in Example 7.1.2 is included as case $n = 2$. For each n the system (7.20) is (S_n, V^F, V^F)-symmetric, where V^F gives the permutation of the variables. In addition the polytopes have Z_2-symmetry by interchanging $(2, 2, \ldots, 2) \leftrightarrow$

$(0, 0, \ldots, 0)$. While the polytope P_1 is a square for $n = 2$ (see Figure 7.1) and has $(Z_2 \times Z_2)$-symmetry the symmetry for arbitrary n is $Z_2 \times S_n$. Thus one can exploit the symmetry of $Z_2 \times S_n$ for the computation of the mixed volume $V_n(\mathcal{P})$. Generalizing the triangulation $\triangle^{(3)}$ in Example 7.2.2 we choose 1 as lifting value for the vertex associated with $x_1 x_2 \cdots x_n$ and 7 for the other vertices. Then the triangulation consists of one orbit. The cell $\{(0, \ldots, 0), (1, \ldots, 1), (2, 0, \ldots, 0), \ldots, (0, \ldots, 0, 2, 0)\}$ has volume $\frac{2^{n-1}}{n!}$. Having isotropy S_{n-1} it generates $2n - 1$ other cells. By formula (7.19) $V_n(\mathcal{P}) = n! \cdot \frac{2^{n-1}}{n!} \cdot (2n)$. Table 7.1 demonstrates that this bound is much better than the other bounds. As n increases, solving the problem based on the total degree D or the n-homogeneous bound B becomes hopeless, where the mixed volume remains exact. ■

7.2.2 Symmetric polyhedral homotopy continuation

Solutions of symmetric systems have a special structure: if $\mathbf{x}$ is a solution then it generates also *conjugate solutions* $V^F(g)\mathbf{x}$ which form the *orbit* $\mathcal{O}_\mathbf{x}$. It suffices to know one *generator* of an orbit. The cardinality of $\mathcal{O}_\mathbf{x}$ depends of course on the *isotropy* $G_\mathbf{x} = \{\, g \in G \mid V^F(g)\mathbf{x} = \mathbf{x} \,\}$ of $\mathbf{x}$. So one distinguishes different types of orbits. Usually one determines solutions with special isotropy G' by restricting to the *fixed-point space* of G'

$$\mathrm{Fix}(G', V^F) = \{\, \mathbf{x} \in \mathbb{C}^n \mid V^F(g)\mathbf{x} = \mathbf{x} \,\} \tag{7.21}$$

and considering the *reduced system* $F^{red}(\mathbf{x}) = \mathbf{0}$,

$$F^{red} : \mathrm{Fix}(G', V^F) \to \mathrm{Fix}(G', W^F), \quad \mathbf{x} \mapsto F(\mathbf{x}). \tag{7.22}$$

Generically, the reduced system is expected to have solutions, if the dimensions satisfy $\dim(\mathrm{Fix}(G', V^F)) \geq \dim(\mathrm{Fix}(G', W^F))$.

Example 7.2.9 (Example 7.1.2 continued.) The system (7.5) may have solutions with trivial isotropy (x_1, x_2), $x_1 \neq x_2$, $x_1 \neq 0$, $x_2 \neq 0$ with conjugates $V^F(s_1)\mathbf{x} = (x_2, x_1)$, $V^F(s_2) = (-x_1, -x_2)$, $V^F(s_1 s_2) = (-x_2, -x_1)$ and solutions with isotropy $Z_2(s_1)$ of type (x_1, x_1), $x_1 \neq 0$ with conjugate $V^F(s_2)\mathbf{x} = (-x_1, -x_1)$ and solutions with isotropy $Z_2(s_1 s_2)$ of type $(x_1, -x_1)$ with conjugate $V^F(s_2)\mathbf{x} = (-x_1, x_1)$. ■

The proof of the following lemma is straightforward.

Lemma 7.2.10 *Assume F is (G^F, V^F, W^F)-symmetric and its support $\mathcal{A}$ is $(G^\mathcal{A}, V^\mathcal{A}, W^\mathcal{A})$-symmetric, with $V^\mathcal{A}$ and $W^\mathcal{A}$ induced by V^F and W^F. If the lifting function ω is $(G^\mathcal{A}, V^\mathcal{A}, W^\mathcal{A})$-symmetric, then the induced polyhedral homotopy $\widehat{F}(\mathbf{x}, t) = \mathbf{0}$ is (G^F, V^F, W^F)-symmetric, i.e.,*

$$\widehat{F}(V^F(g)\mathbf{x}, t) = W^F(g)\widehat{F}(\mathbf{x}, t), \quad \forall g \in G, \forall \mathbf{x} \in \mathbb{C}^n, t \in \mathbb{C}. \tag{7.23}$$

For almost all choices of coefficients in F and almost all $t \in [0, 1]$, the system $\widehat{F}(\mathbf{x}, t) = \mathbf{0}$ has at $t = t_0$ the same solution structure with respect to G as $F(\mathbf{x}) = \mathbf{0}$.

This lemma has important practical relevance. If a solution $\mathbf{x}_0$ of an initial form system $F_{\mathbf{v}}(\mathbf{x}) = \mathbf{0}$ has isotropy $G_{\mathbf{x}_0}$, then we may restrict to the fixed-point space and do path following of $\widehat{F}^{red}(\mathbf{x}, t) = \mathbf{0}$. Secondly, we restrict to generators of an orbit $\mathcal{O}_{\mathbf{x}}$. So the question to be addressed is whether the solution structure of the initial systems corresponding to a subdivision induced by a $(G^{\mathcal{A}},\ V^{\mathcal{A}},\ W^{\mathcal{A}})$-symmetric lifting equals the solution structure of $\widehat{F}(\mathbf{x}, 1) = 0$. The subdivision is invariant and the structure of conjugates and isotropies transports to cells.

A consequence of Proposition 7.2.6 is that the solutions in $\mathrm{Fix}(G, V^F)$ can only be obtained from cells with isotropy G. Secondly, for the (G, V^F, V^F)-symmetric initial form systems the restriction to the fixed-point space can help to solve them. By constructing a $(G, V^{\mathcal{A}}, W^{\mathcal{A}})$-symmetric lifting and an invariant subdivision we should derive as many G-symmetric cells as possible. The same holds for all other isotropy subgroups of G.

Example 7.2.11 (Examples 7.2.2 continued.) Since $V^{\mathcal{A}}(s_2) = W^{\mathcal{A}}(s_2) = id$, all lifting functions are $(Z_2(s_2), V^{\mathcal{A}}, W^{\mathcal{A}})$-symmetric and all cells of the induced subdivision have isotropy $Z_2(s_2)$. In order to get cells with isotropy $Z_2(s_1)$ we consider the following lifting:

$$\begin{aligned} \widehat{\mathcal{A}}_1 &= \{(2,2,7),(2,0,1),(0,2,0),(1,1,8),(0,0,7)\}\,, \\ \widehat{\mathcal{A}}_2 &= \{(2,2,7),(2,0,0),(0,2,1),(1,1,8),(0,0,7)\}\,. \end{aligned} \tag{7.24}$$

The cells of type $(1,1)$ of the induced subdivision have isotropy $Z_2(s_1) \times Z_2(s_2)$:

$$\mathcal{C}_1 = (\{(0,0),(0,2)\}\,, \{(0,0),(2,0)\})\,,\quad \mathcal{C}_2 = (\{(0,2),(2,2)\}\,, \{(2,0),(2,2)\})\,. \tag{7.25}$$

The initial form systems are $(Z_2(s_1) \times Z_2(s_2), V^F, W^F)$-symmetric:

$$F_{\mathbf{v}^{(1)}} = \left\{ \begin{array}{l} x_2^2 + 5 = 0 \\ x_1^2 + 5 = 0 \end{array} \right. \qquad F_{\mathbf{v}^{(2)}} = \left\{ \begin{array}{l} x_2^2 + x_1^2 x_2^2 = 0 \\ x_1^2 + x_1^2 x_2^2 = 0 \end{array} \right. . \tag{7.26}$$

Both systems have the same solution structure: two orbits of different types. There are two solutions with isotropy $Z_2(s_1)$ which are conjugate to each other by $V^F(s_2)$ and two solutions with isotropy $Z_2(s_1 s_2)$ which are conjugate to each other by $V^F(s_1)$. See Figure 7.3 for the symmetric continuation diagram. ■

One may ask the question why not first reduce to fixed-point spaces and then construct the homotopy. In order to find solutions with trivial isotropy this restriction is no effort. Eventually, this restriction can be done during continuation, for tracing the paths starting at solutions that belong to a particular fixed-point space.

7.2.3 The symmetric lifting algorithm

This section gives a characterization of symmetric lifting functions and an algorithm to generate symmetric lifting functions.

In order to classify symmetric lifting functions (see Definition 7.2.1), we describe the orbit of $\mathbf{a} \in A_i$ more precisely by

$$\mathcal{O}_{(\mathbf{a},i)} := \{\ (\mathbf{b}, j) \mid \mathbf{b} = V^{\mathcal{A}}(g)\mathbf{a} \in A_j, j = \widetilde{W}^{\mathcal{A}}(g^{-1})i\ \}, \tag{7.27}$$

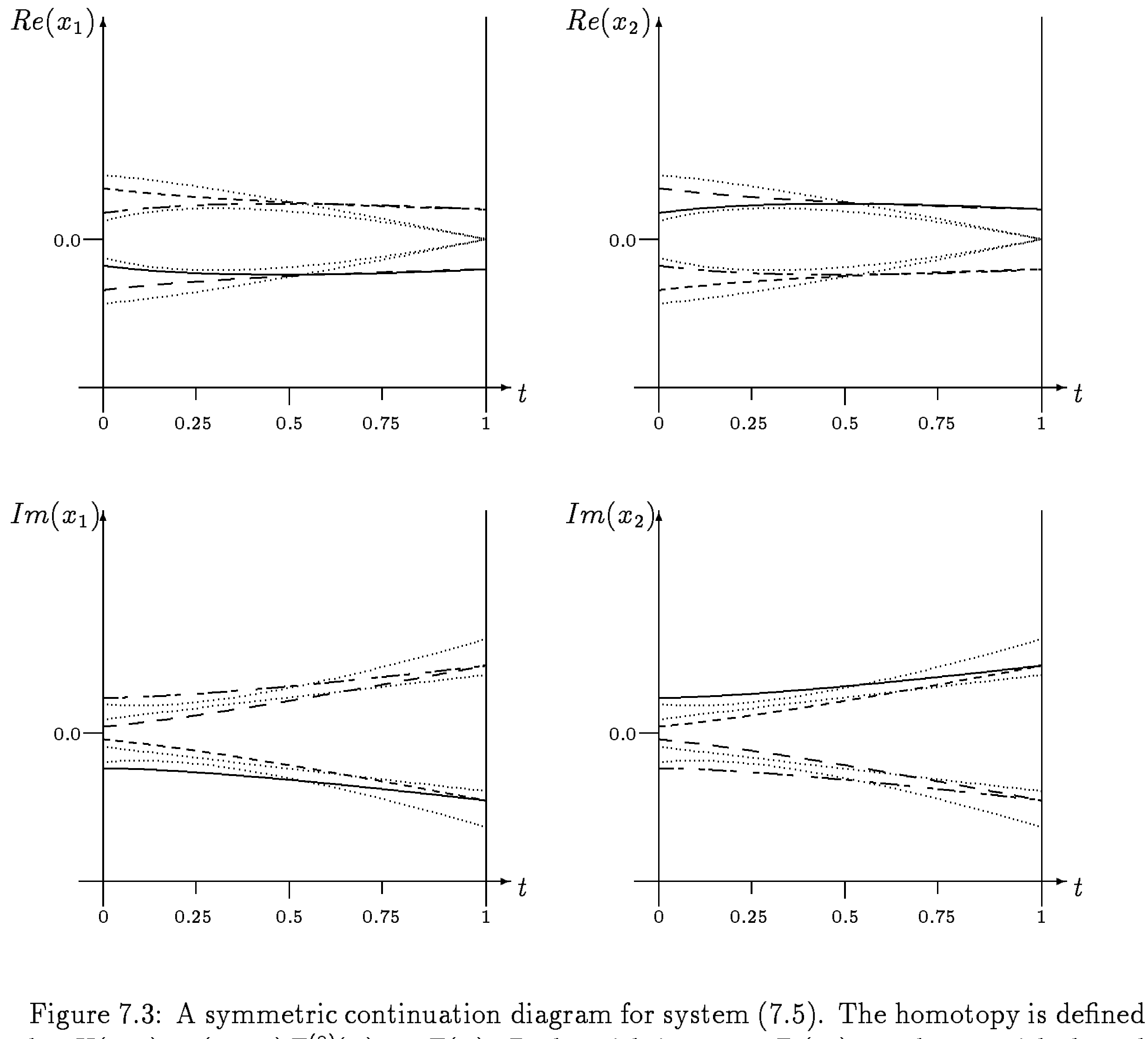


Figure 7.3: A symmetric continuation diagram for system (7.5). The homotopy is defined by $H(\mathbf{x}, t) = (1-t)F^{(0)}(\mathbf{x}) + tF(\mathbf{x})$. Paths with isotropy $Z_2(s_1)$ are drawn with dotted lines. The paths with isotropy $Z_2(s_1 s_2)$ are drawn as a line or dashed.

where $V^{\mathcal{A}}(g)$ describes a permutation of vector components and $\widetilde{W}^{\mathcal{A}}(g^{-1})$ a permutation of indices corresponding the permutation of supports $\mathcal{A}$ by $W^{\mathcal{A}}(g^{-1})$. The isotropy subgroup can then be described as

$$G_{(\mathbf{a},i)} = \{ \ g \in G \mid V^{\mathcal{A}}(g)\mathbf{a} = \mathbf{a} \text{ and } \widetilde{W}^{\mathcal{A}}(g^{-1})i = i \ \}. \tag{7.28}$$

Lemma 7.2.12 *A lifting function ω is $(G^{\mathcal{A}}, V^{\mathcal{A}}, W^{\mathcal{A}})$-symmetric, if and only if for all orbits $\mathcal{O}$ holds: all tuples $(\mathbf{a}, i)$ in $\mathcal{O}$ have the same lifting value $\omega_i(\mathbf{a})$.*

We have to make sure that points in the same orbit receive the same lifting value. Therefore, we first classify the points into orbits and give then all points in the same orbit the same lifting value. See respectively Algorithm 7.2.13 and Algorithm 7.2.14.

Algorithm 7.2.13 Classification of supports into orbits.

Input: $\mathcal{A} = (A_1, A_2, \ldots, A_r)$,	*supports*
$\quad G, V^{\mathcal{A}}, W^{\mathcal{A}}$.	*symmetry group*
Output: $\mathcal{O}$, $m = \#\mathcal{O}$.	*classified supports*
$m := 0$;	*initialize counter for orbits*
$\mathcal{O} := \emptyset$;	*initialize set of all orbits*
$\mathcal{R} := \mathcal{A}$;	*points to be processed*
while $\mathcal{R} \neq (\emptyset, \emptyset, \ldots, \emptyset)$ loop	
$\quad \mathbf{a} \in R_i$;	*select an element*
$\quad m := m + 1$;	*update orbit counter*
$\quad \mathcal{O}_{(\mathbf{a},i)} :=$ orbit$(\mathbf{a}, i, \mathcal{R}, V^{\mathcal{A}}, W^{\mathcal{A}})$;	*compute the orbit $\mathcal{O}_{(\mathbf{a},i)}$*
$\quad \mathcal{O} := \mathcal{O} \cup \mathcal{O}_{(\mathbf{a},i)}$;	*update $\mathcal{O}$*
$\quad \mathcal{R} := \mathcal{R} \setminus \mathcal{O}_{(\mathbf{a},i)}$;	*update $\mathcal{R}$*
end while.	
	with routines:
orbit$(\mathbf{a}, i, \mathcal{R}, V^{\mathcal{A}}, W^{\mathcal{A}})$	*compute the orbit*
$G' :=$ *isotropy*$(\mathbf{a}, i, \mathcal{R}, V^{\mathcal{A}}, W^{\mathcal{A}})$;	
$\mathcal{O}_{(\mathbf{a},i)} :=$ for each left coset $g + G' \in G/G'$ collect	
$\quad \mathbf{b} := V^{\mathcal{A}}(g)\mathbf{a} \in \mathcal{R}^{(j)}$, $j = \widetilde{W}^{\mathcal{A}}(g^{-1})i$	
$\quad (\mathbf{b}, j)$.	
isotropy$(\mathbf{a}, i, \mathcal{R}, V^{\mathcal{A}}, W^{\mathcal{A}})$	*compute isotropy group*
$G' :=$ for each $g \in G$ collect	
$\quad$ if $V^{\mathcal{A}}(g)\mathbf{a} = \mathbf{a}$ and $W^{\mathcal{A}}(g^{-1})i = i$ then g.	

Algorithm 7.2.14 Assignment of lifting values to the orbits.

Input: $\mathcal{A} = (A_1, A_2, \ldots, A_r)$, *supports*
$\mathcal{O}$, $m = \#\mathcal{O}$. *supports classified into orbits*
Output: $\widehat{\mathcal{A}}$. *supports lifted symmetrically*

$w \in \mathbb{Z}^m$; *choose m random integers*
for $k = 1, 2, \ldots, \#\mathcal{O}$ loop *perform the lifting*
  for each $(\mathbf{b}, j) \in \mathcal{O}_{(\mathbf{a},i)}$ loop
    $\widehat{A}_j := \widehat{A}_j \cup \{(\mathbf{b}, \omega_k)\}$; *lift the point*
  end for;
end for.

To ensure a good condition of the solution paths, while path tracking, the coefficients of the start systems should be chosen at random, of course with the preservation of the symmetric structure. The following algorithm provides a possible implementation for generating random coefficients for a start system $F^{(0)}$. The notation $j = \widetilde{W}^{\mathcal{A}}(g)i$ has the same meaning as the $\widetilde{W}^{\mathcal{A}}$ used above. Note that in Algorithm 7.2.15 we ignore the fact that F could be semimixed.

Algorithm 7.2.15 Generating a symmetric random coefficient start system.

Input: $\mathcal{A} = (A_1, A_2, \ldots, A_n)$, *supports of F*
G^F, V^F, W^F. *symmetry group with representations*
Output: $F^{(0)} = (f_1^{(0)}, f_2^{(0)}, \ldots, f_n^{(0)})$. *$(G, V^F, W^F)$-symmetric randomized system*

$\mathcal{R} := \mathcal{A}$; *points to be processed*
$\mathcal{T} := (\emptyset, \emptyset, \ldots, \emptyset)$; *sets of terms*
for all $i \in \{1, 2, \ldots, n\}$ loop
  while $R_i \neq \emptyset$ loop
    $\mathbf{a} \in R_i$; *select a point*
    $c_{\mathbf{a}} \in \mathbb{C}$; *generate random coefficient*
    for each $g \in G/\text{isotropy}(\mathbf{a}, i)$ do *compute its orbit*
      $j = \widetilde{W}^{\mathcal{A}}(g^{-1})i$;
      $T_j := T_j \cup \{c_{\mathbf{a}} \mathbf{x}^{V^F(g)\mathbf{a}}\}$; *update $\mathcal{T}$*
      $R_j := R_j \setminus \{V^F(g)\mathbf{a}\}$; *update $\mathcal{R}$*
    end for;
  end while;
end for;
for all $i \in \{1, 2, \ldots, n\}$ loop *construct $F^{(0)}$*
  $f_i^{(0)} := \displaystyle\sum_{c_{\mathbf{a}}\mathbf{x}^{\mathbf{a}} \in T_i} c_{\mathbf{a}} \mathbf{x}^{\mathbf{a}}$;
end for.

In this algorithm we restrict to the case of permutation representations which immediately transport to the supports. Sign symmetries such as $Z_2(s_2)$ in Example 7.1.2 are automatically fulfilled.

Example 7.2.16 (Example 7.2.8 continued.) As explained in Example 7.2.2, there is only one subdivision of $\mathcal{A} = (A_1)$ suitable for solving the system $F^{(2)}$, i.e.: $\triangle^{(2)}$, with lifting defined by $\omega_2 = [7, 1, 1, 8, 7]$, for $n = 2$. This lifting will now be generalized to general dimensions.

First we discuss $n = 3$ and use the fact that all supports are equal. According to the term order of equation (7.20), choose $\omega_3 = [7, 1, 1, 1, 8, 7]$, which lifts $(1, 1, 1)$ out. The induced subdivision yields the following two initial form systems:

$$F_{\mathbf{v}^{(1)}}(\mathbf{x}) = \begin{cases} 3x_1^2 + x_2^2 + x_3^2 + 5 = 0 \\ x_1^2 + 3x_2^2 + x_3^2 + 5 = 0 \\ x_1^2 + x_2^2 + 3x_3^2 + 5 = 0 \end{cases} \quad \text{and} \quad F_{\mathbf{v}^{(2)}}(\mathbf{x}) = \begin{cases} x_1^2x_2^2x_3^2 + 3x_1^2 + x_2^2 + x_3^2 = 0 \\ x_1^2x_2^2x_3^2 + x_1^2 + 3x_2^2 + x_3^2 = 0 \\ x_1^2x_2^2x_3^2 + x_1^2 + x_2^2 + 3x_3^2 = 0 \end{cases} . \tag{7.29}$$

Since the lifting function generalizes to a S_3-symmetric lifting function $\omega = (\omega_3, \omega_3, \omega_3)$, the two initial form systems are (S_3, V^F, V^F)-symmetric. While S_3 describes the symmetry of the polytopes, $F^{(3)}$ has another symmetry which does not operate on the polytopes, but automatically inherits to $F_{\mathbf{v}^{(1)}}$ and $F_{\mathbf{v}^{(2)}}$. $F^{(3)}$ consists of polynomials which are invariant under the reflections $V^F(s_1) : \mathbf{x} \mapsto (-x_1, -x_2, x_3)$, $V^F(s_2) : \mathbf{x} \mapsto (x_1, -x_2, -x_3)$ and $V_F(s_1s_2) : \mathbf{x} \mapsto (-x_1, x_2, -x_3)$. These reflections together build the group isomorphic to the Kleinian group, denoted by $V_4 = Z_2(s_1) \times Z_2(s_2)$. Thus the symmetry group of $F^{(3)}$ is $S_3 \times Z_2^2$. The V_4-symmetry is inherited to the initial form systems. Additionally, $F_{\mathbf{v}^{(1)}}, F_{\mathbf{v}^{(2)}}$ consists of even polynomials invariant w.r.t. $V^F(s_3) : \mathbf{x} \mapsto -\mathbf{x}$ which is not true for $F^{(3)}$ itself. The first system $F_{\mathbf{v}^{(1)}}$ has exactly 8 solutions. Since the symmetry group of $F_{\mathbf{v}^{(1)}}$ is $S_3 \times Z_2^3$ these eight solutions form one $(S_3 \times Z_2^3)$-orbit generated by one $\mathbf{x}$ with isotropy S_3. Because the symmetry group of $H(\mathbf{x}, t)$ is $S_3 \times Z_2^2$, a proper subgroup of $S_3 \times Z_2^3$, this orbit breaks into two $(S_3 \times Z_2^2)$-orbits during continuation. Two pathfollowings have to be done starting with $\mathbf{x}$ and $-\mathbf{x}$. The second system $F_{\mathbf{v}^{(2)}}$ has 16 solutions, generated by 4 solutions with isotropy S_3. The 16 solutions are arranged in two $(S_3 \times Z_2^3)$-orbits and give rise to four $(S_3 \times Z_2^2)$-orbits of solutions of $F^{(3)}$. Altogether only $6 = 2 \times 3 = 2 + 4$ instead of 24 continuation paths need to be traced.

For general n we state that the symmetry group of $F^{(n)}$ is $S_n \times Z_2^{n-1}$. The liftings ω_2, ω_3 can be generalized into $\omega_n = [7, 1, \ldots, 1, 8, 7]$. The initial form systems (7.29) generalize to two systems with symmetry $S_n \times Z_2^{n-1}$, where $Z_2(s_i)$ is given by the reflection

$$V^F(s_i) : \mathbf{x} \mapsto (x_1, \ldots, x_{i-1}, -x_i, -x_{i+1}, x_{i+1}, \ldots, x_n), \quad i = 1, \ldots, n-1. \tag{7.30}$$

For n even the reflection $\mathbf{x} \mapsto -\mathbf{x}$ is an element of Z_2^{n-1}. For n odd this is an additional sign-symmetry of $F_{\mathbf{v}^{(1)}}$ and $F_{\mathbf{v}^{(2)}}$ which is not valid for $F^{(n)}$ itself. By restriction to fixed-point spaces we see that for n even the solutions of $F_{\mathbf{v}^{(1)}}$ are two $(S_n \times Z_2^{n-1})$-orbits: one is generated by a solution with isotropy S_n and the other by a solution of type $(-x_1, x_1, \ldots, x_1)$. For n odd $F_{\mathbf{v}^{(1)}}$ has two $(S_n \times Z_2^{n-1})$-orbits as solutions, generated by $\mathbf{x}$ and $-\mathbf{x}$ with isotropy S_n. This is one $(S_n \times Z_2^n)$-orbit. Analogously, $F_{\mathbf{v}^{(2)}}$ has $2(n-1)$ $(S_n \times Z_2^{n-1})$-orbits as solutions. Altogether this means that the $n2^n$ solutions of $F^{(n)}$ are generated by $2n$ solutions. ∎

7.3 Symmetric dynamic lifting

The aim is to develop a robust lifting mechanism that ensures that no additional degeneracies will be introduced and that the subdivision is proper. We start by illustrating these two problems that can occur when constructing a symmetric subdivision.

7.3.1 Two examples and two problems

Example 7.3.1 Consider the following symmetric polynomial system:

$$F(\mathbf{x}) = \left\{ \begin{array}{l} x_1^2 x_2^2 + c_{11} x_1^2 x_2 + c_{11} x_1 x_2^2 + c_{12} x_1 + c_{12} x_2 + c_{13} = 0 \\ x_1^2 x_2^2 + c_{21} x_1^2 x_2 + c_{21} x_1 x_2^2 + c_{22} x_1 + c_{22} x_2 + c_{23} = 0 \end{array} \right. , \quad c_{kl} \in \mathbb{C}_0. \tag{7.31}$$

The system is symmetric with respect to the transposition $x_1 \leftrightarrow x_2$. The total degree equals 16, whereas the 2-homogeneous Bézout bound equals 8. Both supports are equal, we denote as usual the Newton polytope by P, see Figure 7.4. The BKK bound equals $V_2(P,P) = 2!\mathrm{vol}_2(P) = 6$.

We assume that the coefficients c_{kl}, for $k = 1,2$ and $l = 1,2,3$, are sufficiently chosen at random. More precisely, we can apply the second theorem of Bernshteĭn to verify the exactness of the BKK bound. In Figure 7.4 we have associated with the faces those polynomial systems that should have a solution in $\mathbb{C}_0^2$ to make the system deficient.

By this example we want to show that a simple classification of the supports, followed by an assignment of lifting values orbit pro orbit, as done by the static symmetric lifting algorithm, can lead to a worthless homotopy. For instance, if the classified support is lifted like

$$\{\{(0,0,1)\}, \{(1,0,0),(0,1,0)\}, \{(2,1,1),(1,2,1)\}, \{(2,2,3)\}\}, \tag{7.32}$$

then we obtain the subdivision of the support shown in Figure 7.4.

We have to avoid to construct such a subdivision, because it has cells that contain the two inner lines drawn in P on Figure 7.4. The simplicial cells in the induced subdivision yield the following initial form systems:

$$\left\{ \begin{array}{l} c_{12}(x_1 + x_2) + c_{13} = 0 \\ c_{22}(x_1 + x_2) + c_{23} = 0 \end{array} \right. \quad \text{and} \quad \left\{ \begin{array}{l} x_1^2 x_2^2 + c_{11}(x_1^2 x_2 + x_1 x_2^2) = 0 \\ x_1^2 x_2^2 + c_{21}(x_1^2 x_2 + x_1 x_2^2) = 0 \end{array} \right. . \tag{7.33}$$

We see that both systems have no finite solutions. So this lifting does not yield a good polyhedral homotopy to compute all solutions. ■

Example 7.3.2 The following example is the well-known system that has to be solved to construct a quadrature formula with two knots (y_1, y_2) and two corresponding weights (w_1, w_2).

$$F(w_1, w_2, y_1, y_2) = \left\{ \begin{array}{rcrcl} w_1 & + & w_2 & = & m_0 \\ w_1 y_1 & + & w_2 y_2 & = & m_1 \\ w_1 y_1^2 & + & w_2 y_2^2 & = & m_2 \\ w_1 y_1^3 & + & w_2 y_2^3 & = & m_3 \end{array} \right. , \quad \text{with } m_k \in \mathbb{R}. \tag{7.34}$$

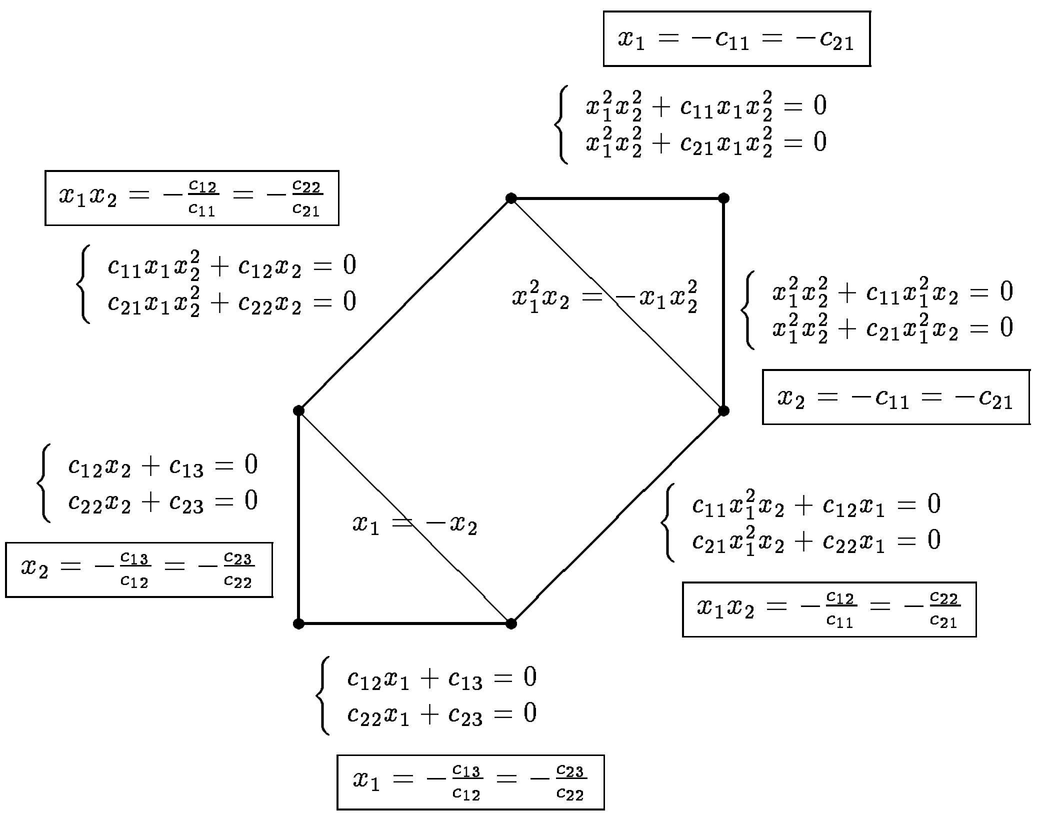


Figure 7.4: A symmetric Newton polytope with face systems. The boxes contain the discriminant conditions on the coefficients under which the face systems have a solution in $\mathbb{C}_0^n$.

The solutions of the system remain invariant when we apply the permutation $(w_1, w_2, y_1, y_2) \mapsto (w_2, w_1, y_2, y_1)$.

To exploit the symmetry we classify the support sets into eight orbits. A polyhedral homotopy induced by a symmetric lifting is presented next.

$$\widehat{F}(w_1, w_2, y_1, y_2, t) = \left\{ \begin{array}{lcrcl} (& w_1 & + & w_2 &)t^1 & = & m_0t^3 \\ (& w_1y_1 & + & w_2y_2 &)t^2 & = & m_1t^4 \\ (& w_1y_1^2 & + & w_2y_2^2 &)t^5 & = & m_2t^7 \\ (& w_1y_1^3 & + & w_2y_2^3 &)t^6 & = & m_3t^8 \end{array} \right. \tag{7.35}$$

We see that the inner product of all points in each support set of the polynomials of $\widehat{F}$ with the normal $\mathbf{v} = [2, 2, 0, 0, 1]^T$ is the same. So there is only one cell in the induced subdivision. ∎

It is most embarrassing when these problems are discovered after a lot of arithmetical work has been wasted on the computation of these useless subdivisions.

7.3.2 Regular symmetric nondegenerate proper subdivisions

We want to construct regular subdivisions that are symmetric, nondegenerate and proper. More precisely, the desired regular symmetric subdivision should satisfy the requirements stated in the following definitions.

Definition 7.3.3 Consider S_ω a regular symmetric subdivision and $\widehat{F}(\mathbf{x}, t) = \mathbf{0}$, the polyhedral homotopy induced by ω. If $\exists \mathcal{C} \in S_\omega$, $\exists \mathbf{v} \neq \mathbf{0}$, such that, $\partial_\mathbf{v} F_\mathcal{C} \neq \partial_\mathbf{v} F$ and $\partial_\mathbf{v} F_\mathcal{C}(\mathbf{x}) = \mathbf{0}$ has a solution in $\mathbb{C}_0^n$ for any symmetric random choice of the coefficients, then S_ω *is degenerate.* Otherwise S_ω *is nondegenerate.*

Proposition 7.3.4 *A subdivision induced by a linear lifting is nondegenerate.*

Proof. By a linear lifting, there is a one-to-one correspondence between the faces of the original and the lifted polytope. This implies that a linear lifting does not introduce additional edges of the lower hull that causes the corresponding initial form systems to be deficient. □

However, to exploit the symmetry, we have to use a nonlinear lifting function with the risk of introducing degeneracies. Linear lifting functions are also useless in the unmixed case.

Proposition 7.3.5 *Consider $\mathcal{A}$ as a fully mixed tuple of sets. Any fine mixed subdivision of $\mathcal{A}$ is nondegenerate.*

Proof. Follows from the fact that every binomial system has exactly as many roots in $\mathbb{C}_0^n$ as its BKK bound. □

Proposition 7.3.6 *Let S_ω be a regular symmetric nondegenerate subdivision. If $V_n(\mathcal{P})$ counts $F(\mathbf{x}) = \mathbf{0}$ exactly, then there are no deficient initial form systems.*

If the conditions of Proposition 7.3.5 or Proposition 7.3.6 are satisfied, then we have the same level of robustness as with the ordinary polyhedral homotopy method.

Definition 7.3.7 Let $\mathcal{A}$ be a tuple of sets. A subdivision S is *improper* if there is only one cell $\mathcal{C} \in S$ such that $V_n(\mathcal{C}) = V_n(\mathcal{A})$. If S contains at least two mixed cells, then S is a *proper subdivision of* $\mathcal{A}$.

Because we are only interested in the mixed cells, we prefer this more general definition of proper, than simply distinguishing the case $\mathcal{C} = \mathcal{A}$. Note that hereby it is not clear whether every improper subdivision is nondegenerate.

For the following theorem, we assume that the orbit we consider is fully mixed, i.e., the n components are independent of each other. We denote this orbit by $\mathcal{O}$, with the origin as fixed point augmented to each component, so that we have $\mathcal{O} = (\{\mathbf{0}, O_1\}, \{\mathbf{0}, O_2\}, \ldots, \{\mathbf{0}, O_n\})$. Let $\mathcal{C} = (\{\mathbf{0}, \mathbf{a}^{(1)}\}, \{\mathbf{0}, \mathbf{a}^{(2)}\}, \ldots, \{\mathbf{0}, \mathbf{a}^{(n)}\})$ so that $\mathbf{a}^{(k)} \in O_k$, for $k = 1, 2 \ldots, n$. Suppose that $V_n(\mathcal{C}) \neq 0$.

Theorem 7.3.8 *To decide whether there exists a lifting function ω such that $\mathcal{C}$ is contained in a proper subdivision of $\mathcal{O}$, it suffices to solve a linear optimization problem of dimension n with m linear inequalities, where $m = \sum_{k=1}^{n}(\#O_k - 1)$. If the system of linear inequalities is feasible, then its solution gives a lifting function and the inner normal to the cell $\widehat{\mathcal{C}}$. Otherwise, no such lifting exists.*

Proof. Without loss of generality, we can give the origin of each component the same lifting value ω_0. Depending on the choice of ω_0, we have to determine n values ω_k of the lifting function for the respective orbits O_k, for $k = 1, 2, \ldots, n$. Equivalently, we want to know the components of the inner normal $\mathbf{v} = [v_1, v_2, \ldots, v_n, 1]^T$ to $\widehat{\mathcal{C}}$. The correspondence between the lifting values ω_k and the components v_k of the inner normal is given by the following system of linear equalities:

$$\sum_{l=1}^{n} a_l^{(k)} v_l + \omega_k = \omega_0, \quad k = 1, 2, \ldots, n. \tag{7.36}$$

The equalities in (7.36) express that $\mathbf{v}$ is a normal vector to the cell $\mathcal{C}$. In order that $\mathbf{v}$ is an inner normal, the following system of inequalities needs to be satisfied:

$$\sum_{l=1}^{n} b_l v_l + \omega_l > \omega_0, \quad \forall \mathbf{b} \in O_k \setminus \{\mathbf{a}^{(k)}\}, \quad k = 1, 2, \ldots, n. \tag{7.37}$$

The strict inequalities ensure that the induced subdivision is proper. We can use the linear equalities in (7.36) to eliminate the unknown lifting the system (7.37). The number of inequalities equals indeed $m = \sum_{k=1}^{n}(\#O_k - 1)$. If the system of linear equalities is feasible, then its solution gives us the inner normal $\mathbf{v}$. By (7.36) we can extract the lifting function. The infeasibility of the linear optimization problem proves that no proper regular subdivision with the cell $\mathcal{C}$ can exist. □

Example 7.3.9 (Example 7.3.2 continued.) Following the notations of Theorem 7.3.8, we write

$$\begin{array}{rl} \widehat{\mathcal{O}} = (& \{\underline{(0,0,0,0,\omega_0)}, \underline{(1,0,0,0,\omega_1)}, (0,1,0,0,\omega_1)\}, \\ & \{\underline{(0,0,0,0,\omega_0)}, \underline{(1,0,1,0,\omega_2)}, (0,1,0,1,\omega_2)\}, \\ & \{\underline{(0,0,0,0,\omega_0)}, (1,0,2,0,\omega_3), \underline{(0,1,0,2,\omega_3)}\}, \\ & \{\underline{(0,0,0,0,\omega_0)}, (1,0,3,0,\omega_4), \underline{(0,1,0,3,\omega_4)}\} \quad). \end{array} \tag{7.38}$$

The underlined vectors are the points in $\mathcal{C}$. We have $V_n(\mathcal{C}) = 1$. In (7.39) we respectively show the system of equalities, the inequality system and the inequalities after elimination of the lifting values.

$$\left\{ \begin{array}{r} v_1 + \omega_1 = \omega_0 \\ v_1 + v_3 + \omega_2 = \omega_0 \\ v_2 + 2v_4 + \omega_3 = \omega_0 \\ v_2 + 3v_4 + \omega_4 = \omega_0 \end{array} \right. \quad \left\{ \begin{array}{r} v_2 + \omega_1 > \omega_0 \\ v_2 + v_4 + \omega_2 > \omega_0 \\ v_1 + 2v_3 + \omega_3 > \omega_0 \\ v_1 + 3v_3 + \omega_4 > \omega_0 \end{array} \right. \quad \left\{ \begin{array}{r} v_2 - v_1 > 0 \\ v_2 + v_4 - v_1 - v_3 > 0 \\ v_1 + 2v_3 - v_2 - 2v_4 > 0 \\ v_1 + 3v_3 - v_2 - 3v_4 > 0 \end{array} \right. \tag{7.39}$$

A solution to this linear inequality system yields the inner normal $\mathbf{v} = [-2, 1, 2, 0, 1]^T$ and lifting $\omega_0 = 1 = \omega_2$, $\omega_1 = 3$ and $\omega_3 = 0 = \omega_4$. ■

By Theorem 7.3.8 we see that all lifting values that induce a particular regular subdivision are contained in a polyhedral cone. All these polyhedral cones of lifting vectors that induce different regular subdivision constitute a fan, the so-called *secondary fan*, which has been introduced by Gel'fand, Kapranov and Zelevinsky, see [56] and [57]. They proved that all regular triangulations of a polytope are in one-to-one correspondence with the vertices of the *secondary polytope*. We refer to [11], [33] and [185].

Concerning the practical elaboration of the linear optimization problem, we can make the following remarks. For polyhedral homotopy continuation, we are particularly interested in an integer solution of the linear inequality system. Because we can scale with the last component of the inner normal, we can first solve the system over the reals, and if feasible, then round the solution to the closest rational vector, using continued fractions (see, e.g., [65, pages 135-136]), where the common denominator equals then the last component v_{n+1} of the inner normal.

We see that it is natural to construct the linear *optimization* problem where we minimize the height of the components of the inner normal, i.e., we try to find a solution with the smallest integer numbers. This optimization problem can be applied to construct well-conditioned polyhedral homotopies, based on a given regular subdivision, as an alternative to the dynamic lifting algorithm. We could call this idea *secondary lifting*, because it is inspired on the secondary fan.

When we have not directly a good choice for an initial cell at hand, we can enumerate all possible candidates, until we obtain a feasible linear optimization problem. In Algorithm 7.3.10 we present the search for an initial mixed cell by exhaustive enumeration. To simplify the notations, we assume that the system is fully mixed.

Algorithm 7.3.10 Construction of an initial mixed cell of a regular symmetric subdivision by exhaustive enumeration of orbits.

Input: $\mathcal{A} = (A_1, A_2, \ldots, A_n)$, $\forall k : \mathbf{0} \in A_k$,	*supports that contain origin*
$\mathcal{O} = (O_1, O_2, \ldots, O_n)$.	*orbits ordered pro component*
Output: fail,	*if fail, then no fine mixed cell*
$\mathcal{C}$, $\mathbf{v}$, ω.	*otherwise, cell with normal and lifting*

Lift($\mathcal{A}$, $\mathcal{O}$,fail,$\mathcal{C}$, $\mathbf{v}$, ω, 1).	*call with $k = 1$*
with procedure Lift($\mathcal{A}$, $\mathcal{O}$,fail,$\mathcal{C}$, $\mathbf{v}$, ω, k):	*auxiliary procedure*

Input: $\mathcal{A}, \mathcal{O}$	*as above*
k, $\mathcal{C}$ with	*current component*
$\dim(C_1, \ldots, C_{k-1}) = k - 1$.	
Output: fail, $\mathcal{C}$, $\mathbf{v}$, ω.	*as above*

At level k, $1 \leq k < n$:	
for each $O \in O_k$ loop	*enumerate orbits*
for all $l \in \{1, 2, \ldots, \#O\}$ loop	*enumerate all points in O*
$\mathbf{a} \in O$;	*pick next point*
$C_k := \{\mathbf{0}, \mathbf{a}\}$;	*update* $\mathcal{C}$
if $\dim(C_1, \ldots, C_k) = k$	*to ensure* $V_n(\mathcal{C}) \neq 0$
then Lift($\mathcal{A}$, $\mathcal{O}$,fail,$\mathcal{C}$, $\mathbf{v}$, ω, $k+1$);	*proceed to next level*
end if;	
return when not fail;	*one mixed cell suffices*
end for;	
end for.	
At level $k = n$:	
Solve linear optimization problem.	*Apply Theorem* 7.3.8

Another simplification made by the algorithm is that we can choose the orbits component after component. For equi-invariant systems, the choice is restricted to the first component, because after choosing an orbit for the first component, also all other components are determined. Note that this algorithm is an alternative to the construction of an initial mixed cell, see Algorithm 6.2.7.

When extending the algorithm to the semimixed and unmixed case, we have to be careful to avoid the construction of a degenerate subdivision. To this end we may not choose cells that have two points of the same orbit and the same polytope that are not connected by an edge of that polytope.

Example 7.3.11 (Example 7.3.1 continued.) In Figure 7.5 we show a triangulation that is not degenerate. It was obtained by choosing $\{(0,0),(2,1),(2,2)\}$ as initial cell. The initial triangulation was thereafter updated by adding the orbit $\{(1,0),(0,1)\}$, after the application of a conservative lifting.

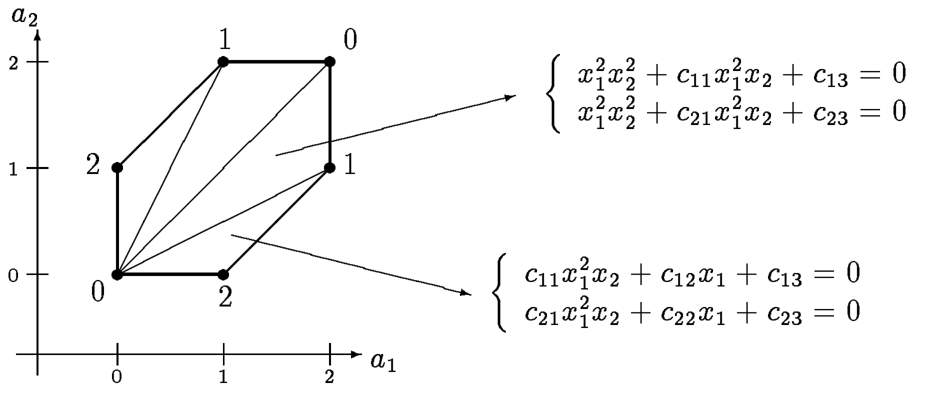


Figure 7.5: The projection of a regular symmetric nondegenerate triangulation of the Newton polytope with generating initial form systems.

It suffices to follow the paths starting at the solutions of the generating initial form system to compute all generators of the solution set of $F(\mathbf{x}) = \mathbf{0}$. ∎

7.3.3 Symmetric dynamic lifting

The symmetric dynamic lifting algorithm is presented as Algorithm 7.3.12. Again we assume that the system is fully mixed. Before starting this algorithm we need to have computed all mixed cells that have been obtained after the initial assignment of the lifting.

Algorithm 7.3.12 The symmetric dynamic lifting algorithm.

Input: $\widehat{\mathfrak{S}}_\omega$, $\widehat{\mathcal{A}}$,	*mixed cells and lifted points*
$(\widehat{\mathcal{F}}_1, \widehat{\mathcal{F}}_2, \dots, \widehat{\mathcal{F}}_n)$,	*edges of lower hulls of* $\widehat{\mathcal{A}}$
$\mathcal{O}$.	*remaining orbits*
Output: $\widehat{\mathfrak{S}}_\omega$, $\widehat{\mathcal{A}}$,	*a regular symmetric subdivision*
$(\widehat{\mathcal{F}}_1, \widehat{\mathcal{F}}_2, \dots, \widehat{\mathcal{F}}_n)$.	*edges of lower hulls of* $\widehat{\mathcal{A}}$
for all $O \in \mathcal{O}$ loop	*enumerate orbits*
$\widehat{O} := \omega(\widehat{\mathfrak{S}}_\omega, O)$;	*lift the orbit*
for all $k \in \{1, 2, \dots, n\}$ loop	
$\widehat{\mathcal{F}_k^O} := New\_Edges(\widehat{A}_k, \widehat{O}_k)$:	*compute new edges of lower hull*
$\widehat{\mathfrak{S}_\omega^O} := Algorithm\_5.3.29(\widehat{\mathcal{F}}_1, \dots, \widehat{\mathcal{F}_k^O}, \dots, \widehat{\mathcal{F}}_n)$;	*compute new mixed cells*
$\widehat{\mathfrak{S}}_\omega := \widehat{\mathfrak{S}}_\omega \cup \widehat{\mathfrak{S}_\omega^O}$;	*update mixed cells*
$\widehat{A}_k := \widehat{A}_k \cup \widehat{O}_k$; $\widehat{\mathcal{F}}_k := \widehat{\mathcal{F}}_k \cup \widehat{\mathcal{F}_k^O}$;	*update lifted points and edges*
end for;	
end for.	

This algorithm is a straightforward adaptation of the dynamic lifting algorithm to compute the mixed cells in a placeable subdivision.

Algorithm 7.3.12 is certainly too primitive to be able to ensure that all degeneracies can be avoided. What helps is the *random* enumeration of all orbits. Another crucial step in the algorithm is the lifting of the orbit. A good idea is to lift all points in the orbit conservatively w.r.t. all cells in $\widehat{\mathfrak{S}}_\omega$, although we cannot ensure that all mixed cells will be fine mixed.

A more sophisticated version of Algorithm 7.3.12 requires the use of linear programming models to ensure a safer lifting strategy, so that degeneracies can be avoided. Perhaps a repetitive use of Algorithm 7.3.10 with additional constraints that represent the preservation of the already computed mixed cells might prove its effectiveness. Finally, we can mention that the Cayley trick can be worked out so that it constructs symmetric fine mixed subdivisions.

7.4 Applications

There is an ubiquity of practical applications that yield symmetric polynomial systems. The examples discussed in this section were found in the literature, see [13, 53, 119], were communicated to us electronically or can be found in the POSSO benchmark suite.

7.4.1 The polynomial systems

For each of the examples, the BKK bound has been computed followed by the calculation of the solutions by means of a symmetric coefficient homotopy. In the first example, we show how product homotopies can be combined with polyhedral homotopies. For some examples we were able to exploit the deficiency.

The system of E.R. Speer

The following system has been given by E.R. Speer and in [53] the Gröbner basis has been computed.

$$F(\mathbf{x}) = \left\{ \begin{array}{l} 4\beta(n+2a_1-8x_1)(a_2-a_3) - x_2x_3x_4 + x_2 + x_4 = 0 \\ 4\beta(n+2a_1-8x_2)(a_2-a_3) - x_1x_3x_4 + x_1 + x_3 = 0 \\ 4\beta(n+2a_1-8x_3)(a_2-a_3) - x_1x_2x_4 + x_2 + x_4 = 0 \\ 4\beta(n+2a_1-8x_4)(a_2-a_3) - x_1x_2x_3 + x_1 + x_3 = 0 \quad , \end{array} \right. \tag{7.40}$$

where $a_1 = x_1 + x_2 + x_3 + x_4$, $a_2 = x_1x_2x_3x_4$, $a_3 = x_1x_2 + x_2x_3 + x_3x_4 + x_4x_1$ and where β and n are parameters to the system. Here we consider them as complex constants to be chosen at random. The system is (D_4, V^F, V^F)-symmetric, where $D_4 = \{id, r, r^2, r^3, s, sr, sr^2, sr^3\}$ represented by $V^F(r)\mathbf{x} = (x_4, x_1, x_2, x_3)$ and $V^F(s)\mathbf{x} = (x_2, x_1, x_4, x_3)$. We restrict to $Z_2(s) \times Z_2(sr^2) = \{id, s, sr^2, r^2\} \subset D_4$, where $V^F(r^2)\mathbf{x} = (x_3, x_4, x_1, x_2)$, $V^F(sr^2)\mathbf{x} = (x_4, x_3, x_2, x_1)$.

Application of Bézout's Theorem gives 625 as the total degree of this system. This means that with the traditional homotopy, 625 continuation paths have to be traced. The lowest m-homogeneous Bézout number ([163]) is obtained with a 4-homogenization, yielding $B_Z = 384$. In [164], a $(Z_2(s) \times Z_2(sr^2))$-symmetric homotopy could be constructed. Since one restricted to nonconjugates, there were only 83 of a total of 271 paths to trace. Theorem 5.1.9 gives the sharper BKK bound 96.

A symmetric lifting could be chosen, but the system has a lot of terms, which makes the construction of the mixed subdivision quite lengthy. Therefore it is better to exploit the product structure in the system. So we consider the start system

$$F^{(0)}(\mathbf{x}) = \left\{ \begin{array}{c} (1+2b_1(V^F(id)\mathbf{x}) - 8x_1)(a_2 - b_3(V^F(id)\mathbf{x})) = 0 \\ (1+2b_1(V^F(s)\mathbf{x}) - 8x_2)(a_2 - b_3(V^F(s)\mathbf{x})) = 0 \\ (1+2b_1(V^F(sr^2)\mathbf{x}) - 8x_3)(a_2 - b_3(V^F(sr^2)\mathbf{x})) = 0 \\ (1+2b_1(V^F(r^2)\mathbf{x}) - 8x_4)(a_2 - b_3(V^F(r^2)\mathbf{x})) = 0 \quad , \end{array} \right. \tag{7.41}$$

where $b_1(\mathbf{x}) = c_{11}x_1 + c_{12}x_2 + c_{13}x_3 + c_{14}x_4$
and $b_3(\mathbf{x}) = c_{31}x_1x_2 + c_{32}x_2x_3 + c_{33}x_3x_4 + c_{33}x_4x_1 + c_{35}$,
with all coefficients c_{kl} randomly chosen complex constants. Observe that $F^{(0)}$ is $(Z_2(s)\times$

$Z_2(sr^2), V^F, V^F$)-symmetric. Since the Newton polytopes of F are contained in those of $F^{(0)}$, one can be sure that the BKK bound of $F^{(0)}$ is greater or equal to the BKK bound of F. This application shows how polyhedral and product homotopies can be applied in a symmetry preserving way.

type	#systems in $F^{(0)}$	#generating systems	BKK bound	#generated solutions
0	1	1	1	1
1	4	1	4	16
2	6	3	3×8	48
3	4	1	8	32
4	1	1	0	0
Total:	16	7	37	97

Table 7.2: Solving the subsystems of (7.41): The number of nonlinear equations, number of systems, the BKK bound of the generating system, and the number of generated solutions.

Because of the product structure the solution of (7.41) simplifies to the solution of subsystems. The subsystems are here nonlinear, but a lot sparser than the original system, which means that the BKK bound can be computed faster. The BKK bound of F' is the sum of the BKK bounds of the subsystems yielding 97. The augmentation of the BKK bound with one is due to the addition of the point $(0,0,0,0)$ to the Newton polytopes of F in order to exploit its product structure. There are 16 subsystems, which can be divided into five groups, according to their type. A subsystem is said to be of type k if it contains k nonlinear equations. Only seven subsystems need to be considered, due to symmetry. Table 7.2 lists the characteristics of the subsystems.

Note that the subsystems should be considered as fully mixed, in order to have nondegenerate initial form systems which correspond to the cells in their induced subdivision. The last subsystem deserves some special attention. Although its BKK bound equals zero, it has solutions in $\mathbb{C}_0^n$, due to the symmetric choice of the coefficients. The 4-homogeneous Bézout bound equals 24. These 3×8 solutions of the system can be found by considering the restriction to the following fixed point spaces: (x_1, x_1, x_3, x_3), (x_1, x_2, x_1, x_2) and (x_1, x_2, x_2, x_1), as each restriction yields 8 solutions. Hence, the last subsystem only contributes to solutions in fixed point spaces, which can be computed separately, by directly putting the restrictions on the original system F. So, by considering the generating solutions of $F^{(0)}$ which lie not in any fixed point space, only 10 instead of 96 paths need to be traced.

A symmetrized four-bar mechanism

The following application [119] leads to a system for which the supports are symmetrical, but the system is not: $F = (f_1, f_2, f_3, f_4)$ with

$$\begin{aligned} f_l(\mathbf{x}) \;\; = \;\; & a_{l1}x_1^2x_3^2 + a_{l2}x_1^2x_3x_4 + a_{l3}x_1^2x_3 + a_{l4}x_1^2x_4^2 + a_{l5}x_1^2x_4 \\ & + a_{l6}x_1^2 + a_{l7}x_1x_2x_3^2 + a_{l8}x_1x_2x_3x_4 + a_{l9}x_1x_2x_3 + a_{l10}x_1x_2x_4^2 \\ & + a_{11}x_1x_2x_4 + a_{l12}x_1x_3^2 + a_{l13}x_1x_3x_4 + a_{l14}x_1x_3 + a_{l15}x_1x_4^2 \\ & + a_{l16}x_1x_4 + a_{l17}x_2^2x_3^2 + a_{l18}x_2^2x_3x_4 + a_{l19}x_2^2x_3 + a_{l20}x_2^2x_4^2 \\ & + a_{l21}x_2^2x_4 + a_{l22}x_2^2 + a_{l23}x_2x_3^2 + a_{l24}x_2x_3x_4 + a_{l25}x_2x_3 \\ & + a_{l26}x_2x_4^2 + a_{l28}x_2x_4 + a_{l28}x_3^2 + a_{l29}x_4^2, \qquad l = 1, \ldots, 4. \end{aligned} \tag{7.42}$$

This polynomial system expresses the geometric constrains of the precision-point problem, described in [119]. The total degree equals 256, the BKK bound equals 80. Note that the optimal m-homogeneous equals 96. When we add a constant nonzero coefficient to each polynomial, then the BKK bound of the augmented system equals also 96. So, we can say that the zero solution counts for 16 and that we have avoided computing it, by applying a polyhedral homotopy. For general choices of the parameters of the system, there are only 36 finite regular solutions and two planes of solutions which are not at infinity.

The polytopes are D_2-symmetric, $D_2 = \{id, s, r, rs\}$, represented by

$$V^F(s)\mathbf{x} = (x_3, x_4, x_1, x_2) \quad \text{and} \quad V^F(r)\mathbf{x} = (x_4, x_3, x_2, x_1). \tag{7.43}$$

A symmetric start system $F^{(0)}$ has been chosen with the first polynomial

$$f_1^{(0)}(\mathbf{x}) = a'_{11}x_1^2x_3^2 + a'_{12}x_1^2x_4^2 + a'_{13}x_2^2x_3^2 + a'_{14}x_2^2x_4^2 + a'_{15}x_1^2 + a'_{15}x_2^2 + a'_{16}x_3^2 + a'_{16}x_4^2 = 0. \tag{7.44}$$

By application of the group actions, the other equations of $F^{(0)}$ can be constructed. Note that the monomials whose corresponding vector of exponents belongs to the interior of the polytope have been left out. They have been lifted out. The system $F^{(0)}$ has as many regular solutions as the BKK bound. Not only is $F^{(0)}$ sparser and hence easier to solve, but there is also the additional sign symmetry which makes it easier to solve. In other words, the group Z_2^4 generated by the reflections which change the sign of one variable is acting on $F^{(0)}$, but not on F. For the solution of $F^{(0)}$, it is helpful to consider the polytopes to be semimixed, $r = 2$, with $k_1 = k_2 = 2$, so that the cells in the induced subdivision lead to nondegenerate initial form systems. By the exploitation of all the symmetry, only three paths were to follow to solve $F^{(0)}$. During continuation of the paths defined by $H(\mathbf{x}, t) = (1 - t)F^{(0)}(\mathbf{x}) + tF(\mathbf{x})$ all 80 solution paths need to be followed, as only the supports of F have D_2-symmetry and the system F is not symmetric.

An application from neurofysiology

The following system has been posted by Sjirk Boon to the newsgroups sci.math.num-analysis and sci.math.symbolic. It is derived from an neurofysiology application.

$$F(\mathbf{x}) = \begin{cases} x_1^2 + x_3^2 - 1 = 0 \\ x_2^2 + x_4^2 - 1 = 0 \\ x_5x_3^3 + x_6x_4^3 - c_1 = 0 \\ x_5x_1^3 + x_6x_2^3 - c_2 = 0 \\ x_5x_3^2x_1 + x_6x_4^2x_2 - c_3 = 0 \\ x_5x_3x_1^2 + x_6x_4x_2^2 - c_4 = 0 \ . \end{cases} \tag{7.45}$$

The total degree equals 1024, whereas the BKK bound equals only 20. The parameters of the system are c_k, $k = 1, 2, 3, 4$. We observe that for general values of the parameters c_k, there are only eight finite solutions were found. This deficiency is due to the fact that the face system $F_{\mathbf{v}}$ with inner normal $\mathbf{v} = [-1, -1, -1, -1, -1, -1]^T$

$$F_{\mathbf{v}}(\mathbf{x}) = \begin{cases} x_1^2 + x_3^2 = 0 \\ x_2^2 + x_4^2 = 0 \\ x_5x_3^3 + x_6x_4^3 = 0 \\ x_5x_1^3 + x_6x_2^3 = 0 \\ x_5x_3^2x_1 + x_6x_4^2x_2 = 0 \\ x_5x_3x_1^2 + x_6x_4x_2^2 = 0 \end{cases} \tag{7.46}$$

has BKK bound equal to zero, but has solutions $(y_1, y_2, iy_1, iy_2, y_3, -y_3)$, $y_1, y_2, y_3 \in \mathbb{C}$, $i^2 = -1$. This is an illustration of Bernshteĭn's second theorem, see Theorem 5.1.17. The type of deficiency is analogous to that of the system presented in Example 7.1.4.

We consider here the symmetric version of this problem. By taking $c_1 = c_2$ and $c_3 = c_4$, the system F becomes $(D_2(s_1, r_1) \times D_2(s_2, r_2))$-symmetric, with $D_2(s_k, r_k) = \{id, s_k, r_k, r_ks_k\}$, $k = 1, 2$ represented by

$$V^F(s_1)\mathbf{x} = (x_2, x_4, x_1, x_3, x_6, x_5) \quad V^F(r_1)\mathbf{x} = (x_3, x_1, x_4, x_2, x_5, x_6), \tag{7.47}$$

and

$$V^F(s_2)\mathbf{x} = (-x_1, x_2, -x_3, x_4, -x_5, x_6) \quad V^F(r_2)\mathbf{x} = (x_1, -x_2, x_3, -x_4, x_5, -x_6). \tag{7.48}$$

Application of the symmetric lifting algorithm leads to six orbits. The symmetric subdivision has three cells with respective volumes 12, 4 and 4. The last two are conjugates, so at most 16 paths are to be followed. By exploitation of the symmetry, only three paths need to be computed for solving the randomized system $F^{(0)}$. It turns out that the first subsystem has no finite solutions, while the second system has one generating solution. This generating solution can then be used to compute the generating solution of $F(\mathbf{x}) = \mathbf{0}$.

To solve the original nonsymmetrical problem, one can use the symmetric system as start system. In this sense, a parameter-homotopy or cheater's homotopy has been constructed.

The cyclic 5-roots problem

The following system belongs to a family of systems, the so-called cyclic n-roots problem, given in [13]:

$$F(\mathbf{x}) = \begin{cases} x_1 + x_2 + x_3 + x_4 + x_5 = 0 \\ x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_1 = 0 \\ x_1x_2x_3 + x_2x_3x_4 + x_3x_4x_5 + x_4x_5x_1 + x_5x_1x_2 = 0 \\ x_1x_2x_3x_4 + x_2x_3x_4x_5 + x_3x_4x_5x_1 + x_4x_5x_1x_2 + x_5x_1x_2x_3 = 0 \\ x_1x_2x_3x_4x_5 - 1 = 0 \end{cases} . \tag{7.49}$$

It is a notorious test problem for Gröbner basis computation although the Buchberger algorithm is not able to exploit the symmetry. The system remains invariant when the unknowns are permuted in a cyclic way and when they are read backwards. More precisely, the system is D_5-invariant, where D_5 is the dyhedral group spanned by r and s, $D_5 = \{id, r, r^2, r^3, r^4, s, rs, rs^2, rs^3, rs^4\}$. The generators of D_5 can be represented as follows:

$$V^F(r) : \mathbb{C}^5 \to \mathbb{C}^5 : \mathbf{x} \mapsto (x_2, x_3, x_4, x_5, x_1) \tag{7.50}$$

and

$$V^F(s) : \mathbb{C}^5 \to \mathbb{C}^5 : \mathbf{x} \mapsto (x_5, x_4, x_3, x_2, x_1). \tag{7.51}$$

So the name n-cyclic is misleading ignoring part of the symmetry. The total degree equals 120. A generalized Bézout number based on a set structure, see [163], equals 108. The BKK bound is 70, which matches the number of finite solutions.

The exploitation of the symmetry requires the construction of a D_5-invariant homotopy. A problem occurs when the lifting algorithm is applied to F, as all lifting values for one polynomial are the same and no proper mixed subdivision is obtained. In order to exploit this kind of symmetry we add points which are invariant w.r.t. D_5 and do not change the mixed volume. For the kth polynomial $(x_1x_2x_3x_4x_5)^{\frac{k}{5}}$ is added, $k = 1, 2, 3, 4$. Because the Newton polytopes are left unchanged, the BKK bound of this new system will remain the same. However, the new system is not a polynomial system anymore. The transformation $y_k = x_k^5$, $k = 1, 2, \ldots, 5$, is not used because it introduces new solutions and blows up the BKK bound.

A special trick is used which exploits the special structure of the system. From the last equation $x_1x_2x_3x_4x_5 = 1$ one obtains $(x_1x_2x_3x_4x_5)^{\frac{k}{5}} = \gamma$, $k = 1, 2, 3, 4$, where γ is the fifth root of unity. Replacing the added terms $(x_1x_2x_3x_4x_5)^{\frac{k}{5}}$ by γ^k fortunately leaves the BKK bound unchanged. The start system for the cyclic n-roots problem is then

$$F^{(0)}(\mathbf{x}) = \begin{cases} x_1 + x_2 + x_3 + x_4 + x_5 - \gamma = 0 \\ x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_1 - \gamma^2 = 0 \\ x_1x_2x_3 + x_2x_3x_4 + x_3x_4x_5 + x_4x_5x_1 + x_5x_1x_2 - \gamma^3 = 0 \\ x_1x_2x_3x_4 + x_2x_3x_4x_5 + x_3x_4x_5x_1 + x_4x_5x_1x_2 + x_5x_1x_2x_3 - \gamma^4 = 0 \\ x_1x_2x_3x_4x_5 - 1 = 0 \end{cases} , \tag{7.52}$$

where γ is not treated as a variable, but as a constant. This can be interpreted as adding $(0, 0, 0, 0, 0)$ to the supports A_k, for $k = 1, 2, 3, 4$.

For the start system a D_5-invariant mixed subdivision can be constructed. The static lifting has been taken as follows: [17 17 17 17 17 1] for the terms is the first four equations (1 for the constant term) and [13 7] for the last equation. The resulting mixed subdivision contains 40 cells and is D_5-invariant. Table 7.3 lists the generating inner normals, together with the volume of the cell and the number of generated solutions.

A D_5-invariant homotopy is constructed. In order to know the solutions of F, it is sufficient to follow the paths in the homotopy $H(\mathbf{x}, t) = (1-t)F^{(0)}(\mathbf{x}) + tF(\mathbf{x})$, for $t : 0 \rightarrow 1$, starting at the representatives of orbits of solutions of $F^{(0)}$. One orbit, the classical solution $(1, \gamma, \gamma^2, \gamma^3, \gamma^4)$ is already known. By a change of the right hand side of $F^{(0)}$ into $(1, \gamma, \gamma^3, \gamma, 1)$, we have the subsystem associated with the first cell equal to

$$\left\{ \begin{array}{rcrcl} x_1 & - & 1 & = & 0 \\ x_1 x_2 & - & \gamma & = & 0 \\ x_1 x_2 x_3 & - & \gamma^3 & = & 0 \\ x_1 x_2 x_3 x_4 & - & \gamma & = & 0 \\ x_1 x_2 x_3 x_4 x_5 & - & 1 & = & 0 \end{array} \right. , \qquad (7.53)$$

which has already this classical solution. Hence, by exploitation of the symmetry, only six instead of 70 continuation paths need to be traced.

generating normal	$\mathrm{vol}_5(C)$	$\#\mathcal{O}_C$	$\mathrm{vol}_5(C) \times \#\mathcal{O}_C$
$(-16, 10, 0, 0, 0, 1)$	1	10	10
$(-16, 0, 10, 0, 0, 1)$	3	10	30
$(-16, 10, -10, 10, 0, 1)$	1	10	10
$(-16, 13, -3, -13, 13, 1)$	2	10	20
Total:	7		70

Table 7.3: The orbit structure of the 5-cyclic subdivision. The normals of four cells are given, one for each orbit of cells of the D_5-invariant subdivision. Secondly, their volume, order of the orbit, and the number of generated initial solutions are contained.

By application of Algorithm 7.3.10, we find [5 7 9 10 9] as respective lifting values for the orbits different from the fixed-point. The induced symmetric subdivision has forty cells. Considering only the generating cells, we have to trace seven solution paths to solve a symmetric random coefficient start system.

Although the addition of a constant helps exploiting the symmetry, this enlargement augments the complexity of computing the mixed volume. Therefore, for general dimensions, it is better to consider the reduced cyclic n-roots problem.

The reduced cyclic n-roots problem

The reduced cyclic 5-roots problem (see [43, 44]) is given by

$$F(\mathbf{y}) = \left\{ \begin{array}{r} 1 + y_1 + y_2 + y_3 + y_4 = 0 \\ y_1 + y_1 y_2 + y_2 y_3 + y_3 y_4 + y_4 = 0 \\ y_1 y_2 + y_1 y_2 y_3 + y_2 y_3 y_4 + y_3 y_4 + y_4 y_1 = 0 \\ y_1 y_2 y_3 + y_1 y_2 y_3 y_4 + y_2 y_3 y_4 + y_3 y_4 y_1 + y_4 y_1 y_2 = 0 \end{array} \right. . \qquad (7.54)$$

The system is invariant when we consider the unknowns in reverse order, i.e., under the permutation $\mathbf{y} \mapsto (y_4, y_3, y_2, y_1)$. The interesting aspect of this reduced problem is that the mixed volume is 5 times less, 14 instead of 70, which reduces the cost needed for tracing the paths. In general, the mixed volume drops by a factor equal to n.

When we want to exploit the symmetry of the original cyclic n-roots problem, we have to add to each equation a random constant, which makes that the system has more monomials and hence the the mixed volume is harder to compute, although the BKK bound is the same. With the reduced problem, we can exploit the symmetry at low cost. In (7.55) we display the polyhedral homotopy induced by a symmetric lifting. The coefficients c_{ij}, $i = 1, \ldots, 4$, $j = 1, 2, 3$, are randomly chosen complex numbers.

$$\widehat{F}(\mathbf{y}, t) = \left\{ \begin{array}{r} c_{11}t + c_{12}y_1t^7 + c_{13}y_2t^5 + c_{13}y_3t^5 + c_{12}y_4t^7 = 0 \\ c_{21}y_1t^5 + c_{22}y_1y_2t^7 + c_{23}y_2y_3t + c_{22}y_3y_4t^7 + c_{21}y_4t^5 = 0 \\ c_{31}y_1y_2t^5 + c_{32}y_1y_2y_3t^7 + c_{32}y_2y_3y_4t^7 + c_{31}y_3y_4t^5 + c_{33}y_4y_1t = 0 \\ c_{41}y_1y_2y_3t^7 + c_{42}y_1y_2y_3y_4t + c_{41}y_2y_3y_4t^5 + c_{43}y_3y_4y_1t^5 + c_{43}y_4y_1y_2t^7 = 0 \end{array} \right. \tag{7.55}$$

To solve $\widehat{F}(\mathbf{y}, 1) = \mathbf{0}$, we only have to follow seven solution paths, which matches the number of generating solutions.

In Table 7.4, we list the characteristics of the reduced cyclic n-roots problem and report on the cardinalities of the cells in the symmetric subdivision.

n	#cells	#gen. cells	$V_n(\mathcal{P})$	#paths		$N(F)$
4	14	7	14	7	7	14
5	22	12	26	15	13	26
6	118	59	132	66	66	132
7	267	139	320	170	160	144

Table 7.4: Characteristics for the reduced cyclic n-roots problem. For $n = 5$ up to 8 we list the number of cells and the number of generating cells in the symmetric subdivision. Furthermore, we give the mixed volume $V_n(\mathcal{P})$, the number of paths to trace to solve the symmetric start system and the target system. Finally, the number of regular solutions is displayed.

From Table 7.4 we see that the subdivision has a fine structure, i.e., the number of cells equals nearly the number of solutions and the volume of most cells equals one. We also see that the subdivision yields an efficient homotopy, that the number of paths to follow corresponds greatly to the number of generating solution to the symmetric random coefficient start system.

The system called Reimer5

The following system belongs to the POSSO test suite (available by anonymous ftp at gauss.dm.unipi.it):

$$F(x,y,z,t,u) = \begin{cases} 2x^2 - 2y^2 + 2z^2 - 2t^2 + 2u^2 - 1 = 0 \\ 2x^3 - 2y^3 + 2z^3 - 2t^3 + 2u^3 - 1 = 0 \\ 2x^4 - 2y^4 + 2z^4 - 2t^4 + 2u^4 - 1 = 0 \\ 2x^5 - 2y^5 + 2z^5 - 2t^5 + 2u^5 - 1 = 0 \\ 2x^6 - 2y^6 + 2z^6 - 2t^6 + 2u^6 - 1 = 0 \end{cases} \tag{7.56}$$

The mixed volume equals the total degree $D = 6! = 720$. There are six defective edges, determined by the inner normals $(-1,-1,0,0,0)$, $(-1,0,0,-1,0)$, $(0,-1,-1,0,0)$, $(0,-1,0,0,-1)$, $(0,0,-1,-1,0)$ and $(0,0,0,-1,-1)$, with $x = y$, $x = t$, $y = z$, $y = u$, $z = t$ and $t = u$ as solutions at infinity. Due to these defective edges, there are only 144 finite solutions, instead of 720. The symmetry group is generated by

$$(x,y,z,t,u) \mapsto (z,y,x,t,u), \quad (x,y,z,t,u) \mapsto (u,y,z,t,x) \tag{7.57}$$

$$\text{and} \quad (x,y,z,t,u) \mapsto (x,t,z,y,u). \tag{7.58}$$

From the twelve generating solutions, two of them are real, the other ten are complex conjugate.

There can only be one mixed cell in any subdivision for this system. Either the volume of a mixed cell equals zero, or it equals 720. Consider for instance the binomial system defined by taking from each equation the constant term and another monomial with a different variable. This yields the start system based on the total degree. All other choices of two monomials from each equation will yield a system whose mixed volume either equals zero or the total degree.

The first lifting consists of giving all monomials, except the constant term, the same lifting. Here we denote the continuation parameter by s, see (7.59) for the corresponding polyhedral homotopy. To avoid singularities, we randomize the constant terms by choosing random complex numbers $q_i \in \mathbb{C}_0$. This randomization does not alter the number of regular solutions, because the deficiency remains unchanged.

$$\widehat{F}^{(1)}(x,y,z,t,u,s) = \begin{cases} (2x^2 - 2y^2 + 2z^2 - 2t^2 + 2u^2)s^7 + q_1 s^1 = 0 \\ (2x^3 - 2y^3 + 2z^3 - 2t^3 + 2u^3)s^7 + q_2 s^1 = 0 \\ (2x^4 - 2y^4 + 2z^4 - 2t^4 + 2u^4)s^7 + q_3 s^1 = 0 \\ (2x^5 - 2y^5 + 2z^5 - 2t^5 + 2u^5)s^7 + q_4 s^1 = 0 \\ (2x^6 - 2y^6 + 2z^6 - 2t^6 + 2u^6)s^7 + q_5 s^1 = 0 \end{cases} \tag{7.59}$$

The inner normal to the mixed cell in the induced subdivision equals $[-3, -3, -3, -3, -3, 1]^T$, the corresponding subsystem $F_C^{(1)}$ has to be refined. The polyhedral homotopy $\widehat{F}^{(2)}$ is displayed in (7.60), hereby we randomize the coefficients of the monomials that contain u, by choosing random complex coefficients $q_{ui} \in \mathbb{C}_0$. Note that this randomized system has two defective edges less and hence, it will have more regular solutions than $F^{(1)}$, 288 to be precise.

$$\widehat{F}^{(2)}(x,y,z,t,u,s) = \begin{cases} (2x^2-2y^2+2z^2-2t^2)s^7 + q_{u1}u^2s^1 + q_1s^1 = 0 \\ (2x^3-2y^3+2z^3-2t^3)s^7 + q_{u2}u^3s^1 = 0 \\ (2x^4-2y^4+2z^4-2t^4)s^7 + q_{u3}u^4s^1 = 0 \\ (2x^5-2y^5+2z^5-2t^5)s^7 + q_{u4}u^5s^1 = 0 \\ (2x^6-2y^6+2z^6-2t^6)s^7 + q_{u5}u^6s^1 = 0 \end{cases} \tag{7.60}$$

The inner normal to the mixed cell in the induced subdivision equals $[-2, -2, -2, -2, 0, 1]^T$ and the corresponding cell is the support of the system $F^{(3)}$, displayed in (7.61). To solve $F^{(3)}$, we have to refine the cell further, by lifting once more. Hereby, we randomize the coefficients of the monomials that contain t. Note that the resulting randomized system has only two defective edges, instead of four, which corresponds to an increase in the number of regular solutions, more precisely, $F^{(3)}$ has 480 regular solutions.

$$\widehat{F}^{(3)}(x,y,z,t,u,s) = \begin{cases} q_{u1}u^2s^0 + q_1s^0 = 0 \\ (2x^3-2y^3+2z^3)s^7 + q_{t2}t^3s^1 + q_{u2}u^3s^1 = 0 \\ (2x^4-2y^4+2z^4)s^7 + q_{t3}t^4s^1 = 0 \\ (2x^5-2y^5+2z^5)s^7 + q_{t4}t^5s^1 = 0 \\ (2x^6-2y^6+2z^6)s^7 + q_{t5}t^6s^1 = 0 \end{cases} \tag{7.61}$$

The inner normal to the mixed cell in the induced subdivision equals $[-3, -3, -3, 0, 0, 2]^T$. In 7.62, the polyhedral homotopy constructed when refining the corresponding mixed cell is displayed. In this step we randomize the monomials that contain z by choosing random complex number $q_{zi} \in \mathbb{C}_0$. The randomized system has only one defective edge and 600 finite solutions.

$$\widehat{F}^{(4)}(x,y,z,t,u,s) = \begin{cases} q_{u1}u^2s^0 + q_1s^0 = 0 \\ q_{t2}t^3s^0 + q_{u2}u^3s^0 = 0 \\ (2x^4-2y^4)s^7 + q_{z3}z^4s^1 + q_{t3}t^4s^1 = 0 \\ (2x^5-2y^5)s^7 + q_{z4}z^5s^1 = 0 \\ (2x^6-2y^6)s^7 + q_{z5}z^6s^1 = 0 \end{cases} \tag{7.62}$$

The inner normal to the mixed cell $\mathcal{C}$ in the induced subdivision equals $[-6, -6, 0, 0, 0, 5]^T$. The restriction of $F^{(4)}$ w.r.t. this cell is displayed in (7.63).

$$F^{(4)}_{\mathcal{C}}(x,y,z,t,u) = \begin{cases} q_{u1}u^2 + q_1 = 0 \\ q_{t2}t^3 + q_{u2}u^3 = 0 \\ q_{z3}z^4 + q_{t3}t^4 = 0 \\ 2x^5 - 2y^5 + q_{z4}z^5 = 0 \\ 2x^6 - 2y^6 = 0 \end{cases} \tag{7.63}$$

Note that $F^{(4)}_{\mathcal{C}}$ contains four binomial equations, which can be used for eliminating four unknowns. The values for the fifth unknown can then be obtained by solving the remaining equation with three monomials. This yields 600 finite solutions.

Table 7.5 summarizes the continuation process. There are four nested liftings, with four corresponding randomizations. Continuing from randomized to more deficient systems, the number of finite solutions decreases.

homotopy	#finite solutions at the start and at the end of the paths								
$\widehat{F}^{(4)}$	600	$\rightarrow$	600						
			$\downarrow$						
$\widehat{F}^{(3)}$			480 (240)	$\rightarrow$	480 (240)				
					$\downarrow$				
$\widehat{F}^{(2)}$					288 (144)	$\rightarrow$	288 (144)		
							$\downarrow$		
$\widehat{F}^{(1)}$							144 (12)	$\rightarrow$	144 (12)
									$\downarrow$
									144 (12)

Table 7.5: Schematic view of recursive polyhedral continuation. The horizontal arrows indicate the polyhedral continuation. The vertical arrows represent the continuation from a randomized system to a specific system. The number of generating solutions are given between brackets.

The first homotopy we use to extend the 600 solutions, to solve $F^{(4)}(x,y,z,u,t) = \mathbf{0}$, is given in (7.64).

$$\widehat{F}_{\mathbf{v}^{(4)}}(x,y,z,t,u,s) = \left\{ \begin{array}{l} q_{u1}u^2 + q_1 = 0 \\ q_{t2}t^3 + q_{u2}u^3 = 0 \\ (2x^4 - 2y^4)s + q_{z3}z^4 + q_{t3}t^4 = 0 \\ 2x^5 - 2y^5 + q_{z4}z^5 = 0 \\ 2x^6 - 2y^6 + q_{z5}z^6 s = 0 \end{array} \right. \tag{7.64}$$

We have observed that the solutions $(x, y = x, 0, 0, 0)$ remain at infinity, when we let s go from 0 to 1. The homotopy used to extend the 288 solutions to solve $F^{(2)}$ is displayed in (7.65).

$$\widehat{F}_{\mathbf{v}^{(2)}}(x,y,z,t,u,s) = \left\{ \begin{array}{l} (2x^2 - 2y^2 + 2z^2 - 2t^2)s + q_{u1}u^2 + q_1 = 0 \\ 2x^3 - 2y^3 + 2z^3 - 2t^3 + q_{u2}u^3 = 0 \\ (2x^4 - 2y^4 + 2z^4 - 2t^4)s + q_{u3}u^4 = 0 \\ (2x^5 - 2y^5 + 2z^5 - 2t^5)s + q_{u4}u^5 = 0 \\ (2x^6 - 2y^6 + 2z^6 - 2t^6)s + q_{u5}u^6 = 0 \end{array} \right. \tag{7.65}$$

Also with this homotopy we have observed that the solutions at infinity remain at infinity.

The computational improvement might seem marginal for this example. However, it is important to validate the results. By this nested lifting method, we can keep track of the solution paths going to infinity, which adds to the reliability of our solver. Moreover, the symmetric structure can be exploited.

A geometric problem

The following system has been communicated to me by Bernd Sturmfels, who received it from Neil Sloane. The application field deals with the construction of nice sets of points in Euclidean space, see [24].

$$F(a,b,c,d,e,f,g) = \left\{ \begin{array}{r} 1 - 2a^5 - 2b^5 - 2c^5 - 2d^5 - 4e^5 - 4f^5 - 4g^5 = 0 \\ 1 - 18a^4b - 18c^4d - 36e^4f - 36g^4e - 36f^4g = 0 \\ 1 - 18b^4a - 18d^4c - 36e^4g - 36g^4f - 36f^4e = 0 \\ 1 - 42a^3b^2 - 42c^3d^2 - 84e^3f^2 - 84g^3e^2 - 84f^3g^2 = 0 \\ 1 - 42b^3a^2 - 42d^3c^2 - 84e^3g^2 - 84g^3f^2 - 84f^3e^2 = 0 \\ 1 - 252e^3fg - 252g^3ef - 252f^3ge = 0 \\ 1 - 420e^2f^2g - 420g^2e^2f - 420f^2g^2e = 0 \end{array} \right. \tag{7.66}$$

The total degree of this system equals $5^7 = 78,125$, while an improved Bézout bound based on a set structure yields 64,855. The BKK bound of this system is 23,750. The symmetry group G is generated by

$$(a\ b\ c\ d\ e\ f\ g) \mapsto (b\ a\ d\ c\ e\ g\ f) \quad \text{and} \quad (a\ b\ c\ d\ e\ f\ g) \mapsto (c\ d\ a\ b\ g\ e\ f) \tag{7.67}$$

which yields $\#G = 12$.

The inner normal to one one of the defective faces is given by $\mathbf{v} = [-1, -1, -1, -1, -1, -1, -1]^T$, with $\partial_{\mathbf{v}} F$ all highest degree monomials. Solutions of $\partial_{\mathbf{v}} F$ are given by

$$\left\{ \begin{array}{l} a = -c, \qquad d = -b, \\ e = -\frac{1}{2}(1 + \sqrt{3}i)f, \\ g = -\frac{1}{2}(1 - \sqrt{3}i)f. \end{array} \right. \tag{7.68}$$

For most computer algebra systems, this is trivial to verify.

The following system represents the deficiency preserving homotopy that is invariant w.r.t. the symmetry group G

$$\widehat{F}^{(0)}(a,b,c,d,e,f,g,t) = \left\{ \begin{array}{lcl} q_1t^0 & + & (-2a^5 - 2b^5 - 2c^5 - 2d^5 - 4e^5 - 4f^5 - 4g^5)t^{10} = 0 \\ q_2t^1 & + & (-18a^4b - 18c^4d - 36e^4f - 36g^4e - 36f^4g)t^{10} = 0 \\ q_2t^1 & + & (-18b^4a - 18d^4c - 36e^4g - 36g^4f - 36f^4e)t^{10} = 0 \\ q_3t^3 & + & (-42a^3b^2 - 42c^3d^2 - 84e^3f^2 - 84g^3e^2 - 84f^3g^2)t^{10} = 0 \\ q_3t^3 & + & (-42b^3a^2 - 42d^3c^2 - 84e^3g^2 - 84g^3f^2 - 84f^3e^2)t^{10} = 0 \\ q_4t^1 & + & (-252e^3fg - 252g^3ef - 252f^3ge)t^{10} = 0 \\ q_4t^2 & + & (-420e^2f^2g - 420g^2e^2f - 420f^2g^2e)t^{10} = 0 \end{array} \right. \tag{7.69}$$

where $q_1, \ldots, q_4$ are randomly chosen complex numbers.

The orbit structure of the subdivision induced by this lifting is given in Table 7.6 and in Table 7.7 the root counts are listed. Due to the symmetric lifting (see [165]), only seven generating cells in the subdivision need to be considered and instead of 23,750 solution paths, 'only' 8,375 paths need to be traced. By respecting the defective face, there are only two cells that yield start solutions, see Table 7.7. So, to reach the 3,000

cell	inner normal $\mathbf{v}$ of $\mathcal{C}_{\mathbf{v}}$	$\#O_{\mathcal{C}_{\mathbf{v}}}$
1	$[-14,-9,-14,-9,-14,-9,6,5]^T$	6
2	$[-23,-13,-8,-23,-23,-13,-8,10]^T$	12
3	$[-2,-2,-2,-2,-2,-2,-2,1]^T$	1
4	$[-10,-5,-9,-9,-9,-9,-9,5]^T$	4
5	$[-4,-3,-3,-4,-4,-3,-3,2]^T$	6
6	$[-30,-25,-30,-25,-27,-27,-26,15]^T$	2
7	$[-19,-19,-19,-19,-20,-15,-15,10]^T$	3
		34

Table 7.6: The orbit structure with generating inner normals $\mathbf{v}$.

cell	$\mathrm{vol}(\mathcal{C}_{\mathbf{v}})$	$N(F_{\mathcal{C}_{\mathbf{v}}})$	$\#O_{\mathcal{C}_{\mathbf{v}}} \times \mathrm{vol}_n(\mathcal{C}_{\mathbf{v}})$	$\#O_{\mathcal{C}_{\mathbf{v}}} \times N(F_{\mathcal{C}_{\mathbf{v}}})$
1	625	0	3,750	0
2	125	0	3,000	0
3	2,500	0	2,500	0
4	1,375	375	5,500	1,500
5	250	250	3,000	1,500
6	2,625	0	5,250	0
7	750	0	2,250	0
	8,375	625	23,750	3,000

Table 7.7: The volumes and numbers of isolated solutions.

isolated solutions of F, only 250 generating paths need to be computed. There are two real solutions, generating a total number of 24. One generating solution has all its components positive.

The approach is recursive. Table 7.7 shows that many more restricted systems are deficient. This is due to the directions $[-1,\ -1,\ -1,\ -1,\ 0,\ 0,\ 0]^T$ and $[0,\ 0,\ 0,\ 0,\ -1,\ -1,\ -1]^T$ that yield faces of $\partial_{\mathbf{v}} P$, respectively w.r.t. the unknowns in $\{a,b,c,d\}$ and in $\{e,f,g\}$.

The restricted system that corresponds to the fourth cell will be elaborated. That polynomial system is a lot sparser than the original problem. By the first binomial equation, it can be reduced to a six-dimensional polynomial system. In (7.70), the deficiency preserving homotopy w.r.t. the normal $[0,\ 0,\ 0,\ 0,\ -1,\ -1,\ -1]^T$ is displayed.

$$\widehat{F}^{(4)}(a,b,c,d,e,f,g,t) = \begin{cases} q_1t^0 - 2a^5t^2 = 0 \\ q_2t^1 - 18a^4bt^3 - 18c^4dt^3 + (-36e^4f - 36g^4e - 36f^4g)t^7 = 0 \\ q_2t^1 + (-36e^4g - 36g^4f - 36f^4e)t^7 = 0 \\ -42c^3d^2t^2 + (-84e^3f^2 - 84g^3e^2 - 84f^3g^2)t^7 = 0 \\ -42d^3c^2t^2 + (-84e^3g^2 - 84g^3f^2 - 84f^3e^2)t^7 = 0 \\ q_4t^0 + (-252e^3fg - 252g^3ef - 252f^3ge)t^7 = 0 \\ (-420e^2f^2g - 420g^2e^2f - 420f^2g^2e)t^7 = 0 \end{cases} \tag{7.70}$$

The lifting respects the symmetry on the monomials with unknowns in $\{e, f, g\}$ and induces a symmetric subdivision. Computing the subdivision is a matter of seconds. This subdivision contains four cells, one defective cell with volume 1000, and three cells that are symmetric w.r.t. each other, yielding together 375 finite solutions. The corresponding restricted system to that one defective cell is

$$F^{(4a)}(a,b,c,d,e,f,g) = \begin{cases} q_1 - 2a^5 = 0 \\ -18a^4b - 18c^4d - 36e^4f - 36eg^4 - 36f^4g = 0 \\ -18cd^4 - 36e^4g - 36ef^4 - 36fg^4 = 0 \\ -42c^3d^2 - 84e^3f^2 - 84e^2g^3 - 84f^3g^2 = 0 \\ -84e^3g^2 - 84e^2f^3 - 84f^2g^3 = 0 \\ q_4 - 252e^3fg - 252ef^3g - 252efg^3 = 0 \\ -420e^2f^2g - 420e^2g^2f - 420f^2g^2e = 0 \end{cases} . \tag{7.71}$$

To verify whether $F^{(4a)}$ is inconsistent, it suffices to consider only the three last equations, which only contain the unknowns e, f and g. Numerically, we have to follow 20 paths, because the BKK bound equals of that three-dimensional system equals 20. This is very cheaply, compared to the computational work need to track 1000 paths in 7-dimensional space. So, to solve $F^{(4)}$, using the homotopy $\widehat{F}^{(4)}$, only 125 generating paths starting at the solutions of $F^{(4b)}$ need to be traced.

7.4.2 Performance

Table 7.8 summarizes all characteristical figures for the applications.

We see that in a lot of cases, the BKK bound improves the Bézout bound considerably. Moreover, when the symmetry can be exploited, fewer solution paths have to be traced to compute all isolated solutions. Finally, for some applications, where the mixed volume does not yield an exact root count and overestimates the number of isolated solutions, we were able to exploit the deficiency.

system	description	n	D	B	$V_n(\mathcal{P})$	NP	$N(F)$
(7.40)	system of Speer	4	625	246	96	10	50
(7.42)	planar four-bar	4	256	96	80	3	36
(7.45)	neurofysiology	6	1,024	216	20	1	8
(7.49)	cyclic 5-roots	5	120	108	70	6	70
(7.54)	reduced cyclic	4	24	20	14	7	14
		5	120	84	26	15	26
		6	720	504	132	66	132
		7	5,040	3,024	320	170	144
(7.56)	system Reimer5	7	720	720	720	600	12
(7.66)	geometric problem	7	78,125	64,855	23,750	250	3,000

Table 7.8: Characteristics for the applications. For each system, the dimension n, total degree D, the best Bézout bound B and the mixed volume $V_n(\mathcal{P})$ is listed. The last two columns contain respectively the number of paths to be followed for solving the start system $F^{(0)}(\mathbf{x}) = \mathbf{0}$ and the number of computed solutions $N(F)$ of the system $F(\mathbf{x}) = \mathbf{0}$.

7.5 Conclusions and future research

In this chapter we have shown how the lifting method can be extended to construct symmetric polyhedral homotopies that enable to compute the generating solutions more efficiently. We have investigated how degeneracy can arise and proposed ways to deal with these problems. The applications illustrate the usefulness of the presented approach.

We have not considered the efficient construction of symmetric subdivisions. For large applications, it would be worthwhile to investigate whether pruning methods can be developed to compute directly the generating cells in the symmetric subdivision. In case the system has fewer solutions than the BKK bound, it seems natural to exploit the deficiency by means of a symmetric subdivision.

Chapter 8

Software

In this chapter, we will describe the software environment that has been developed while investigating the homotopy continuation methods. Previous reports on our software were given in [162] and [159].

We were motivated to write our own software to gain insight in the practical details of the computations. We also wanted to investigate the feasibility of the methods, to show that the methods are effective for solving practical applications. Moreover, the defects of an existing implementation usually point at new research directions. Last and not least, we hope to present a useful collection of programs that perform well on a broad class of applications.

As related software dedicated to solving polynomial systems by homotopy continuation, we briefly mention four different packages that are publicly available. HOMPACK [118, 181] and CONSOL [104] are written in FORTRAN77. HOMPACK, written by Watson, Billups and Morgan, is a general package for homotopy continuation with a polynomial driver. It has been parallelized to various architectures, see [4, 68] and has been recently extended with some facilities for the end game, see [144]. The code for CONSOL, written by Morgan, is contained in [104, Appendix 6]. Morgan, Sommese and Wampler [114, 115, 116] have recently described techniques to handle end point singularities. The programs pss [94] and Pelican [70] are written in C. Malajovitch created pss to apply homotopy continuation with verification by α-theory [95]. The program contains facilities for parallel continuation. Pelican, authored by Huber, provides the polyhedral methods in [72], with the dynamic lifting algorithm as well.

Our package is designed as an integrated workbench, offering various tools. The aims are modularity (wide variety of homotopy methods), effectiveness (on a broad class of applications), portability (on more than one hardware platform) and user friendliness (menu-driven approach, no compilation necessary). We have written the code in Ada and compiled it on three different hardware platforms, for which executables are available on request. The two main features are the wide variety of homotopy methods and the powerful facilities for mixed-volume computation.

The aim of this chapter is to describe the structure of the package, to indicate its main features and to comment on our computational experiences.

8.1 The four stages of the solver

The input is a system $F(\mathbf{x}) = \mathbf{0}$ of n multivariate polynomials $F = (f_1, f_2, \ldots, f_n)$ in n unknowns, $f_k \in \mathbb{C}[x_1, x_2, \ldots, x_n]$, $k = 1, 2, \ldots, n$. The output of our solver consists of root counts and approximations to all the isolated roots of a polynomial system.

In Table 8.1 are the specifications of the main program PHC. We only mention the major components of the output.

Input:	F	*a polynomial system*
Output:	rc	*a root count*
	$F^{(0)}$	*a start system*
	$X_{F^{(0)}}$	*all isolated solutions of* $F^{(0)}$
	X_F	*all isolated solutions of* F

Table 8.1: Specifications of PHC.

In Figure 8.1, the four stages of the solver are displayed. The aim of the pre-processing stage is to bring the system in a form more suitable to homotopy continuation. We refer to [104, Chapter 5] for a description of scaling. Nonlinear reduction is explained in Chapter 2. In the second stage, either a product homotopy (based on Bézout's theorem), or a coefficient homotopy (based on Bernshteĭn's theorem) can be constructed. The tuning of continuation parameters and path following by means of predictor-corrector method is performed in the third stage. The post-processing stage consists of the validation of the computed results, the computation of local condition numbers and eventually the re-run of some difficult solution paths.

$\boxed{\text{1. Pre-Processing}}$		$\boxed{\text{4. Post-Processing}}$
$\Downarrow$ ◇ Scaling of the coefficients ◇ Reduction		$\Uparrow$ ◇ Refining of the roots ◇ Re-run of paths
$\boxed{\text{2. Homotopy Construction}}$	$\Longrightarrow$	$\boxed{\text{3. Homotopy Continuation}}$
◇ Bézout $\Rightarrow$ product homotopy ◇ Bernshteĭn $\Rightarrow$ coefficient homotopy		◇ Tuning of continuation parameters ◇ Choice of Predictor-Corrector

Figure 8.1: The four stages in the solver.

An important difference in Figure 8.1 compared with the CONSOL-logic (see [104, Chapter 4]) is the second stage, which is not only not well elaborated in CONSOL, but is also closely related to the continuation stage. The design of HOMPACK has the same drawback, the package gives the user no support when another start system, other than the one based on the total degree, needs to be used.

For small polynomial systems, the four stages should be very well integrated and customized. For large applications, each stage has to be clearly separated in order to have several stepping stones. Therefore, we have added options to the program, so that it provides several tools which address only one particular stage of the solver.

8.2 Tools and modes

We have to respect a strict processing order when applying homotopy continuation. Therefore, an interactive program with (hopefully) self-explanatory menus guides the user in the solving process. This menu-driven approach also enables some kind of batch processing. Enumerating all isolated solutions quickly leads to a computationally lengthy job (ranging from one hour, a night, several days...). So after we have chosen the algorithms from the menus, the program can run quietly in the background, writing the results to a file.

There are three modes to run the program. The first one is the full mode, where we run through all the different stages at once. If we only want to deal with only one specific stage of the solving procedure, we have to invoke the program via the tool mode. The third and last mode is the black-box or batch processing mode. We will now give a more elaborate description of the last two modes.

Often only a part of the solver is needed. For example, first we want to focus on estimating the number of roots to get an idea about the complexity of the problem before attempting to solve it. So, in this first stage, we want to try several root counting methods before starting the continuation. Once an efficient homotopy is constructed, there is no need to go through all the menus of the root counting methods and the corresponding homotopy constructors. Therefore, the different components in the stages of the solver can be invoked separately. The hierarchy of the interactive drivers is displayed in Figure 8.2.

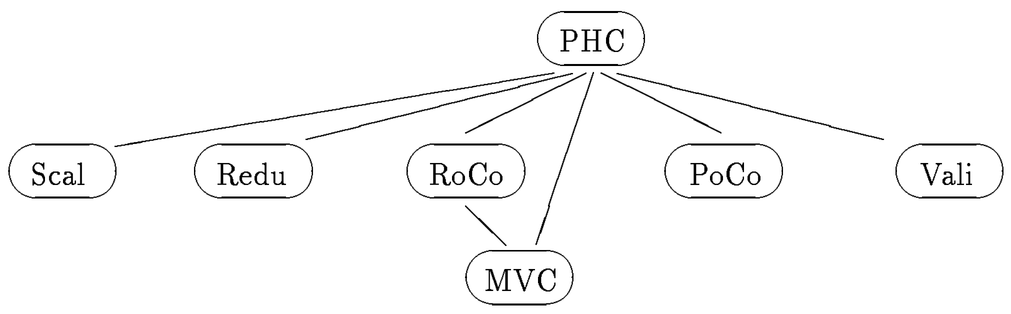


Figure 8.2: Schematic overview of the tools offered by the package PHC.

The program PHC is the main interactive driver and gives access to several tools. The two most important ones are RoCo (Root Counting) and PoCo (Polynomial Continuation). They are directly activated when calling PHC with the options '-r', for doing root counting only, and '-p', for polynomial continuation when we have already a homotopy at our disposal. The program MVC is a subdriver of RoCo and can be invoked by calling PHC with the option '-m'. Table 8.2 summarizes the options.

stage	acronym	description of the tool	option
1	`scal`	scaling	`phc -s`
	`redu`	reduction	`phc -d`
2	`roco`	root counting	`phc -r`
	`mvc`	mixed-volume computation	`phc -m`
3	`poco`	polynomial continuation	`phc -p`
4	`vali`	validation	`phc -v`

Table 8.2: Overview of tools, acronyms and options of PHC.

The third mode to invoke the program is the black-box or batch processing mode, by calling phc with the option '-b'. This mode avoids the usage of menus and applies the methods either with the values of the parameters provided in the input file, or otherwise with the default settings of the parameters. This is useful either for small examples (to get a quick and dirty run) or when we want to redo the computations, with small modifications of the specific settings of the parameters we have made. These settings of the parameters have to be provided as part of the input file, following the same way they have been written in the log file.

The tools produce several intermediate results, such as a mixed subdivision, a start system and solution sets. These are permanent objects and are stored as files, so that they survive after the program has ended.

8.3 The design: towards a workbench

The software forms the basis to perform computational explorations with the intention to gain more insight into the complexity of the problem. Therefore, instead of aiming at a black-box solver, which works well on average polynomial systems (which are not practical applications of course), a more open design, i.e., towards the development of a workbench, turns out to be more interesting. The best solution methods always consist of an appropriate combination of good algorithms.

Figure 8.3 pictures the four large components of the software. The mathematical library contains the prerequisites needed, not typically related to polynomial homotopy continuation. Root counting methods based on Bézout's and Bernshteĭn's theorem have been implemented, as well as the corresponding homotopy constructors. This is the second component. The third component consists of the predictor-corrector methods and data abstractions needed for polynomial path following. At last, the interface represents all communication with the user, who can be either the end-user or the application programmer.

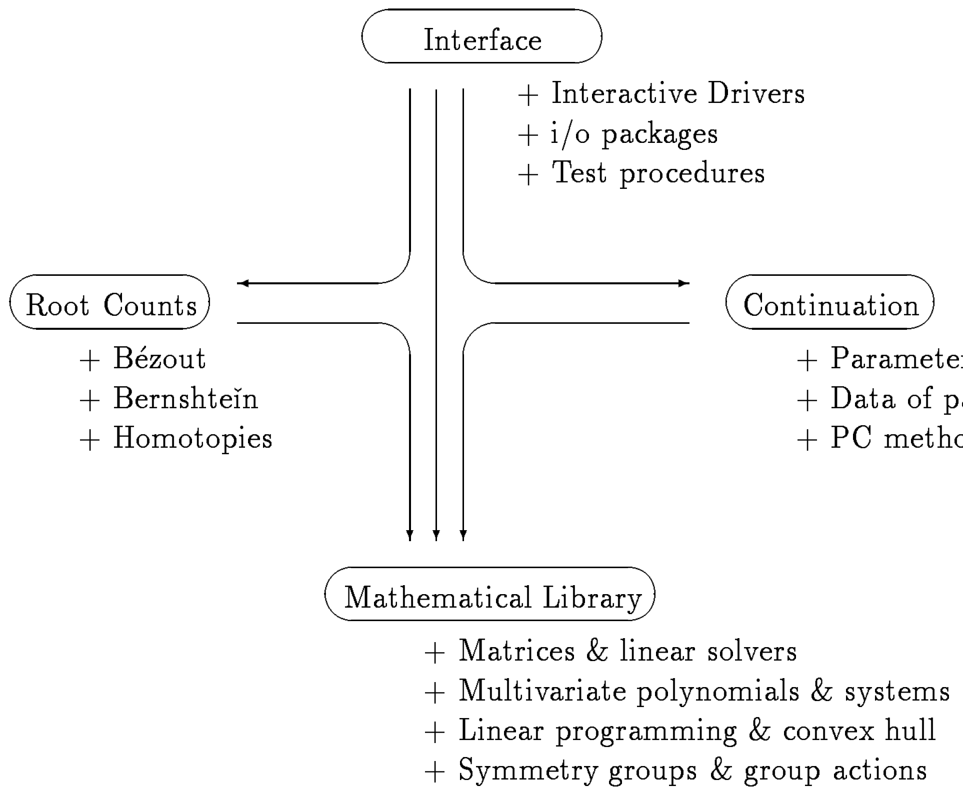


Figure 8.3: The four components with their dependencies

It should be stressed that the design pictured in Figure 8.3 is mainly a logical view of the entire software package. Especially the i/o operations of specific data types, although contained in separate packages (for each data type a separate package), are situated close to the definition of the package exporting that data type.

8.4 Object-oriented programming with Ada

In this section we try to motivate the benefits of object-oriented programming and to justify our choice for Ada as programming language.

Object-oriented development methods can be considered as structured programming techniques to help us out of the software crisis, see [18]. The essence of the software crisis is that building efficient, reliable, maintainable and understandable software systems is a very labor intensive and hence expensive job. But is there really a software crisis in the world of scientific computing? There are many examples of general-purpose and user-friendly mathematical software for performing numerical and symbolic computations interactively. Furthermore, various PACKages (such as HOMPACK) are commonly available, dedicated to one specific problem area, implementing one particular class of algorithms.

The following paragraph is a literal quotation from the announcement of OONSCI'96 (the Fourth Annual Object-Oriented Numerics Conference): "Scientific application software continues to grow in complexity due to increasing functionality, evolution of codes, the use of parallelism, and other factors. As a result, more and more time is spent writing and maintaining computer code, and less time is spent on the application itself. Object technology offers a way to break this cycle. Structuring applications as assemblages of software objects, rather than as monolithic programs, may pave the way for more efficient and complex applications by making software reusable, distributable, and modular."

During the last years, the algorithms needed in homotopy continuation have grown in complexity, using concepts from different and not directly related fields of computational mathematics. Polyhedral homotopy methods combine path following methods, from numerical analysis, and algorithms to construct convex hulls, from discrete and computational geometry. CONSOL and HOMPACK mainly focus on the continuation methods, rather than on the algebraic nature of the problem. The homotopy used in those packages is not well enough separated from the path following algorithms. To implement modern homotopy continuation methods, we need to apply modular programming methods.

Of course, structured programming can be done in languages such as FORTRAN77 and even in C, but why not use a language which supports and even enforces modular programming? The concept of a 'package' is a fundamental construction in Ada, see [7] and [18]. Ada has been specially designed for implementing large and complex software systems, that require high demands on safety and reliability. The quality of any program is largely characterized by its design. But usually, the structure of a program grows incrementally, with modifications depending on the change of demands made by the user. In the design of scientific software, this means that programs have to be able to incorporate the growing understanding of the mathematical structure of the problem. We experienced that the package mechanism of Ada is very helpful in guiding the transition between the different stages of designing a reliable software system.

Recently, the language has gone through a revision process and can be regarded as the first standardized full object-oriented programming language, already implemented by the gnu-ada compiler, see [35]. The use of Ada is also becoming more and more widespread.

8.5 The major tools of the four stages

In this section we take a closer look at each tool. For each tool we give the specifications, list the main features and comment on the parameters that need to be tuned. As the package is still under continuing development, we also try to mention some of the things that should be added.

8.5.1 Scal: Scaling of systems and solutions

We have implemented two different types of scaling, following the description in [104, Chapter 5]. First there is *equation scaling*, where each polynomial is divided by a constant. This is combined with the second type of scaling, so-called *variable scaling*, where we scale each variable by a constant. Table 8.3 lists the specifications of this tool. We can use it either to scale a polynomial system or to de-scale the computed solutions back into the original coordinates.

Input:	F	*a polynomial system*
	or $X_{F'}$	*the solutions of a scaled system*
Output:	F'	*a scaled system*
	or X_F	*the solutions of the original system*

Table 8.3: Specifications of Scal.

As basis of our implementation, we followed closely [104, Chapter 5] and restricted ourselves to scaling, no centering is provided. The package has also no facilities to deal with multi-precision arithmetic, so it could very well be that a lot of precision is lost by transforming from one coordinate system into the other.

8.5.2 Redu: Linear and nonlinear reduction

As we explained in Chapter 2, there are two systematic ways implemented for reducing the system. The first one is linear reduction, that consists in the application of Gaussian elimination on the coefficient matrix, see [104, Chapter 7]. Furthermore, we have implemented nonlinear reduction as the second way to reduce the total degree of a polynomial system. Hereby we compute S-polynomials to eliminate the leading term. Table 8.4 lists the specifications for this tool. When the user chooses for nonlinear reduction, several bounds can be given to limit the complexity of the reduction algorithm, see Chapter 2 for a detailed description of the meaning of those parameters.

Recently we have extended the reduction tool with two facilities to deal with overdetermined systems. The first one is described by Sommese and Wampler in [143]. Suppose we have a system of m equations in n unknowns, with $m > n$, then we can generate a

Input:	F	*a polynomial system*
	M_{eq}	*maximum number of replacements with equal total degree*
	M_S	*bound on the number of computed S-polynomials*
	M_R	*bound on the number of computed R-polynomials*
Output:	F'	*a reduced system,* $D_{F'} \leq D_F$
	#eq	*number of replacements with equal total degree*
	#S	*number of computed S-polynomials*
	#R	*number of computed R-polynomials*

Table 8.4: Specifications of Redu.

matrix with randomly chosen complex numbers c_{ij} to rewrite the system as follows

$$F(\mathbf{x}) = \begin{cases} f_1 = 0 \\ f_2 = 0 \\ \quad\vdots \\ f_m = 0 \end{cases} \rightarrow \begin{cases} f_1 + c_{1n+1} f_{n+1} + c_{1n+2} f_{n+2} + \cdots + c_{1m} f_m = 0 \\ f_2 + c_{2n+1} f_{n+1} + c_{2n+2} f_{n+2} + \cdots + c_{2m} f_m = 0 \\ \qquad\qquad\vdots \\ f_n + c_{nn+1} f_{n+1} + c_{nn+2} f_{n+2} + \cdots + c_{nm} f_m = 0 \quad . \end{cases} \quad . \tag{8.1}$$

In [143], Sommese and Wampler prove that every isolated solution of the original system is an isolated solution of the constructed square system, with the corresponding multiplicity.

This method is recommended when the solution set of any choice of n equations from the system of m equations is unbounded. But often, it might happen that already the first choice of n equations has only isolated solutions and that the other equations are used to weed out this solution set. Then we would like to reduce these first n equations by using the remaining $m - n$ equations. This mechanism is provided in the second facility, which uses R-polynomials to eliminate the leading monomials. Especially in view of handling sparse polynomial systems, this second type of reduction seems to be interesting.

8.5.3 RoCo: root counting and homotopy construction

In Table 8.5, the specifications of the tool RoCo are listed. The output contains a root count for F, either based on Bézout's or Bernshtein's theorem. Corresponding to this root count, a start system $F^{(0)}(\mathbf{x}) = \mathbf{0}$ can be constructed, that allows to create a homotopy for computing all isolated solutions of $F(\mathbf{x}) = \mathbf{0}$.

Input:	F	*a polynomial system*
Output:	rc	*a root count*
	$F^{(0)}$	*a start system*
	$X_{F^{(0)}}$	*the solutions of* $F^{(0)}$

Table 8.5: Specifications of RoCo. The last two output arguments are optional.

The main menu of RoCo is listed as Table 8.6. The distinction between m-homogeneous and multi-homogeneous Bézout numbers depends on whether we model the structure by one or by many partitions. With multi-homogenization, for every polynomial there can be a different partition of the sets of unknowns. The menu is executed in a loop, to enable the computation of various root counts before proceeding to the continuation stage. The last part of this menu appears also when calling the tool MVC.

```
Choose one of the following :
0.  exit - current start system is based on total degree
---- product homotopies --------------------------------------
1.  m-homogeneous Bezout number              (one partition)
2.  multi-homogeneous Bezout number        (many partitions)
3.  bound based on set structure analysis
4.  exploit the symmetry by symmetric set structure analysis
---- polyhedral homotopies -----------------------------------
5.  mixture between Bezout and BKK bound (implicit lifting)
6.  mixed-volume computation                (static lifting)
7.  incremental mixed-volume computation  (dynamic lifting)
8.  symmetric mixed-volume computation  (symmetric lifting)
--------------------------------------------------------------
Make your choice :
```

Table 8.6: The main menu of RoCo.

8.5.4 MVC: four lifting strategies

First we give the program specifications of MVC, see Table 8.7.

Input:	F	*a polynomial system*
Output:	$V_n(\mathcal{P})$	*the mixed volume*
	S_ω	*a regular subdivision*
	$F^{(0)}$	*a randomized system*
	$X_{F^{(0)}}$	*the solutions of* $F^{(0)}$

Table 8.7: Specifications of MVC. The last two output arguments are optional.

In Figure 8.4, the different lifting strategies are pictured.
The first decision level is concerned with the combination of product and polyhedral homotopies. In many applications, the polytopes can be decomposed as the sum of smaller ones. This leads to a product of polynomials whose Newton polytopes are the smaller polytopes in the decomposition. If for some polynomials in the system, an exact decomposition of the Newton polytopes can be obtained, then the computation of the mixed volume can be computed significantly faster.

Implicit lifting is based on the recursive formula to compute the mixed volume, used by Bernshteĭn in his proof. Huber and Sturmfels developed a lifting method for computing

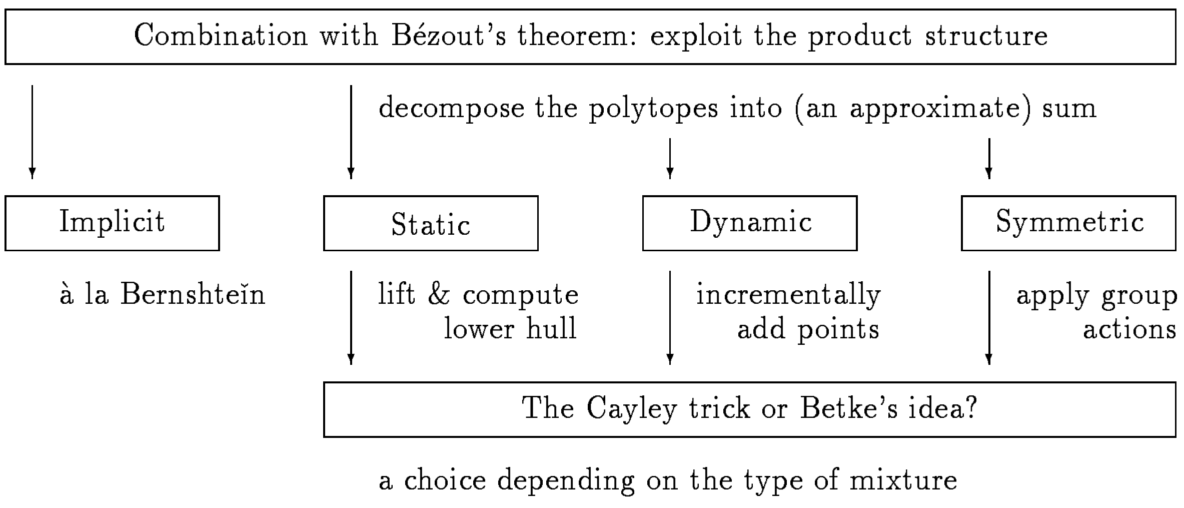


Figure 8.4: The four lifting strategies and three decision levels

mixed subdivisions and mixed volumes. Here this is called static lifting to distinguish it from dynamic lifting. Static lifting means that first all points are lifted, followed by the computation of the lower hull. Six different lifting functions are provided, see Table 8.8 for the menu offered by static lifting.

```
Choose a kind of lifting function :
 Linear     :  1.  Random linear vectors will be chosen.
            :  2.  You can choose the vectors yourself.
 Polynomial :  3.  The system to be solved will be used.
            :  4.  You can choose the polynomials yourself.
 General    :  5.  For each point a random lifting value.
            :  6.  You can choose all lifting values yourself.
Type the number of your choice :
```

Table 8.8: The different lifting functions for static lifting.

There is linear lifting, lifting by means of a polynomial function or by any general assignment of lifting values to the points. To keep the lifting values low, the last two lifting facilities are very useful, because they allow to bound the lifting values. Note that recursion becomes inevitable. Therefore, subdivisions are also stored by preserving this recursive structure.

The dynamic lifting algorithm allows the incremental update of the mixed subdivision. In this way it combines both the advantages of implicit and dynamic lifting. In addition, recursion has been built in both the static and dynamic lifting algorithm. This implies that, with an initial lifting function the induced subdivision does not need to be mixed, because it can be refined recursively. Last but not least, symmetric subdivisions can be constructed by means of the symmetric lifting algorithm, by either the static or dynamic version. Hereby only the generating solutions of the symmetric system have to be computed.

The third decision level offers the choice between the Cayley trick and Betke's idea. The Cayley trick reduces the mixed-volume computation to the computation of the volume of a larger, $(n+r-1)$-dimensional polytope, where r equals the number of different polytopes in the tuple $\mathcal{P}$. In practice, it turns out that this trick is efficient when the system is semimixed, i.e., when r is small. The idea of Betke is to make appropriate face-face combinations to construct the cells which contribute to the mixed volume. This idea works efficient when the system is fully mixed, i.e, when $r = n$.

Figure 8.4 has been drawn based on the idea of what should be, rather than what is already available. Not every feature mentioned in Figure 8.4 is fully implemented. The combination of product and polyhedral homotopies, i.e., the combination of the bounds of Bézout and Bernshteĭn, is only supported with implicit lifting. Up to now, the Cayley trick works only when choosing for the dynamic lifting algorithm.

By default, the program checks the connectivity of the mixed cells in the subdivision (see Conjecture 6.2.20), each time a mixed subdivision has been computed. Until now, the verification of this conjecture always confirmed the strong connectivity of the mixed cells.

8.5.5 PoCo: Polynomial Continuation

In Table 8.9, we list the specifications of the tool PoCo. The user can give either an artificial-parameter homotopy, by explicitly giving a start system, or a natural-parameter homotopy, in the form of a polynomial system with one more unknown than the number of equations. In both cases a list of start solutions should be provided.

Input:	$(F, F^{(0)})$	*a target and a start system*
	or H	*a natural-parameter homotopy*
	$X_F^{(0)}$	*start solutions*
Output:	X_F	*the solutions of* F

Table 8.9: Specifications of PoCo.

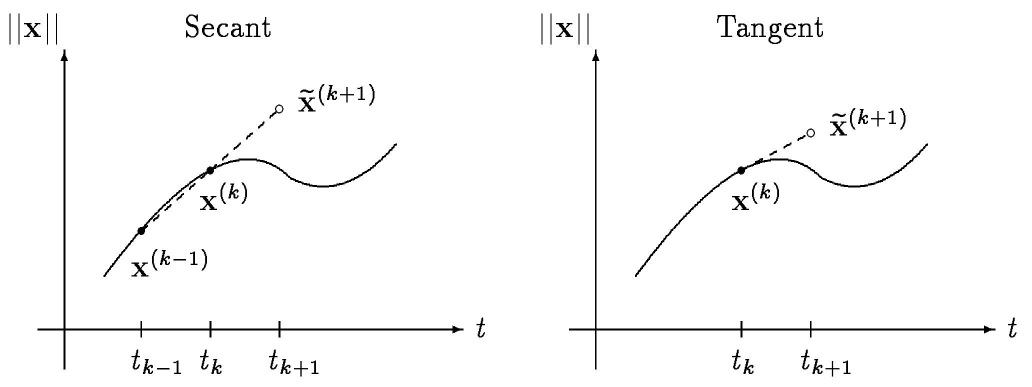
The following of the solution paths is organized in two stages. During the first stage we have $t < 1$. No singularities are likely to occur in this stage. Moreover, we are not interested in the exact solutions for $t < 1$. So we can afford ourselves to track the paths more loosely. The second stage is at the end of each path, when $t \approx 1$. Here we may have to deal with paths converging to singular solutions and with paths running to infinity, as $t \to 1$. In this stage, we have to track the paths more carefully.

The solution paths are not likely to turn back as the continuation parameter t increases. Therefore, instead of pseudo arc-length, only increment-and-fix predictor corrector methods have been implemented. This means that after each increase of t, t remains fixed while correcting the solution $\mathbf{x}$. There are four different predictor methods available. They are listed in Table 8.10, with a short description.

	Name	Description
t	Real	Δt = step length λ
	Complex	$\Delta t \in \mathbb{C}$ when close to singularity
$\mathbf{x}$	Secant	$\Delta \mathbf{x}$ is based on linear extrapolation
	Tangent	uses information of Jacobian matrix

Table 8.10: 2×2 predictors for $t := t + \Delta t$ and $\mathbf{x} := \mathbf{x} + \Delta \mathbf{x}$.

The complex predictor for t is based on an idea of Drexler, see [36], who proposed to take t in complex space when singularities during continuation occurred.
The differences between secant and tangent predictor are illustrated in Figure 8.5.



$\tilde{\mathbf{x}}^{(k+1)} := \mathbf{x}^{(k)} + \lambda(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)})$ $\qquad$ $\tilde{\mathbf{x}}^{(k+1)} := \mathbf{x}^{(k)} + \lambda \frac{d\mathbf{x}(t_k)}{dt}$

Figure 8.5: The secant and tangent predictor with λ as step length.

In Figure 8.5, we see that the tangent predictor is able to follow the solution path more accurately. This justifies the extra work to be done, such as computing derivatives and solving a linear system.

So, there is a tradeoff to make. Just like in the tuning of the continuation parameters. We distinguish four different sets of parameters. First, we have the global parameters for monitoring the path tracker for all paths, such as setting an upper bound on the number of steps along a path. The second set of parameters deals with the control of the step size. The Newton corrector takes the third set of parameters into account. Finally, tolerances are needed to decide whether a solution is singular or lies at infinity and to determine whether two or more solutions from a 'cluster', i.e., when they lie too close to each other to be considered as different solutions.

In Table 8.11, an overview of the default values, recommended for a first run, is presented. The default values are set optimistically, to handle small polynomial systems.

```
****************** CURRENT CONTINUATION PARAMETERS *****************
CONDITION :
0.   modeling the condition of the homotopy            :  0
MONITOR :
1.   number of paths in block                          :  1
2.   kind of predictor ( Secant for x, Secant for t )  :  0
3.   maximum number of steps along a path              :  500
4.   distance from target to start end game            :  1.000E-01
5.   maximum number of re-runs allowed                 :  1
STEP CONTROL (PREDICTOR) :
6.   minimum step size                                 :  1.000E-04
7.   maximum step size                                 :  1.000E-01
8.   reduction factor for step size                    :  7.500E-01
9.   expansion factor for step size                    :  1.250E+00
10.  number of successful steps before expanding       :  1
PATH CLOSENESS (CORRECTOR) :
11.  maximum number of iterations along the path       :  4
12.  maximum number of iterations at end of path       :  4
13.  relative precision for residuals along the path   :  1.000E-04
14.  absolute precision for residuals along the path   :  1.000E-04
15.  relative precision for corrections along the path :  1.000E-04
16.  absolute precision for corrections along the path :  1.000E-04
17.  relative precision for residuals at end of path   :  1.000E-08
18.  absolute precision for residuals at end of path   :  1.000E-08
19.  relative precision for corrections at end of path :  1.000E-08
20.  absolute precision for corrections at end of path :  1.000E-08
SOLUTIONS :
21.  tolerance for inverse condition of Jacobian       :  1.000E-08
22.  tolerance for clustering of solutions             :  1.000E-08
23.  tolerance for solution at infinity                :  1.000E+08
********************************************************************
```

Table 8.11: The tableau of continuation parameters with default values, displayed as one of the menus of PHC.

The central parameter in the tableau is the condition parameter, it models the condition of the homotopy. Increasing this parameter will result in a modification of the other parameters, in the sense that paths will be tracked more accurately. Practical experiences have shown that when we force the path tracker to stay very close to the path, we risk of getting stuck at a singularity.

There are several output levels during continuation, they are listed in Table 8.12.

```
Do you want intermediate output during continuation ?  (y/n) y
Which information during continuation is wanted ?
 0 :  only the solution at the end of the path
 1 :  intermediate solutions
 2 :  information of the predictor:  t and step length
 3 :  information of the corrector:  corrections and residuals
 4 :  intermediate solutions and information of the predictor
 5 :  intermediate solutions and information of the corrector
 6 :  information of predictor and corrector
 7 :  intermediate solutions, information of predictor and corrector
Make your choice :
```

Table 8.12: The menu of PoCo with output options during continuation.

8.5.6 Vali: Validation of the computations

We distinguish between a basic validation and an elaborated end game strategy. As basic validation we have implemented the application of Newton's method to the whole solution list. Hereby we compute estimates for the condition number of the Jacobian matrix and check whether all solutions are different from each other. Table 8.13 lists the specifications for this basic validation.

Input:	F	*a polynomial system*
	X_F	*approximations to the solutions of* F
Output:	X_F	*refined solutions of* F

Table 8.13: Specifications of Vali.

An example of a more elaborated end game strategy is the computation of winding numbers, as explained in [114] by Morgan, Sommese and Wampler. Our validation contains a (primitive) facility of computing winding numbers.

8.6 Computers and compilers

Our software has been developed with VADS (Verdix Ada Development System) on three different machine architectures. We started on a SUN3/280, moved to a DECStation 5200, and later to an IBM RS/6000 workstation. In Table 8.14 we give an overview of the computers where we can run our software, together with the compilers we used. We distinguish five different machines, from three different computer manufacturers.

machine	operating system	compilers	
SUN 3/280	SunOS 4.1.1	VADS 6.0.3d	
DECStation 3100, 5200	Ultrix V4.4	VADS 6.1.0d	
IBM RS/6000	AIX 3.2.5	VADS 6.2.3a	
SPARCstation2	SunOS 4.1.3		gnu-ada 3.03
SPARCserver 1000MP	Solaris 2.3	VADS 6.2.3b	gnu-ada 3.03

Table 8.14: Supported hardware platforms and compilers

Recently we have started working on compiling the software by means of the gnu-ada compiler (see [35]). The latest version (version 3.03) has enabled to realize a full portable version of our program. In the future we expect that more and more ports of the gnu-ada compiler will emerge, so that we can run our software everywhere.

Finally we have to make a note on the evolution of speed. The DECSstation 5200 and the IBM RS/6000 are about seven times faster than the SUN3/280 we started working on. We can run our executable compiled at our IBM RS/6000 on the node of a IBM SP2 computer, which is a collection of IBM Risc/6000 processors. To our experience, one node is about five time faster than the IBM RS/6000 workstation we compile our software at. On the whole, the machines we have at our disposal grew 35 times faster. Fortunately, by using advanced homotopy continuation methods, the speedup achieved is sometimes much larger, so that choosing the right algorithm remains crucial, even on a fast machine.

8.7 Computational experiences

All computations have been performed on an IBM RS/6000 workstation. It should be stressed that the timings are only meant to give a relative idea of the performance of the implemented algorithms. Obviously, what can be solved is mainly a function of the state of the current technology and the timings are only valid for a very limited duration.

First we illustrate how reduction can be used to make the homotopy based on the total degree more performant. Then we apply multi-homogenization to the class of examples we presented in Chapter 3. The practical performance of product homotopies on symmetric polynomial systems is presented thirdly. In the fourth subsection we show the performance of the criterion to omit superfluous monomials. In the fifth subsection we compare various lifting techniques. A report on the solution of various instances of a polynomial system that arose in the design of a planar four-bar mechanism is given in the sixth subsection. The last two subsections deal with the cyclic n-roots problem. In the seventh subsection we give our experiences on the cost of obtaining a low-valued lifting function and finally, we report on solving the reduced cyclic n-roots problem.

8.7.1 The benefits of reduction

In Table 8.15, the practical performance results of the reduction algorithm applied on the examples of Chapter 2 (see Table 2.1) are given. As in Table 2.1, D_F and $D_{F'}$ denote the respective total degrees of the original and the reduced system, while $N(F)$ equals the number of finite solutions. We compare the work to be done for reduction with the time needed to solve the polynomial system.

The applications		Characteristics			Timings (cpu sec.)			
system	description	D_F	$D_{F'}$	$N(F)$	red.	cont.	total	no red.
(2.33)	overconstrained	12	4	3	0.18	0.61	0.79	4.87
(2.34)	quadrature over grid	120	8	5	1.48	4.33	5.81	293.64
(2.35)	optimizing Wood	36	16	9	0.08	11.14	11.22	39.90
(2.36)	lumped-parameter	16	4	4	0.18	0.76	0.94	9.59
(2.37)	electrochemistry	60	40	15	0.08	65.13	65.21	92.08

Table 8.15: Execution times to illustrate the benefits of reduction. The first three columns with timings respectively indicate the time needed for reduction, for continuation and the total time. The last column gives the time needed to solve the system without reduction.

Comparing the last two columns in Table 8.15, we see that it is worthwhile to apply first reduction to the deficient systems. The time needed to reduce the system is negligible in comparison to the time needed to solve the original problem.

8.7.2 Applying multi-homogenization

In Table 8.16, we give the practical performance of multi-homogenization, applied to the examples, also presented in Table 3.3. This time we have made no explicit comparison with the homotopy based on the total degree, but it suffices to consider the ratio $\frac{D}{B}$ to see that by multi-homogenization we can solve the applications at least 3 up to 500 times faster.
From Table 8.16 we may conclude that the time needed by the homotopy method is little compared to the time consumed in the path following stage.

8.7.3 Symmetry and product homotopies

The practical performance of the product homotopies applied to some symmetrical polynomial systems (see also Table 4.13) is displayed in Table 8.17.

In Table 8.17, we see that solving the linear-product start system for the 7-dimensional Lotka-Volterra application takes more time than the path following of the generating

The applications		Characteristics			Timings (cpu sec.)		
system	description	D	B	$N(F)$	per.	hom.	cont.
(3.24)	easy for elimination	1,024	2	2	0.85	0.17	3.65
(3.25)	generalized eigenvalues	243	10	10	0.13	0.22	14.83
(3.26)	conformal analysis	64	16	16	0.00	0.08	6.55
(3.27)	camera motion	64	20	20	0.13	0.18	56.59
(3.28)	model of PUMA robot	128	16	16	0.24	0.65	49.67
(3.29)	six-revolute-joint	256	96	32	0.25	0.99	466.85

Table 8.16: Practical performance of multi-homogenization. The timings include the time needed by the heuristic partitioner (per.), the time of the homotopy method (hom.) and the continuation (cont.).

The applications		Characteristics				Timings (cpu sec.)		
system	description	D	B	$N(F)$	#paths	per.	hom.	cont.
(4.30)	Lotka-Volterra	81	73	73	13	0.01	0.38	12.14
		243	243	233	19	0.04	3.81	60.14
		729	729	717	26	0.15	40.21	138.85
		2,187	2,187	2,173	34	0.58	426.64	232.55
(4.33)	system of Speer	625	271	50	83	0.27	6.76	920.18
(4.35)	Virasoro algebras	256	256	256	76	0.56	3.73	350.40
(4.38)	cyclic 5-roots	120	120	70	12	0.34	17.76	44.16

Table 8.17: Practical performance of the product homotopies. The column with heading 'per.' indicates the timings for the computation of the generalized permanent. The next column gives the time needed for the construction and the solution of the symmetric linear-product start system, whereas the last column displays the timings for the continuation.

paths. This is explained by the fact that the number of start solutions exceeds by far the number of generating solutions.

8.7.4 Omitting superfluous monomials

In this section we present the practical performance results of the criterion proposed in Theorem 6.2.10 to omit monomials whose exponents do not influence the mixed volume. See also Table 6.3.

In Table 8.19 we give the execution times of our program on an IBM RS/6000. The legend for the headings is displayed in Table 8.18.
We see from Table 8.19 that the computational cost of applying the criterion is often negligible, compared to the cost of the pruning method.

We have given the timings for the elimination of the vertex points to give an idea

ver	:	elimination of the nonvertex points
lifa	:	computation of the edges on the lower hull of the lifted polytopes
prune	:	pruning method to compute inner normals to mixed cells
vol	:	calculating the volumes of the mixed cells
infl	:	successive application of the simple criterion until no omission possible

Table 8.18: Legend to the headings of Table 8.19.

Description of the applications	For all points				With the criterion			
	ver	lifa	prune	vol	infl	lifa	prun	vol
system of A.H. Wright	0.07	0.09	2.31	0.04	0.42	0.00	0.00	0.01
system called Reimer5	0.06	0.06	1.11	0.01	0.49	0.03	0.00	0.02
model of PUMA robot	0.06	0.04	3.33	0.01	1.47	0.03	0.13	0.06
heart-dipole problem	0.18	0.29	101.97	0.33	3.41	0.25	75.14	0.51
a geometric problem	0.12	0.14	125.59	0.54	1.87	0.05	13.35	0.16

Table 8.19: Execution times for all stages of the lift-and-prune method, once computed for all points and once after the application of our criterion.

about the relative cost compared to the criterion. When vertex points can be eliminated, then points in the interior can become vertex points. Because of this, we have computed the sets of vertex points only when we did not intend to apply our criterion.

8.7.5 Lifting and polyhedral homotopies

In Table 8.21 and Table 8.22 we list the results of our computational experiences with implicit, static and dynamic lifting methods. The characteristics of the applications are displayed in Table 8.20.

The '–' in Table 8.20, Table 8.21 and Table 8.22 indicates that the memory complexity of the Cayley trick grew beyond the limits of our machine. In general, we see that this trick becomes expensive when the system is fully mixed. Also for the current implementation of the implicit lifting method, some systems were too hard to handle.

The lifting values were always between 0 and 17, for both the static and dynamic lifting. With the recursive mechanism in the static lifting algorithm we can also establish low lifting values, as with the dynamic lifting algorithm.

For the six-revolute-joint problem, we have combined implicit lifting with set-structure analysis, i.e., for the first four polynomials, products of random linear polynomials have been used to eliminate four unknowns. Also for the camera motion problem, we have used a product structure for one of the equations, but only when applying the implicit lifting. For the combustion problem, we solved the scaled problem.

It is interesting to compare the results of Table 8.21 and Table 8.22 with Table 8.16, for those systems we solved both with a multi-homogeneous homotopy and with a poly-

The applications		Characteristics					Cardinalities		
system	description	n	r	D	$V_n(\mathcal{P})$	$N(F)$	d	$\#\mathfrak{S}_\omega$	$\#\lhd$
(3.27)	camera motion	6	3	64	20	20	3	17	415
(3.28)	PUMA robot	8	8	128	16	16	1	1	580
(3.29)	six-revolute-joint	8	8	256	64	32	1	4	–
(5.66)	combustion 3000°	10	10	96	16	16	2	3	1,164
(5.67)	chem. equilibrium	5	5	108	16	16	3	4	247
(5.70)	benchmark i1	10	10	59,049	66	50	3	13	–
(5.71)	integration formula	9	9	36,000	136	16	–	15	–
(6.36)	inverse position	11	11	1,024	288	16	–	12	–
(6.37)	heart-dipole	8	8	576	121	4	–	15	–
(6.38)	Butcher's problem	7	7	4,608	24	5	6	8	2,050
(7.45)	neurofysiology	6	6	1,024	20	8	3	2	70

Table 8.20: Characteristics and cardinalities of the applications. We list the dimension n and the number of different support sets r. The column with heading 'd' gives the depth of the three with useful directions in the implicit lifting method, while $\#\mathfrak{S}_\omega$ denotes the number of mixed cells and $\#\lhd$ the total number of cells in the triangulation constructed by the Cayley trick.

hedral method. The polyhedral homotopy is more advantageous when the mixed volume provides a more accurate root count (like the six-revolute-joint problem) or when the system is sparser (like the model of the PUMA robot). On the other hand, when the mixed volume does not improve the multi-homogeneous Bézout number and when the system is rather dense, as is the case with the camera motion problem, it is better to use a multi-homogeneous homotopy.

From Table 8.22 we see that the polyhedral homotopy is usually better when induced by the dynamic lifting algorithm. Only for the inverse position and the heart-dipole problem, it is very hard for the current implementation of dynamic lifting to compute a subdivision, because the flattening mechanism increases the number of linear programming problems that have to be solved. We discuss this phenomenon in greater detail in Section 8.7.7.

The applications		Time for $V_n(\mathcal{P})$			
system	description	imp.	stat.	dyn.	Cay.
(3.27)	camera motion	1.2	20.6	20.0	9.0
(3.28)	PUMA robot	14.6	3.5	10.2	32.9
(3.29)	six-revolute-joint	2.0	100.8	292.2	–
(5.66)	combustion 3000°	0.2	13.7	3.4	113.7
(5.67)	chem. equilibrium	2.7	1.5	1.4	4.0
(5.70)	benchmark i1	15.0	5.9	11.2	–
(5.71)	integration formula	–	796.2	561.3	–
(6.36)	inverse position	–	662.3	7,833.6	–
(6.37)	heart-dipole	–	90.4	869.3	–
(6.38)	Butcher's problem	51.8	14.5	18.5	140.5
(7.45)	neurofysiology	0.3	0.2	0.3	1.0

Table 8.21: Comparison between implicit, static and dynamic lifting methods. We list timings for implicit (imp.), static (stat.), dynamic (dyn.) lifting and for the Cayley trick.

8.7.6 The design of a planar four-bar mechanism

In this section, we illustrate the usefulness of polynomial homotopy continuation to a specific design problem of mechanics. In [119], polynomial continuation is applied to design a planar four-bar mechanism. The parameters of the system are the five prescribed positions for the coupler curve to pass through. These positions are given by the so-called precision points. We have applied our software to re-calculate the experiments presented in [119].

Solving the random coefficient system takes 16.62 cpu seconds. Note that we first have extracted the nonvertex points, which takes only 0.15 cpu sec., and yields a system that is a lot sparser to deal with. It takes 176.70 cpu seconds to solve the generic problem instance, that is the system, whose coefficients are given in Table 2, in [119]. By solving the random coefficient system, we enter the coefficient space of the polynomial system. The next step, solving a generic instance of the parameters, leads us into the parameter space. This stage is more expensive, because only 36 of the 80 paths converge to finite solutions.

In Table 8.23, we list our computational experiments on twenty different problem instances, with values for the parameters given in [119]. Each time we used the generic problem as start system.

The problems are classified in four groups. The problems 1 to 6 are generic, with no special relations between the precision points. The second group, problems 7 through 11, consists of so-called straight-line problems, with all precision points on a line. All these systems have a solution at infinity with multiplicity three. Degenerate problems, with precision points taken on a circle (problem 12), or close to each other (problems 13 and 15), or with precision points equal to each other (problems 14 and 16). For problems 14 and 16, we had to compute winding numbers, because the general continuation was insufficient to reach the singular solutions. The fourth and last group, problems 17

The applications		Time for continuation			
system	description	imp.	stat.	dyn.	cont.
(3.27)	camera motion	29.7	30.4	28.7	55.3
(3.28)	PUMA robot	6.9	9.7	3.7	12.1
(3.29)	six-revolute-joint	38.4	89.2	60.3	135.2
(5.66)	combustion 3000°	4.9	6.5	5.0	17.9
(5.67)	chem. equilibrium	6.0	8.1	4.1	15.3
(5.70)	benchmark i1	443.2	63.4	50.5	118.1
(5.71)	integration formula	–	1,289.4	482.2	758.3
(6.36)	inverse position	–	1,209.3	1,030.2	2,266.0
(6.37)	heart-dipole	–	808.5	628.6	550.5
(6.38)	Butcher's problem	29.1	39.1	40.7	77.9
(7.45)	neurofysiology	4.0	6.6	5.9	17.3

Table 8.22: Comparison between polyhedral homotopies induced by implicit, static and dynamic lifting methods. We list timings for path following defined by implicit (imp.), static (stat.) and dynamic (dyn.) lifting. The last column gives the times for solving the target system.

through 20, represents symmetrical problems, generated by symmetrical arrangements of the precision points.

In Table 8.23, we see that the timings are more or less uniform for the first two groups. The third group of degenerate problems contains the hardest systems to solve. Note that even in the case that the mixed volume equals zero, we are able to compute solutions of the system. Especially the cost of the necessary computation of the winding numbers raised the overall time. The last group also contains great differences among the execution times, because of the additional deficiency introduced by the symmetric structure.

no.	cont.	$V_n(\mathcal{P})$	$N_\mathbb{C}(F)$	$N_\mathbb{R}(F)$	no.	cont.	$V_n(\mathcal{P})$	$N_\mathbb{C}(F)$	$N_\mathbb{R}(F)$
1	127.10	80	36	8	11	143.80	80	33	13
2	147.31	80	36	16	12	84.74	0	36	0
3	106.62	80	36	18	13	347.16	80	22	6
4	88.33	80	36	18	14	465.45	80	36	0
5	99.72	80	36	20	15	442.83	80	36	11
6	107.60	80	36	24	16	568.84	80	5	1
7	131.58	80	33	15	17	155.20	64	18	6
8	134.53	80	33	1	18	69.87	64	36	3
9	126.06	80	33	1	19	124.16	80	36	12
10	145.99	80	33	15	20	122.84	64	24	13

Table 8.23: Solving specific problem instances for the design of a planar four-bar mechanism. For each problem, we list the time needed for continuation (cont.), the BKK bound $V_n(\mathcal{P})$, the number of complex and real solutions.

8.7.7 Controlling the height of the lifting

The cyclic n-roots problem is one of the most popular benchmark systems. In this section we try to reveal some important practical aspects of the mixed-volume computation. In Table 8.24, we see the results of the static lifting algorithm, applied with integer random lifting values, once generated in $[0, 77]$ and once in $[0, 17]$, for $n = 5$ up to 9.

n	5	6	7	8	9
$V_n(\mathcal{P})$	70	156	924	2,560	11,016
time, in $[0, 77]$	0.6	5.7	76.4	802.6	11,247.7
#cells	13	24	120	294	1,041
#cells to refine	0	6	5	18	172
avg time pro cell	0.05	0.19	0.61	2.57	9.27
time, in $[0, 17]$	0.6	5.4	84.1	918.5	13,651.2
#cells	13	22	115	256	953
#cells to refine	0	7	21	77	392
avg time pro cell	0.05	0.19	0.62	2.76	10.15

Table 8.24: The influence of the height of the lifting on static lifting. For $n = 5$ up to 9, timings (cpu sec.) are given. We also list the number of cells, the number of cells needed as refinement and the average time pro cell, for lifting values generated at random once in $[0, 77]$ and once in $[0, 17]$.

We see in Table 8.24, that as the dimension increases, the average time needed to compute one cell, is multiplied with the factor between three and four. When we reduce the height of the lifting function, we see that the computational time increases, but that the structure of the subdivision becomes more complicated, that we need more cells to refine cells that are not fine mixed. By taking all computed cells into account, we see that the average time to compute one cell remains of the same order.

In Table 8.25, we collected our computational experiences from the dynamic lifting algorithm, applied for $n = 5$ up to 8, once with no flattening and once with flattening when lifting values higher than 17 are needed. We see that the lifting grows indeed steeper and steeper, when no flattening is applied. Here we observe the same factor of growth for the average time needed to compute one cell, when we increase the dimension.

n	5	6	7	8
$V_n(\mathcal{P})$	70	156	924	2,560
time, no flattening	0.8	6.6	95.5	977.6
growth of lifting	4	20	68	3457
#cells	14	26	126	310
#cells to refine	2	2	29	21
avg time pro cell	0.01	0.24	0.66	2.95
time, flatten at 17	0.9	6.6	200.0	2,910.8
#cells	14	24	80	144
#cells to refine	0	7	95	221
avg time pro cell	0.06	0.21	1.14	8.06

Table 8.25: The influence of the height of the lifting on dynamic lifting. For $n = 5$ up to 8, timings (cpu sec.) are given with the number of cells, the number of cells needed as refinement and the average time pro cell, once done with no flattening and once with a flattening when lifting values higher than 17 are needed. With no flattening, we gave the largest lifting value to indicate the growth of the lifting function.

We see from Table 8.25 that the time needed for the dynamic lifting when we want to apply flattening increases dramatically. The explanation for this phenomenon can be found when we look to the structure of the subdivision. We see that the number of cells decreases, for cyclic 8, at the first level we only have 144 cells instead of 310 cells. This implies that many edges span the same cells and that consequently many face-face combinations yield the same cell. Hence more linear programming problems need to be solved, as is illustrated by Table 8.26.

As part of the output of our program, we obtain the statistics about the face-face combinations that have been made by the pruning algorithm. Based on these statistics, listed in Table 8.26, we have computed a total cost estimate as the sum of $(n - i + 1)^3$ times the number of linear programming problems, for $i = 2, \ldots, n$. This cost estimate corresponds roughly to the cost estimates we have given in Chapter 5 on the pruning algorithm.
From Table 8.26, we see that the number of linear programming problems to be solved in the dynamic lifting algorithm exceeds by far the corresponding number of the static lifting algorithm. The total cost estimate of dynamic lifting is about three times as high as with the static lifting algorithm. This corresponds to the difference in execution time between static and dynamic lifting for the cyclic 8-roots problem in Table 8.25.

	static in $[0, 77]$		dynamic flat at 17	
i	#succ	#fail	#succ	#fail
1	$2.800\ 10^1$	–	$1.860\ 10^2$	–
2	$3.840\ 10^2$	$4.000\ 10^2$	$1.677\ 10^3$	$1.347\ 10^3$
3	$2.249\ 10^3$	$8.503\ 10^3$	$8.405\ 10^3$	$2.189\ 10^4$
4	$4.627\ 10^3$	$5.160\ 10^4$	$1.859\ 10^4$	$1.235\ 10^5$
5	$4.871\ 10^3$	$1.247\ 10^5$	$3.602\ 10^4$	$2.822\ 10^5$
6	$1.930\ 10^3$	$1.345\ 10^5$	$4.756\ 10^4$	$5.459\ 10^5$
7	$3.160\ 10^2$	$5.372\ 10^4$	$2.198\ 10^4$	$7.831\ 10^5$
8	$3.160\ 10^2$	–	$2.198\ 10^4$	–
cost LP	$2.2028161\ 10^7$		$6.8172342\ 10^7$	

Table 8.26: Fastest mixed-volume computation of cyclic 8 versus the slowest. We list for each level the number of linear programming models that lead to successful and failing face-face combinations. At the end we give a total cost estimate.

By this experiment we demonstrated how the growth of the lifting values can be controlled either by the static lifting mechanism, or by the flattening process applied in the dynamic lifting algorithm. The current implementation of dynamic lifting has no facility for dealing with the fact that many edges span the same cells and is hence more expensive.

8.7.8 Solving the reduced cyclic n-roots problem

We have solved the reduced cyclic n-roots problem twice, once with the exploitation of the symmetry and once with a generic black-box polyhedral continuation. The latter method applies dynamic lifting and has a loose setting of the continuation parameters. The timings are displayed in Table 8.27.

	Symmetric lifting				Black box solving		
n	smvc	gphc	poco	total	bphc	bpoco	total
5	0.64	3.26	2.69	6.59	6.25	5.23	11.48
6	12.41	17.67	10.10	40.18	36.52	25.68	62.20
7	101.90	283.50	171.55	556.95	1,266.34	276.86	1,543.10
8	1,803.91	1,074.82	870.47	3,749.20	–	–	–

Table 8.27: Execution times (cpu sec.) for solving the reduced cyclic n-roots problem, once with symmetric lifting and once with a generic black-box polyhedral continuation. First we respectively list the timings for constructing the mixed subdivision, for tracing the generating paths and for solving the target system. The column 'total' contains the added times. For the black-box solver, the timings for the subdivision and the start system are given in the column 'bphc', while the next column displays the continuation to the target system.

From Table 8.27 we see that the black-box solver performs well for $n = 5, 6$, but has difficulties for larger problems. The timings for the exploitation of the symmetry are quite good, although improvements can be made for the construction of the symmetric subdivision. The current implementation of symmetric lifting is only a pre-processor to the pruning stage of the static lifting method. That means that all cells are computed, not just the generating ones.

The cyclic 8-roots problem is a very interesting one. Not only is it deficient w.r.t. the mixed volume, but it also has infinite components of solutions. Still, not only did our program compute all regular solutions, but it also managed to find representatives of the components of solutions.

To conclude we can say that the reduction of the problem does not decrease the computational cost for computing a subdivision, though it reduces the number of paths that need to be followed. Moreover, we can exploit the symmetry, so that we can compute only the generating solutions.

8.8 Conclusions and future directions

In this chapter we have presented the structure of a package for solving polynomial systems by homotopy continuation. One of the main features of the package is the wide range of available homotopy methods, suitable to deal with various structures, such as dense, symmetric, product and sparse polynomial systems. Hereby practical evidence for the feasibility of the investigated methods has been given.

We mention two main directions for further research along the lines of this chapter. The first direction consists of the elaboration of the package as solver based on homotopy continuation. Although the performance results are not that bad, the package needs further elaboration because not every feature uses the most optimal algorithm for its implementation. Furthermore, the package has been written in Ada 83, not using the full OOP capabilities of the new standard.

Another interesting direction could be the application of the developed methods to one application area. This would involve the construction of dedicated homotopies and the careful tune-up of continuation methods. In [131], it is indicated that the homotopy methods developed by Morgan, Sommese and Wampler, have yielded a significant progress to mechanical design problems, see e.g. [113], [177] and [178]. It would be worthwhile to investigate whether the progress made in solving sparse polynomial systems could be useful to solve engineering problems. In order to be effective, the problem formulations should be generated to be as sparse as possible, whereas previous formulations were more focussed on obtaining lower degree equations.

Chapter 9

Conclusions and future research

In this chapter we try to indicate the scientific relevance of our work. Hereby we conclude half a decade of research on homotopy methods, aimed at the development of more performant homotopies.

We have investigated two different classes of root counting methods and implemented automatic solvers, based on various homotopy methods. As a first main result, we established the connection between a permanent and a linear-product system. This insight fuelled further generalized applications of Bézout's theorem. Secondly, we developed various lifting methods to solve sparse polynomial systems more efficiently, following the lines of Bernshteĭn's theorem. The dynamic lifting algorithm provides a stable polyhedral homotopy continuation. Furthermore, we have adapted both the product and polyhedral homotopies to exploit symmetry relations, to construct symmetric homotopies.

The homotopy methods we developed have been implemented and tested on a wide range of applications. Hereby we provide a workbench that offers various homotopy continuation methods. The practical results of our computational experiences indicate the effectiveness of the investigated methods.

To illustrate the impact of the conducted research, we can give the following three examples. In [175], Wampler called our algorithm (presented in [167]) to compute a multi-homogeneous Bézout number 'the basic algorithm', and refined it by developing the row expansion algorithm for computing permanents. Li, Wang and Wang proposed in [88] a new method for generating set structures that yield in many cases a lower Bézout bound than the heuristic we presented in [163]. The dynamic lifting algorithm has been implemented by Huber in his package Pelican [70].

In the next section, we enumerate the main accomplishments of this work once more. Finally, we conclude by suggesting three research projects, which can be seen as a continuation of the current work.

9.1 An overview of the main results

We briefly summarize the main results of this thesis, by listing them as seven independent items. Note that the classification below does not correspond with the structure of the chapters.

1. *Nonlinear reduction.* ([161]) The use of S-polynomials for reducing the total degree of a polynomial system has proven to be effective. At the same time, it shows how concepts of symbolic elimination can be applied and combined in a systematic way with homotopy methods.

2. *Generalized Bézout's theorem.* ([160], [167], [168], [163]) The formal computation of an m-homogeneous Bézout number, based on a table of degrees yielded a big step towards the further generalization of this idea: counting the roots according to Bézout's theorem corresponds to considering a product structure.

3. *Mixed-volume computation is effective.* ([171]) Bernshteĭn's theorem is of crucial importance for dealing with sparse polynomial systems. We came to realize this in full, while implementing the polyhedral homotopy methods.

4. *Symmetric polynomial systems.* ([164], [165]) For both product and polyhedral homotopy methods, we have investigated their aptitude for exploiting symmetry. Symmetric polynomial systems provided the motivation to develop set structure analysis.

5. *The dynamic lifting algorithm.* ([166]) Dynamic lifting leads to stable homotopy continuation and incremental polynomial system solving. It is of independent interest for computing the volume of a polytope.

6. *A database of applications.* The development of new homotopy methods has always been motivated by examples of polynomial systems, derived from practical applications. Consequently we have constructed a considerable large database of interesting polynomial systems, that can serve for benchmarking purposes.

7. *Software.* ([159], [162]) Not only have we implemented all investigated methods, but we have also striven to integrate the methods into one package, whose tools construct a powerful workbench.

While the first five items represent the elaboration of specific methods to a particular class of problems, the last items are oriented towards the integration of methods. Seldomly there is only one single method that is optimal for all practical applications. Therefore we have to create a multi-functional software package that enables the combination of various features of several methods.

9.2 Future research projects

At the end of each chapter some directions for future research were sketched. Here we attempt to list some projects of a larger scale. Without being formal, we just mention the basic ideas and give some simple examples.

9.2.1 Homotopies with constraints

An application normally also imposes additional constraints on the polynomial system. We can distinguish two special instances of such constraints. Firstly, coefficients are related to each other, e.g. some of them are parameters, others not. Secondly, the system might be overconstrained, i.e., there are more equations than unknowns.

In [111], Morgan and Sommese defined a hierarchy of homotopies, based on the performance of the root counts, which range between the total degree and the number of solutions of a system with a generic choice of the parameters. To deal with the additional equations of an overconstrained system, the so-called side conditions, they proposed to use these 'to weed out' the set of start solutions of the homotopy. A similar approach to handle parametric systems was independently and simultaneously presented by Li, Sauer and Yorke in [87], with the introduction of the 'cheater's homotopy'. The term 'cheater' refers to the intrinsic quality of the approach and motivates the future research project.

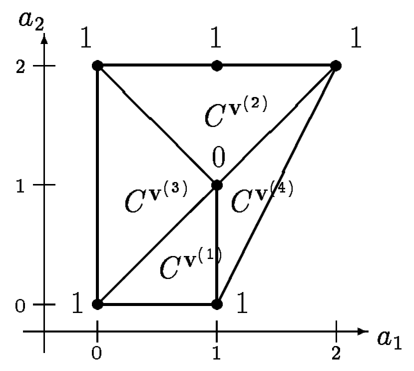
By polyhedral continuation, we have an effective method to enter the coefficient space of the system. To solve a random coefficient system, we have to follow an optimal number of paths. However, the number of solutions of the system in parameter space might be dramatically lower than the mixed volume, hereby reducing the performance of the homotopy continuation solver to a disappointing rate. The question is: could we directly take the additional constraints on the system into account and construct a random coefficient-parameter system efficiently, without cheating? One place to start is Bernshteĭn's second theorem, see Theorem 5.1.17, which gives conditions on the exactness of the mixed volume as root count for a particular polynomial system.

Recall Example 5.1.18, which presented an instance of a deficient polynomial system. For this particular example we can construct a subdivision w.r.t. the deficiency, which leads to a deficiency preserving homotopy.

Example 9.2.1 Consider again the system F presented in (5.9) with Newton polytope P pictured in Figure 5.2. As $C^{\mathbf{v}^{(1)}}$ we have $\{(0,0), (1,0), (1,1)\}$, whereas $C^{\mathbf{v}^{(2)}} = \{(0,2), (1,2), (2,2), (1,1)\}$. In Figure 9.1, the contributing sets $C^{\mathbf{v}^{(1)}}$ and $C^{\mathbf{v}^{(2)}}$ are pictured as cells in a triangulation of P.
In Figure 9.1, a triangulation is induced by constructing the contributing sets $C^{\mathbf{v}^{(1)}}$ and $C^{\mathbf{v}^{(2)}}$. These contributing sets entirely contain the defective faces. This allows to derive the root count as sum of all the root counts $N(F_C)$ of the restricted systems F_C.

In order to take the deficiency of the system into account, the randomized system must have the same defective faces as the system F. Therefore, $F^{(1)}$ provides a better start system to solve F, than the general random coefficient start system.

$$F^{(1)}(\mathbf{x}) = \left\{ \begin{array}{l} c^{(1)}_{\mathbf{v}^{(2)}}(x_1^2x_2^2 - 3x_1x_2^2 + 2x_2^2) + c^{(1)}_{(1,1)}x_1x_2 + c^{(1)}_{\mathbf{v}^{(1)}}(x_1+1) = 0 \\ c^{(2)}_{\mathbf{v}^{(2)}}(x_1^2x_2^2 - 4x_1x_2^2 + 3x_2^2) + c^{(2)}_{(1,1)}x_1x_2 + c^{(2)}_{\mathbf{v}^{(1)}}(2x_1+2) = 0 \end{array} \right. \tag{9.1}$$



C	$N(F_C)$	$V_n(C)$
$C^{\mathbf{v}^{(1)}}$	0	1
$C^{\mathbf{v}^{(2)}}$	1	2
$C^{\mathbf{v}^{(3)}}$	2	2
$C^{\mathbf{v}^{(4)}}$	1	1
	4	6

Figure 9.1: The triangulation induced by a lifting that respects the deficiency. It suffices to count the roots of the systems F_C corresponding to the cells C.

By applying the polyhedral homotopy induced by the lifting used to create the triangulation presented in Figure 9.1. The cell $C^{\mathbf{v}^{(1)}}$ can be discarded, as the solution of $\partial_{\mathbf{v}^{(1)}} F(\mathbf{x}) = \mathbf{0}$ remains invariant for all t. In order to know the only solution in $\mathbb{C}_0^2$ of $\partial_{\mathbf{v}^{(2)}} F(\mathbf{x}) = \mathbf{0}$, two paths need to be followed, though, as $\partial_{\mathbf{v}^{(2)}} F$ is sparser than F, with less work to do. Finally, all four isolated solutions of $F(\mathbf{x}) = \mathbf{0}$ in $\mathbb{C}_0^2$ can be obtained by following the paths defined by the usual convex-linear homotopy with $F^{(1)}(\mathbf{x}) = \mathbf{0}$ as start system. ∎

The above example stimulates the development of deficiency preserving homotopies, based on Bernshteĭn's second theorem. Two important difficulties need to be surmounted. Firstly, subdivisions with constraints are harder to compute than subdivisions induced by a random lifting by the current methods that are available. Secondly, there exists no straightforward theoretical evidence that all isolated solutions can be found by simply applying a polyhedral homotopy with the same type of deficiency as the target system.

Furthermore, not only the exploitation of additional constraints on the solutions are important, but also the ability to restrict the computations to the interesting solutions. For example, one is mainly interested in solutions within some bounded domain. The application of Rouché's theorem, see [169], provides already some answers, although the concept of regional homotopies certainly deserves further elaboration.

9.2.2 Computing only the real roots

Based on mixed subdivisions and polyhedral continuation, the number of real roots can be counted asymptotically, i.e.: for infinitesimal values of the continuation parameter. The construction of a real polynomial system with a maximal number of real roots for a given sign-structure could provide some justification for a multivariate rule of Descartes. This goal could be achieved best by applying homotopy continuation methods.

In [126], Viro presented a method for constructing real algebraic plane curves with a certain prescribed topology. The gluing procedure he described consists in taking convex functions to construct a one-parameter family of systems. The topology is realized asymptotically for sufficiently small values of the parameters. Sturmfels applied the Viro method to the construction of real complete intersections with a prescribed topology, in [147] for the unmixed and in [148] for the fully mixed case. A generalized rule of Descartes has been conjectured by Itenberg and Roy in [73]. They propose an elegant way for counting the asymptotic number of real roots for a given configuration of signed Newton diagrams. To obtain an upper bound, one has to take the maximum of asymptotic real root counts by enumerating all subdivisions.

The conjecture of Itenberg and Roy holds in the univariate case and in the case when their bound equals the mixed volume. So for the latter case, we have already a solid ground to investigate whether we can construct a random start system with the maximum number of real roots and whether there exists a homotopy that can be used to compute all real roots. By the following example, we present the challenging situation, where the number of Itenberg and Roy is always lower than the mixed volume.

Example 9.2.2 In Figure 9.2, we show the last example given [73]. The mixed volume of (P_1, P_2) equals five, whereas for any assignment of signs, the bound of Itenberg and Roy equals three. Any subdivision that would yield a real polyhedral homotopy with more than three roots should have at least three cells, because of the restriction that the mixed volume equals five. In the case of three cells, we can find only two different possible subdivisions, as visualized in Figure 9.2. Both configurations have the decomposition $5 = 1 + 1 + 3$, so there is at least one pair of complex conjugate roots.

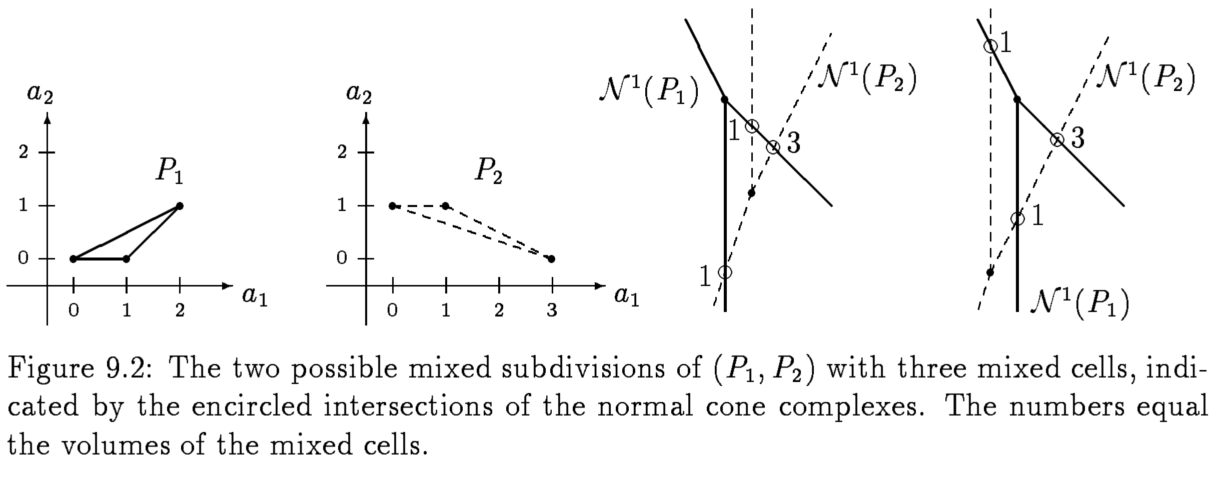


Figure 9.2: The two possible mixed subdivisions of (P_1, P_2) with three mixed cells, indicated by the encircled intersections of the normal cone complexes. The numbers equal the volumes of the mixed cells.

Examples of this kind motivate the application of real root counting techniques, when one expects that the number of real roots will be significantly smaller than the mixed volume. ∎

Note the usefulness of normal cone complexes when searching for the finest possible subdivision. We believe that by polyhedral homotopy continuation the conjecture of the multivariate Descartes' rule can be resolved best, because these tools are all closely related to the Viro method.

9.2.3 Straight-line programs

Modeling the structure of a polynomial system by means of Newton polytopes has led to a computational breakthrough for solving sparse polynomial systems. Could it be possible to develop homotopy methods for systems of polynomials, with the polynomials given by straight-line programs?

A straight-line program is a model of an optimal algorithm for evaluating a multivariate polynomial. This model is an arithmetical network. In the complexity analysis of elimination theory, better complexity results have been obtained in terms of the size of the arithmetical network. We refer to [58], [59], [60] and [127] for more on the connection between straight-line programs and the complexity of solving polynomial systems.

A direct application of an optimal evaluator for multivariate polynomials lies in the continuation stage, where a large amount of arithmetical work is consumed in the evaluation of polynomials. Note that the existence of straight-line programs justifies to measure the complexity of homotopy continuation by $O(Rf(n))$ with R the root count and $f(n)$ a polynomial in n as complexity of evaluating n-variate polynomials, because most polynomial systems in practice are sparse and can be evaluated in polynomial time in n. But a straight-line program can also reveal important information about the structure of the problem that leads to a significant reduction of the solving cost, as is illustrated by the following simple example.

Example 9.2.3 Consider the following polynomial system:

$$F(x,y,z) = \left\{ \begin{array}{c} x^2y^2 - z^2 + y^4 - 1 = 0 \\ x^2 + y^2 - z^2 + x^4 + 1 = 0 \\ x^2 + z^4 + y^4 - 1 = 0 \end{array} \right. . \tag{9.2}$$

To evaluate the polynomials in the system efficiently, we better first compute x^2, y^2 and z^2, and apply thereafter the other necessary arithmetical operations to evaluate. Analogously to the evaluation algorithm, we can substitute x^2, y^2 and z^2 respectively by $\tilde{x}$, $\tilde{y}$ and $\tilde{z}$, before solving the system. The advantage to homotopy continuation is straightforward: the root count will decrease and hence also the number of paths to be traced. ■

The task in this project is to develop systematic ways by which homotopy continuation methods can exploit the concept of straight-line programs to arrive at a better performance.

Bibliography

[1] W.A. Adkins and S.H. Weintraub. *Algebra, an Approach via Module Theory.* Springer–Verlag, New York, 1992.

[2] E.L. Allgower and K. Georg. *Numerical Continuation Methods, an Introduction*, volume 13 of *Springer Ser. in Comput. Math.* Springer–Verlag, Berlin Heidelberg New York, 1990.

[3] E.L. Allgower and K. Georg. Continuation and path following. *Acta Numerica*, pages 1–64, 1993.

[4] D.C.S. Allison, A. Chakraborty, and L.T. Watson. Granularity issues for solving polynomial systems via globally convergent algorithms on a hypercube. *J. of Supercomputing*, 3:5–20, 1989.

[5] D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete Comput. Geom.*, 8(3):295–313, 1992.

[6] J. Backelin and R. Fröberg. How we proved that there are exactly 924 cyclic 7-roots. In *Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation*, pages 101–111. ACM, 1991.

[7] J.G.P. Barnes. *Programming in Ada, Plus an Overview of Ada 9X.* Addison–Wesley, 1994.

[8] M. Beckers. Oplossen van stelsels veeltermvergelijkingen met behulp van continueringsmethodes. Master's thesis, K.U.Leuven, 1985.

[9] D.N. Bernshteĭn. The number of roots of a system of equations. *Functional Anal. Appl.*, 9(3):183–185, 1975. Translated from *Funktsional. Anal. i Prilozhen.*, 9(3):1–4,1975.

[10] U. Betke. Mixed volumes of polytopes. *Arch. Math.*, 58:388–391, 1992.

[11] L.J. Billera, P. Filliman, and B. Sturmfels. Constructions and complexity of secondary polytopes. *Adv. in Math.*, 83:155–179, 1990.

[12] L.J. Billera and B. Sturmfels. Fiber polytopes. *Ann. of Math.*, 135(3):527–549, 1992.

[13] G. Björk and R. Fröberg. A faster way to count the solutions of inhomogeneous systems of algebraic equations, with applications to cyclic n-roots. *J. Symbolic Computation*, 12(3):329–336, 1991.

[14] G. Björk and R. Fröberg. Methods to "divide out" certain solutions from systems of algebraic equations, applied to find all cyclic 8-roots. In M. Gyllenberg and L.E. Persson, editors, *Analysis, Algebra and Computers in Math. research*, volume 564 of *Lecture Notes in Mathematics*, pages 57–70. Dekker, 1994.

[15] H. Blomberg and R. Ylinen. *Algebraic Theory for Multivariate Linear Systems*, volume 166 of *Mathematics in Science and Engineering.* Academic Press, London, 1983.

[16] L. Blum, F. Cucker, M. Shub, and S. Smale. Complexity and real computation: A manifesto. *International Journal of Bifurcation and Chaos*, 6(1):3–26, 1996.

[17] T. Bonnesen and W. Fenschel. *Theorie der konvexen Körper.* Springer–Verlag, Berlin Heidelberg New York, 1934.

[18] G. Booch. *Software Engineering with Ada.* Benjamin/Cummings Publishing Company, 1987.

[19] P. Brunovský and P. Meravý. Solving systems of polynomial equations by bounded and real homotopy. *Numer. Math.*, 43(3):397–418, 1984.

[20] A.D. Bryuno and A. Soleev. Local uniformization of branches of a space curve, and Newton polyhedra. *St. Petersburg Math. J.*, 3(1):53–82, 1992. Translated from *Algebra i Analiz*, 3(1):67–101, 1991.

[21] B. Buchberger. Gröbner bases: An algorithmic method in polynomial ideal theory. In N.K. Bose, editor, *Multidimensional System Theory*, chapter 6, pages 184–232. D. Reidel Publishing Company, Dordrecht Boston Lancaster, 1985.

[22] Yu.D. Burago and V.A. Zalgaller. *Geometric Inequalities*, volume 285 of *Grundlehren der mathematischen Wissenschaften.* Springer–Verlag, Berlin Heidelberg, 1988.

[23] R.B. Burckel. *An introduction to classical complex analysis I.* Academic Press, 1979.

[24] A.R. Calderbank, R.H. Hardin, and N.J.A. Sloane. On the construction of isometric embeddings. In preparation.

[25] J. Canny and J.M. Rojas. An optimal condition for determining the exact number of roots of a polynomial system. In *Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation*, pages 96–101. ACM, 1991.

[26] J.F. Canny and I. Emiris. An efficient algorithm for the sparse mixed resultant. In G. Cohen, T. Mora, and O. Moreno, editors, *Proceedings of the 10th International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*,

volume 263 of *Lecture Notes in Computer Science*, pages 89–104. Springer–Verlag, 1993.

[27] S.N. Chow, J. Mallet-Paret, and J.A. Yorke. Homotopy method for locating all zeros of a system of polynomials. In H.O. Peitgen and H.O. Walther, editors, *Functional differential equations and approximation of fixed points*, volume 730 of *Lecture Notes in Mathematics*, pages 77–88, Berlin Heidelberg, 1979. Springer–Verlag.

[28] M. Chu, T.Y. Li, and T. Sauer. Homotopy method for general λ–Matrix problems. *SIAM J. Matrix Anal. Appl.*, 9(4):528–536, 1988.

[29] K.L. Clarkson, K. Melhorn, and R. Seidel. Four results on randomized incremental constructions. In A. Finkel and M. Jantzen, editors, *STACS 92, 9th Annual Symposium on Theoretical Aspects of Computer Science, Cachan, France, February 1992, Proceedings*, volume 577 of *Lecture Notes in Computer Science*, pages 463–474, Berlin Heidelberg, 1992. Springer–Verlag.

[30] J. Cohen and T. Hickey. Two algorithms for determining volumes of convex polyhedra. *J. Assoc. Comput. Mach.*, 26(3):401–414, 1979.

[31] R. Cools. A survey of methods for constructing cubature formulae. In T.O. Espelid and A. Genz, editors, *Numerical Integration, Recent Developments, Software and Applications*, volume 357 of *NATO ASI Series C: Math. and Phys. Sciences*, pages 1–24, Dordrecht, 1992. Kluwer.

[32] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties and Algorithms. An Introduction to Computational Algebraic Geometry and Commutative Algebra.* Undergraduate Texts in Mathematics. Springer–Verlag, New York, 1992.

[33] J.A. de Loera. *Triangulations of Polytopes and Computational Algebra.* PhD thesis, Cornell University, 1995.

[34] J.P. Dedieu and J.C. Yakoubsohn. Two seminumerical algorithms for solving polynomial systems. Manuscript, 1994.

[35] R. Dewar. GNAT Documents, Intro Unix Version. New York University, 1993.

[36] F.J. Drexler. Eine Methode zur Berechnung sämtlicher Lösungen von Polynomgleichungssystemen. *Numer. Math.*, 29(1):45–58, 1977.

[37] M.E. Dyer and A.M. Frieze. On the complexity of computing the volume of a polyhedron. *SIAM J. Comput.*, 17(5):967–974, 1988.

[38] P. Dyer, M. Gritzmann and A. Hufnagel. On the complexity of computing mixed volumes. To appear in *SIAM J. Comput.*

[39] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *ETACS Monographs on Theoretical Computer Science.* Springer–Verlag, Berlin, Heidelberg, New York, 1987.

[40] H. Edelsbrunner and N.R. Shah. Incremental topological flipping works for regular triangulations. In *Proceedings of the Eight Annual Symposium on Computational Geometry*, pages 43–52. ACM, 1992.

[41] I. Emiris and J. Canny. A practical method for the sparse resultant. In *Proceedings of the 1993 International Symposium on Symbolic and Algebraic Computation*, pages 183–192. ACM, 1993.

[42] I. Emiris and J. Canny. Efficient incremental algorithms for the sparse resultant and the mixed volume. Technical report 839, Computer Science Division, U.C. Berkeley, 1994. To appear in *J. Symbolic Computation*.

[43] I.Z. Emiris. *Sparse Elimination and Applications in Kinematics*. PhD thesis, Computer Science Division, Dept. of Electrical Engineering and Computer Science, University of California, Berkeley, 1994.

[44] I.Z. Emiris and J.F. Canny. Efficient incremental algorithms for the sparse resultant and the mixed volume. *J. Symbolic Computation*, 20(2):117–149, 1995.

[45] M.A. Facello. Implementation of a randomized algorithm for Delaunay and regular triangulations in three dimensions. *Computer Aided Geometric Design*, 12(4):349–370, 1995.

[46] S.-C. Fang and S. Puthenpura. *Linear Approximations and Extensions: Theory and Applications*. Prentice Hall, Englewood Cliffs, N.J., 1993.

[47] O.D. Faugeras and S. Maybank. Motion from point matches: multiplicity of solutions. *Intern. J. Comp. Vision*, 4:225–246, 1990.

[48] J.C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *J. Symbolic Computation*, 16(4):329–344, 1993.

[49] G. Fischer. *Complex Analytic Geometry*. Number 538 in Lecture Notes in Mathematics. Springer–Verlag, Berlin Heidelberg New York, 1976.

[50] C.B. Garcia and T.Y. Li. On the number of solutions to polynomial systems of equations. *SIAM J. Numer. Anal.*, 17(4):540–546, 1980.

[51] C.B. Garcia and W.I. Zangwill. Finding all solutions to polynomial systems and other systems of equations. *Math. Programming*, 16(2):159–176, 1979.

[52] M.R. Garey and D.S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. Freemann, San Francisco, 1979.

[53] K. Gatermann. Symbolic solution of polynomial equation systems with symmetry. In Sh. Watanabe and M. Nagata, editors, *Proceedings of ISSAC-90 (Tokyo, Japan, August 20–24, 1990)*, pages 112–119. ACM, New York, 1990.

[54] R. Gebauer and H.M. Möller. On an installation of Buchberger's algorithm. *J. Symbolic Computation*, 6(2–3):275–286, 1988.

[55] K.O. Geddes, S.R. Czapor, and G. Labahn. *Algorithms for Computer Algebra.* Kluwer, Boston, 1992.

[56] I.M. Gel'fand, M.M. Kapranov, and A.V. Zelevinsky. *Discriminants, Resultants and Multidimensional Determinants.* Birkhäuser, Boston, 1994.

[57] I.M. Gel'fand, Zelevinskiĭ, and M.M. Kapranov. Discriminants of polynomials in several variables and triangulations of Newton polyhedra. *Leningrad Math. J.*, 2(3):449–505, 1991.

[58] M. Giusti, J. Heintz, M.E. Morais, J. Morgenstern, and L.M. Pardo. Straight-line programs in geometric elimination theory. Manuscript, 1995.

[59] M. Giusti, J. Heintz, M.E. Morais, and L.M. Pardo. When polynomial equation systems can be "solved" fast? To appear in *Proc. AAECC-11*, Springer LNCS 948, 1995.

[60] M. Giusti, J. Heintz, and J. Sabia. On the efficiency of effective Nullstellensätze. *Comput. Complexity*, 3:56–95, 1993.

[61] P. Griffiths and J. Harris. *Principles of Algebraic Geometry.* John Wiley & Sons, New York, 1978.

[62] P. Gritzmann and V. Klee. Mathematical programming and convex geometry. In P.M. Gruber and J.M. Wills, editors, *Handbook of Convex Geometry*, pages 627–674. North-Holland, 1993.

[63] P. Gritzmann and V. Klee. On the complexity of some basic problems in computational convexity: II. Volume and mixed volumes. In T. Bisztriczky, P. McMullen, R. Schneider, and A.I. Weiss, editors, *Polytopes: Abstract, Convex and Computational*, pages 373–466. Kluwer, Boston, 1994.

[64] P. Gritzmann and B. Sturmfels. Minkowski addition of polytopes: computational complexity and applications to Gröbner bases. *SIAM J. Discrete Math.*, 6(2):246–269, 1993.

[65] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics.* Springer–Verlag, Berlin Heidelberg New York, 1988.

[66] L.J. Guibas, D.E. Knuth, and M Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7(4):381–413, 1992.

[67] V. Guillemin and A. Pollack. *Differential Topology.* Prentice-Hall, Englewood Cliffs (N.J.), 1974.

[68] S. Harimoto and L.T. Watson. The granularity of homotopy algorithms for polynomial systems of equations. In G. Rodrigue, editor, *Parallel processing for scientific computing*, pages 115–120. SIAM, 1989.

[69] B. Huber. Numerically solving sparse polynomial systems. Presented at AMS-IMS-SIAM Summer Research Conference on *Continuous Algorithms and Complexity*, Mount Holyoke College, South Hadley, MA, June 1994.

[70] B. Huber. Pelican manual. Available via the author's web page.

[71] B. Huber and B. Sturmfels. Bernstein's theorem in affine space. To appear in *Discrete Comput. Geom.*

[72] B. Huber and B. Sturmfels. A polyhedral method for solving sparse polynomial systems. *Math. Comp.*, 64(212):1541–1555, 1995.

[73] I. Itenberg and M.-F. Roy. Multivariate Descartes' rule. Manuscript, 1995.

[74] R. Kearfott. Interval arithmetic thechniques in the computational solution of nonlinear systems of equations: Introduction, examples, and comparisons. In E.L. Allgower and K. Georg, editors, *Computational Solution of Nonlinear Systems of Equations*, pages 337–357, Providence, Rhode Island, 1990. AMS.

[75] R. Kearfott and M. Novoa. Algorithm 681: INTBIS, a portable interval newton/bisection package. *ACM Trans. Math. Softw.*, 16(2):152–157, 1990.

[76] A.G. Khovanskiĭ. Newton polyhedra and the genus of complete intersections. *Functional Anal. Appl.*, 12(1):38–46, 1978. Translated from *Funktsional. Anal. i Prilozhen.*, 12(1),51–61,1978.

[77] A.G. Kushnirenko. Newton Polytopes and the Bézout Theorem. *Functional Anal. Appl.*, 10(3):233–235, 1976. Translated from *Funktsional. Anal. i Prilozhen.*, 10(3),82–83,1976.

[78] J. Lawrence. Polytope volume computation. *Math. of Comp.*, 57(195):259–271, 1991.

[79] C.W. Lee. Regular triangulations of convex polytopes. In P. Gritzmann and B. Sturmfels, editors, *Applied Geometry and Discrete Mathematics - The Victor Klee Festschrift*, volume 4 of *DIMACS Series*, pages 443–456. AMS, Providence, R.I., 1991.

[80] K. Leichtweiß. *Konvexe Mengen*. Springer–Verlag, Berlin Heidelberg New York, 1980.

[81] T.Y. Li. On Chow, Mallet-Paret and Yorke homotopy for solving systems of polynomials. *Bulletin of the Institute of Mathematics. Acad. Sin.*, 11:433–437, 1983.

[82] T.Y. Li. Solving polynomial systems. *The Mathematical Intelligencer*, 9(3):33–39, 1987.

[83] T.Y. Li and T. Sauer. Regularity results for solving systems of polynomials by homotopy method. *Numer. Math.*, 50(3):283–289, 1987.

[84] T.Y. Li and T. Sauer. A simple homotopy for solving deficient polynomial systems. *Japan J. Appl. Math.*, 6(3):409–419, 1989.

[85] T.Y. Li, T. Sauer, and J.A. Yorke. Numerical solution of a class of deficient polynomial systems. *SIAM J. Numer. Anal.*, 24(2):435–451, 1987.

[86] T.Y. Li, T. Sauer, and J.A. Yorke. The random product homotopy and deficient polynomial systems. *Numer. Math.*, 51(5):481–500, 1987.

[87] T.Y. Li, T. Sauer, and J.A. Yorke. The cheater's homotopy: an efficient procedure for solving systems of polynomial equations. *SIAM J. Numer. Anal.*, 26(5):1241–1251, 1989.

[88] T.Y. Li, T. Wang, and X. Wang. Random product homotopy with minimal BKK bound. To appear in the proceedings of the AMS-SIAM Summer Seminar in Applied Mathematics, on *Mathematics of Numerical Analysis: Real Number Algorithms*, Park City, Utah, July-August 1995.

[89] T.Y. Li and X. Wang. The BKK root count in C^N. To appear in *Math. Comp.*

[90] T.Y. Li and X. Wang. Solving deficient polynomial systems with homotopies which keep the subschemes at infinity invariant. *Math. Comp.*, 56(194):693–710, 1991.

[91] T.Y. Li and X. Wang. Nonlinear homotopies for solving deficient polynomial systems with parameters. *SIAM J. Numer. Anal.*, 29(4):1104–1118, 1992.

[92] C.L. Liu. *Introduction to Combinatorial Mathematics*. McGraw-Hill Book Company, New York, 1968.

[93] S. MacLane and G. Birkhoff. *Algebra*. The Macmillan Company, New York, 1967.

[94] G. Malajovich. pss 2.alpha, polynomial system solver, version 2.alpha. README file, distributed by the author through gopher.

[95] G. Malajovich. On generalized Newton algorithms: quadratic convergence, path-following and error analysis. *Theoret. Comput. Sci.*, 133(1):65–84, 1994.

[96] D. Manocha and J.F. Canny. Real time inverse kinematics for general 6r manipulators. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*. IEEE, 1992.

[97] K. Meintjes and A.P. Morgan. A methodology for solving chemical equilibrium systems. *Appl. Math. Comput.*, 22(4):333–361, 1987.

[98] K. Meintjes and A.P. Morgan. Chemical equilibrium systems as numerical test problems. *ACM Trans. Math. Softw.*, 16(2):143–151, 1990.

[99] P. Meravý. Symmetric homotopies for solving systems of polynomial equations. *Math. Slovaca*, 39(3):277–288, 1989.

[100] W. Miller. *Symmetric groups and their applications*, volume 50 of *Pure and Applied Mathematics*. Academic Press, New York, 1972.

[101] H. Minc. *Permanents*, volume 6 of *Encyclopedia of Mathematics and its Applications*. Addison–Wesley, Reading, Massachusetts, 1978.

[102] R. Moore. *Methods and Applications of Interval Analysis*. SIAM Studies in Applied Mathematics. SIAM, Philadelphia, 1979.

[103] J.J. Moré, B.S. Garbow, and K.E. Hillstrom. Testing unconstrained optimization software. *ACM Trans. Math. Softw.*, 7(1):17–41, 1981.

[104] A. Morgan. *Solving polynomial systems using continuation for engineering and scientific problems*. Prentice-Hall, Englewood Cliffs, N.J., 1987.

[105] A. Morgan and V. Shapiro. Box-bisection for solving second-degree systems and the problem of clustering. *ACM Trans. Math. Soft.*, 13(2):152–167, 1987.

[106] A. Morgan and A. Sommese. Computing all solutions to polynomial systems using homotopy continuation. *Appl. Math. Comput.*, 24(2):115–138, 1987.

[107] A. Morgan and A. Sommese. A homotopy for solving general polynomial systems that respects m-homogeneous structures. *Appl. Math. Comput.*, 24(2):101–113, 1987.

[108] A.P. Morgan. A method for computing all solutions to systems of polynomial equations. *ACM Trans. Math. Softw.*, 9(1):1–17, 1983.

[109] A.P. Morgan. Polynomial continuation and its relationship to the symbolic reduction of polynomial systems. In *Workshop on the integration of numerical and symbolic computing methods*, Saratoga Springs, New York, 1991. Academic Press.

[110] A.P. Morgan, A. Sommese, and L.T. Watson. Mathematical reduction of a heart dipole model. *J. Comput. Appl. Math.*, 27:407–410, 1989.

[111] A.P. Morgan and A.J. Sommese. Coefficient-parameter polynomial continuation. *Appl. Math. Comput.*, 29(2):123–160, 1989. Errata: *Appl. Math. Comput.* 51:207(1992).

[112] A.P. Morgan and A.J. Sommese. Generically nonsingular polynomial continuation. In E.L. Allgower and K. Georg, editors, *Computational Solution of Nonlinear Systems of Equations*, pages 467–493, Providence, R.I., 1990. AMS.

[113] A.P. Morgan, A.J. Sommese, and C.W. Wampler. Polynomial continuation for mechanism design problems. In E.L. Allgower and K. Georg, editors, *Computational Solution of Nonlinear Systems of Equations*, pages 495–517, Providence, R.I., 1990. AMS.

[114] A.P. Morgan, A.J. Sommese, and C.W. Wampler. Computing singular solutions to nonlinear analytic systems. *Numer. Math.*, 58(7):669–684, 1991.

[115] A.P. Morgan, A.J. Sommese, and C.W. Wampler. Computing singular solutions to polynomial systems. *Adv. Appl. Math.*, 13(3):305–327, 1992.

[116] A.P. Morgan, A.J. Sommese, and C.W. Wampler. A power series method for computing singular solutions to nonlinear analytic systems. *Numer. Math.*, 63:391–409, 1992.

[117] A.P. Morgan, A.J. Sommese, and C.W. Wampler. A product-decomposition theorem for bounding Bézout numbers. *SIAM J. Numer. Anal.*, 32(4):1308–1325, 1995.

[118] A.P. Morgan, A.J. Sommese, and L.T. Watson. Finding all isolated solutions to polynomial systems using HOMPACK. *ACM Trans. Math. Softw.*, 15(2):93–122, 1989.

[119] A.P. Morgan and C.W. Wampler. Solving a planar four-bar design problem using continuation. *ASME J. of Mechanical Design*, 112:544–550, 1990.

[120] D. Mumford. *Algebraic Geometry I; Complex Projective Varieties*, volume 221 of *Grundlehren der mathematischen Wissenschaften.* Springer–Verlag, Berlin Heidelberg New York, 1976.

[121] C.V. Nelsen and B.C. Hodgkin. Determination of magnitudes, directions, and locations of two independent dipoles in a circular conducting region from boundary potential measurements. *IEEE Trans. Biomed. Engrg.*, BME-28(12):817–823, 1981.

[122] M. Newman. *Integral Matrices*, volume 45 of *Pure and Applied Mathematics.* Academic Press, New York, 1972.

[123] V.W. Noonburg. A neural network modeled by an adaptive Lotka-Volterra system. *SIAM J. Appl. Math.*, 49(6):1779–1792, 1989.

[124] T. Oda. *Convex Bodies and Algebraic Geometry: an Introduction to the Theory of Toric Varieties.* Springer–Verlag, Berlin Heidelberg New York, 1988.

[125] J.M. Ortega and W.C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, New York, 1970.

[126] Viro O.Y. Gluing of plane real algebraic curves and constructions of curves of degrees 6 and 7. In L.D. Faddeev and A.A. Mal'cev, editors, *Topology*, volume 1060 of *Lecture Notes in Mathematics*, pages 187–200. Springer–Verlag, 1984.

[127] L.M. Pardo. How lower and upper complexity bounds meet in elimination theory. Manuscript, 1995.

[128] P. Pedersen. Personal communication at the occasion of the AMS-IMS-SIAM Summer Research Conference on *Continuous Algorithms and Complexity*, Mount Holyoke College, South Hadley, MA, June 1994.

[129] P. Pedersen and B. Sturmfels. Product formulas for resultants and Chow forms. *Math. Zeitschrift*, 214(3):377–396, 1993.

[130] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction.* Springer–Verlag, Berlin, Heidelberg, New York, 1985.

[131] M. Raghavan and B. Roth. Solving polynomial systems for the kinematic analysis and synthesis of mechanisms and robot manipulators. *ASME J. of Mechanical Design*, 117:71–79, 1995.

[132] O. Ragos, M.N. Vrahatis, and F.A. Zafiropoulos. The computation of equilibrium points of up-to-three dimensional dynamical systems using the topological degree theory. Presented at the Fourth International Colloquium on Numerical Analysis, Plovdiv, Bulgaria, 13-17 August 1995.

[133] J.M. Rojas. A convex geometric approach to counting the roots of a polynomial system. *Theoret. Comput. Sci.*, 133(1):105–140, 1994.

[134] J.M. Rojas and X. Wang. Counting affine roots roots of polynomial systems via pointed Newton polytopes. To appear in *J. of Complexity.*

[135] R. Schneider. *Convex Bodies: The Brunn-Minkowski Theory*, volume 44 of *Encyclopedia of Mathematics and its Applications.* Cambridge University Press, Australia, 1993.

[136] R. Schneider. Polytopes and Brunn-Minkowski theory. In R. Schneider T. Bisztriczky, P. McMullen and A.I. Weiss, editors, *Polytopes: Abstract, Convex and Computational*, pages 273–300. Kluwer, Boston, 1994.

[137] S. Schrans and W. Troost. Generalized Virasoro constructions for SU(3). *Nuclear Phys. B*, 345(2–3):584–606, 1990.

[138] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Chichester, 1986.

[139] R. Seidel. Constructing higher-dimensional convex hulls at logarithmic cost per face. In *Proc. of the 18th ACM Symp. on Theory of Computing*, pages 404–413, 1986.

[140] I.R. Shafarevich. *Basic Algebraic Geometry 1: Varieties in Projective Space.* Springer–Verlag, Berlin Heidelberg New York, 1994.

[141] M. Shub and S. Smale. Complexity of Bézout's Theorem I: Geometric Aspects. *J. Amer. Math. Soc.*, 6(2):459–501, 1993.

[142] Z. Smilansky. Decomposability of polytopes and polyhedra. *Geom. Dedicata*, 24:29–49, 1987.

[143] A.J. Sommese and C.W. Wampler. Numerical algebraic geometry. To appear in the proceedings of the AMS-SIAM Summer Seminar in Applied Mathematics, on *Mathematics of Numerical Analysis: Real Number Algorithms*, Park City, Utah, July-August 1995.

[144] M. Sosonkina, L.T. Watson, and D.E. Stewart. Note on the end game in homotopy zero curve tracking. To appear in *ACM Trans. Math. Softw.*

[145] M.C. Steenkamp. Die numeriese oplos van stelsels polinoomvergelykings. Technical report, Nasionale Navorsingsinstituut vir Wiskundige Wetenskappe, Pretoria, 1982.

[146] A.H. Stroud and D. Secrest. *Gaussian Quadrature Formulas.* Prentice-Hall series in automatic computation. Prentice-Hall, Englewood Cliffs (N.J.), 1966.

[147] B. Sturmfels. On the number of real roots of a sparse polynomial system. To appear in the Proceedings of the workshop on '*Hamiltonian and Gradient Flows: Algorithms and Control*', Fields Institute, Waterloo, Ontario, March 1992.

[148] B. Sturmfels. Viro's theorem for complete intersections. To appear in *Annali della Scoula Normale di Pisa.*

[149] B. Sturmfels. On the Newton polytope of the resultant. *Journal of Algebraic Combinatorics*, 3:207–236, 1994.

[150] B. Sturmfels. *Gröbner Bases and Convex Polytopes*, volume 8 of *University Lecture Series.* AMS, Providence, R.I., 1996.

[151] W. Sweldens. *The construction and application of wavelets in numerical analysis.* PhD thesis, K.U.Leuven, 1994.

[152] L.W. Tsai and A.P. Morgan. Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods. *ASME Journal of Mechanics, Transmissions, and Automation in Design*, 107:189–200, 1985.

[153] L.G. Valiant. The complexity of computing the permanent. *Theoret. Comput. Sci.*, 8(2):189–201, 1979.

[154] B.L. van der Waerden. *Einführung in die algebraische Geometrie.* Springer–Verlag, Berlin, 1939.

[155] P. Van Hentenryck, D. McAllester, and D. Kapur. Solving polynomial systems using a branch and prune approach. Technical Report No. CS-95-01, Department of Computer Science, Brown University, 1995.

[156] P. Verlinden. *Cubature formulas and asymptotic expansions.* PhD thesis, K.U.Leuven, 1993.

[157] P. Verlinden, R. Cools, D. Roose, and A. Haegemans. The construction of cubature formulae for a family of integrals: a bifurcation problem. *Computing*, 40(4):337–346, 1988.

[158] J. Verschelde. Oplossen van stelsels veeltermvergelijkingen met behulp van continueringsmethodes. Bachelor's Thesis, K.U.Leuven, 1990.

[159] J. Verschelde. PHC and MVC: two programs for solving polynomial systems by homotopy continuation. Presented at the PoSSo Workshop on software, Paris, March 1-4, 1995.

[160] J. Verschelde, M. Beckers, and A. Haegemans. A new start system for solving deficient polynomial systems using continuation. *Appl. Math. Comput.*, 44(3):225–239, 1991.

[161] J. Verschelde and R. Cools. Nonlinear reduction for solving deficient polynomial systems by continuation methods. *Numer. Math.*, 63(2):263–282, 1992.

[162] J. Verschelde and R. Cools. An Ada workbench for homotopy continuation for solving polynomial systems. *The Ada Belgium Newsletter*, 2(1):23–40, 1993.

[163] J. Verschelde and R. Cools. Symbolic homotopy construction. *Applicable Algebra in Engineering, Communication and Computing*, 4(3):169–183, 1993.

[164] J. Verschelde and R. Cools. Symmetric homotopy construction. *J. Comput. Appl. Math.*, 50:575–592, 1994.

[165] J. Verschelde and K. Gatermann. Symmetric Newton polytopes for solving sparse polynomial systems. *Adv. Appl. Math.*, 16(1):95–127, 1995.

[166] J. Verschelde, K. Gatermann, and R. Cools. Mixed-volume computation by dynamic lifting applied to polynomial system solving. To appear in *Discrete Comput. Geom.*

[167] J. Verschelde and A. Haegemans. The *BQ*-Algorithm for the construction of an m-homogeneous start system. Report TW 147, Dept. of Computer Science, K.U.Leuven, 1991.

[168] J. Verschelde and A. Haegemans. The *GBQ*-Algorithm for constructing start systems of homotopies for polynomial systems. *SIAM J. Numer. Anal.*, 30(2):583–594, 1993.

[169] J. Verschelde and A. Haegemans. Homotopies for solving polynomial systems within a bounded domain. *Theoret. Comput. Sci.*, 133(1):165–185, 1994.

[170] J. Verschelde and A. Haegemans. A criterion for monomials influencing the root count. Report TW 237, Dept. of Computer Science, K.U.Leuven, 1996. Submitted for publication.

[171] J. Verschelde, P. Verlinden, and R. Cools. Homotopies exploiting Newton polytopes for solving sparse polynomial systems. *SIAM J. Numer. Anal.*, 31(3):915–930, 1994.

[172] M.N. Vrahatis. Algorithm 666 CHABIS: A mathematical software package for locating and evaluating roots of systems of nonlinear equations. *ACM Trans. Math. Softw.*, 14(4):330–336, 1988.

[173] M.N. Vrahatis. Solving systems of nonlinear equations using the nonzero value of the topological degree. *ACM Trans. Math. Softw.*, 14(4):312–329, 1988.

[174] C. Wampler and A. Morgan. Solving the 6R inverse position problem using a generic-case solution methodology. *Mech. Mach. Theory*, 26(1):91–106, 1991.

[175] C.W. Wampler. Bezout number calculations for multi-homogeneous polynomial systems. *Appl. Math. Comput.*, 51(2–3):143–157, 1992.

[176] C.W. Wampler. An efficient start system for multi-homogeneous polynomial continuation. *Numer. Math.*, 66(4):517–523, 1994.

[177] C.W. Wampler, A.P. Morgan, and A.J. Sommese. Numerical continuation methods for solving polynomial systems arising in kinematics. *ASME J. of Mechanical Design*, 112(1):59–68, 1990.

[178] C.W. Wampler, A.P. Morgan, and A.J. Sommese. Complete solution of the nine-point path synthesis problem for four-bar linkages. *ASME J. of Mechanical Design*, 114(1):153–159, 1992.

[179] L.T. Watson. Numerical linear algebra aspects of globally convergent homotopy methods. *SIAM Rev.*, 28(4):529–545, 1986.

[180] L.T. Watson. Globally convergent homotopy methods: a tutorial. *Appl. Math. Comput.*, 31(Spec. Issue):369–396, 1989.

[181] L.T. Watson, S.C. Billups, and A.P. Morgan. Algorithm 652: HOMPACK: a suite of codes for globally convergent homotopy algorithms. *ACM Trans. Math. Softw.*, 13(3):281–310, 1987.

[182] A.H. Wright. Finding all solutions to a system of polynomial equations. *Math. of Comp.*, 44(169):125–133, 1985.

[183] Y. Yomdin. Sard's theorem and its improved versions in numerical analysis. In E.L. Allgower and K. Georg, editors, *Computational Solution of Nonlinear Systems of Equations*, pages 701–706, Providence, R.I., 1990. AMS.

[184] T.J. Ypma. Historical development of the Newton-Raphson method. *SIAM Rev.*, 37(4):531–551, 1995.

[185] G.M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer–Verlag, New York, 1995.

List of Polynomial Systems

We give an overview of the polynomial systems we used as examples to illustrate and motivate the development of the algorithms. Some systems occur more than once when we have applied different methods to solve them.

List of Figures

List of Algorithms

List of Tables

Index

Contents