

# Parallel Implementation of Polyhedral Homotopy Methods

BY

YAN ZHUANG

B.S., Shanghai University, Shanghai , 1994  
M.S., University of Illinois at Chicago, 2004

THESIS

Submitted in partial fulfillment of the requirements  
for the degree of Doctor of Philosophy in Mathematics  
in the Graduate College of the  
University of Illinois at Chicago, 2007

Chicago, Illinois

Copyright by

Yan Zhuang

2007

This thesis is dedicated to my parents Jinmei Gan and Miaosheng Zhuang,

my husband Yuan Xu and my daughter Penny Yue Xu.

## ACKNOWLEDGMENTS

I would like to thank everyone who was involved in this work and give me every kind of help and encouragement that makes it possible to be finished. I would like to share my joy and excitement with them.

I would like to express my deep gratitude to my thesis advisor, Dr. Jan Verschelde, for giving me tremendous guide on research. Without his inspiration and support, I wouldn't have my achievement today. He had been a great research advisor and had provided me the perfect environment to learn and research.

I especially want to thank Professor Tien-Yien Li for his valuable suggestions and discussion at Notre Dame. I would like to thank Dr. Hai-Jun Su for his permission of using some pictures of his thesis.

I would also like to thank other committee members: Professor Rafail Abramov, Professor Gyorgy Turan and Professor Stephen Yau.

I want to give my thanks to the Department of Mathematics, Statistics, and Computer Science at UIC for its support and enjoyable environment for my study and research. I also want to thank the department and the WISE for contributions to my trip to Arizona Winter School 2006 on Computational and Algorithmic Aspects of Algebra and Arithmetic, 11-15 March 2006 at the University of Arizona and 113th Annual Meeting of AMS on January 5-8, 2007, in New Orleans, LA.

## ACKNOWLEDGMENTS (Continued)

This material is based upon work supported by the National Science Foundation under Grant No. 0105739 and Grant No. 0134611. Also I want to thank the NCSA for granting access to their IBM P690 (copper) through grant number CCR060019N.

Finally, I want to thank all my family members. I thank my parents and my brother for their love and constant support. I especially thank my husband, Yuan Xu, for his support, understanding and patience.

# TABLE OF CONTENTS

<u>CHAPTER</u>		<u>PAGE</u>
<b>1</b>	<b>INTRODUCTION</b> . . . . .	1
1.1	Motivation: Solve large “sparse” polynomial systems . . . . .	2
1.2	Homotopy Continuation Method . . . . .	5
1.3	Bernshtein Theorem . . . . .	8
1.4	Outline of Polyhedral Homotopies . . . . .	12
1.5	Contributions of this Thesis . . . . .	13
<b>2</b>	<b>PARALLEL IMPLEMENTATION OF THE POLYHEDRAL HOMOTOPY METHODS</b> . . . . .	15
2.1	Regular Triangulations . . . . .	16
2.2	Regular simple Mixed Subdivision . . . . .	18
2.3	Polyhedral Homotopies . . . . .	22
2.3.1	Coordinate Transformations . . . . .	23
2.3.2	The Cayley Trick . . . . .	26
2.4	Distribution of the Workload . . . . .	30
2.5	Dynamic Workload Distribution . . . . .	34
2.6	Computational Results . . . . .	35
2.6.1	Hardware and Software . . . . .	35
2.6.2	Experimental Results on the cyclic $n$ -roots problem . . . . .	38
2.6.3	Mechanism Design: the computation of reachable surfaces . . . . .	39
<b>3</b>	<b>PARALLEL HOMOTOPY ALGORITHMS TO SOLVE POLYNOMIAL SYSTEMS</b> . . . . .	46
3.1	Motivation: we want to solve large systems . . . . .	47
3.2	Jumpstarting Homotopies . . . . .	49
3.3	A Numerically Stable Solver for “Simplex” Systems . . . . .	53
3.4	Scanning Solution Files into Frequency Tables . . . . .	62
3.5	Towards High Performance Continuation ... . . . .	63
<b>4</b>	<b>SOFTWARE AND APPLICATIONS</b> . . . . .	65
4.1	Integrate MixedVol into PHCpack . . . . .	66
4.2	A C Interface to PHCpack . . . . .	68
4.3	Applications . . . . .	71
4.4	Conclusions and future work . . . . .	72
	<b>APPENDICES</b> . . . . .	73
	<b>Appendix A</b> . . . . .	74

TABLE OF CONTENTS (Continued)

<u>CHAPTER</u>	<u>PAGE</u>
Appendix B . . . . .	75
Appendix C . . . . .	79
CITED LITERATURE . . . . .	80
VITA . . . . .	88

## LIST OF TABLES

<u>TABLE</u>		<u>PAGE</u>
I	WALL TIME FOR START SYSTEMS TO SOLVE THE CYCLIC <i>N</i> -ROOTS PROBLEMS, USING 13 WORKERS, WITH STATIC LOAD DISTRIBUTION. . . . .	39
II	WALL TIME IN SECONDS FOR AN INCREASING NUMBER OF WORKERS TO SOLVE A START SYSTEM FOR THE CYCLIC 7-ROOTS PROBLEM. . . . .	40
III	WALL TIME FOR MECHANISM DESIGN PROBLEMS ON OUR CLUSTER AND ARGO. . . . .	45

## LIST OF FIGURES

<u>FIGURE</u>		<u>PAGE</u>
1	Newton polytopes $(P_1, P_2)$ of polynomial system $F$ . . . . .	4
2	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 $t \approx 1$ . . . . .	7
3	By a random choice of a complex $\gamma$ , singularities will not occur for all $t \in [0, 1]$ as on the left, but they may occur at the end, for $t = 1$ . . . . .	8
4	By a random choice of a complex $\gamma$ , divergence will not occur for all $t \in [0, 1]$ as on the left, but they may occur at the end, for $t = 1$ . . . . .	9
5	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 $P_1$ . . . . .	11
6	A mixed subdivision is regular if it is induced by a lifting . . . . .	22
7	The Cayley polytope of two polygons. The first polygon is placed at the vertex $(0,0,0)$ , the second polygon is placed at $(0,0,1)$ . . . . .	27
8	A triangulation of the Cayley polytope. The middle simplex is mixed, the other two simplices are unmixed. . . . .	28
9	A mixed subdivision induced by a triangulation of the Cayley polytope. . . . .	29
10	The example to show how the manager assigns the cells (with respective volumes listed) and #paths to the workers in the static workload distribution. . . . .	32
11	The example shows how the manager assigns the cells (with respective to volumes listed) and #paths to the workers in the dynamic workload distribution. . . . .	36
12	Speedup comparision for the cyclic 7-roots problem . . . . .	41

LIST OF FIGURES (Continued)

<u>FIGURE</u>		<u>PAGE</u>
13	The elliptic cylinder reachable by a PRS serial chain. Figure 4.4 of [66]. . . . .	42
14	The circular torus traced by the wrist center of a “right” RRS serial chain. Figure 4.7 of [66]. . . . .	44
15	The general torus reachable by the wrist center of an RRS serial chain. Figure 4.8 of [66]. . . . .	45

## LIST OF MATHEMATICS SYMBOLS

The symbols used in the thesis with their explanations are listed below. Different meanings for the same symbol occur when there is no confusion from context.

$N$	Number of equations
$n$	Number of unknowns
$\mathbb{C}^*$	$\mathbb{C} \setminus \{0\}$
$\mathbf{a}$	$\mathbf{a} = (a_1, a_2, \dots, a_n) \in \mathbb{N}^n$
$\mathbf{x}$	$\mathbf{x} = (x_1, x_2, \dots, x_n)$
$\mathbf{x}^{\mathbf{a}}$	$\mathbf{x}^{\mathbf{a}} = x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n}$
$F(\mathbf{x})$	Target polynomial system
$G(\mathbf{x})$	Generic start polynomial system
$A = \text{supp}(f)$	support of sparse polynomial $f$
$P = \text{conv}(A)$	Newton polytope of $f$
$\mathcal{A}$	$\mathcal{A} = (A_1, A_2, \dots, A_n)$ , support of $F$
$\mathcal{P}$	$\mathcal{P} = (P_1, P_2, \dots, P_n)$ , Newton polytope of $F$
$\text{vol}(P)$	volume of $P$
$V(\mathcal{P})$	mixed volume of $\mathcal{P}$
$\langle \cdot, \cdot \rangle$	standard inner product of polytope $P$

## LIST OF MATHEMATICS SYMBOLS (Continued)

$\mathbf{v}$	$\mathbf{v} = [v_1, v_2, \dots, v_n]^T$ , inner or outer normal
$\partial_{\mathbf{v}}A$	spans face of polytope
$\partial_{\mathbf{v}}P$	$\partial_{\mathbf{v}}P = \text{conv}(\partial_{\mathbf{v}}A)$ , face of polytope $P$
$\partial_{\mathbf{v}}f$	face polynomial
$\text{Vol}(A)$	Mixed volume of support $A$
$M$	Unimodular matrix, $\det(M) = \pm 1$
$U$	Upper triangular matrix
$\mathbf{x} = \mathbf{z}^M$	unimodular transformation of monomials
$S$	$S = \{C_1, C_2, \dots, C_n\}$ , a subdivision of $A$
$C$	$C \subset A, C \in S$ , a cell in a subdivision
$\omega$	lifting function
$\hat{\mathbf{a}}$	lifted vector, $a_{n+1} = \omega(\mathbf{a})$
$\hat{A}$	lifted support
$\hat{P}$	lifted polytope, $\hat{P} = \text{conv}(\hat{A})$
$S_\omega$	regular subdivision induced by $\omega$
$\hat{S}_\omega$	collection of lifted cells in $S_\omega$
$C_{\mathbf{v}}$	cell characterized by inner normal $\mathbf{v}$ , $\hat{C}_{\mathbf{v}} = \partial_{\mathbf{v}}\hat{A}$
$\hat{C}_{\mathbf{v}}$	lifted cell with inner normal $\mathbf{v}$

## LIST OF MATHEMATICS SYMBOLS (Continued)

$\Delta$	a triangulation of $A$
$\Delta_\omega$	a regular triangulation of $A$ , induced by $\omega$
$\hat{\mathcal{A}}$	$\hat{\mathcal{A}} = (\hat{A}_1, \hat{A}_2, \dots, \hat{A}_n)$ , lifted supports of $F$
$\hat{\mathcal{P}}$	$\hat{\mathcal{P}} = (\hat{P}_1, \hat{P}_2, \dots, \hat{P}_n)$ , lifted Newton polytope of $F$

## SUMMARY

Homotopy continuation methods to compute numerical approximations to all isolated solutions of a polynomial system are known as “pleasingly parallel”, i.e.: because of their low communication overhead, these methods scale very well for a large number of processors. Because so many important problems remain unsolved mainly due to their intrinsic computational complexity, it would be embarrassing not to develop parallel implementations of polynomial homotopy continuation methods.

“Parallel PHCpack”, is a project which started a couple of years ago developed by my advisor Jan Verschelde and his student Yusong Wang (parallel pieri homotopy), and which continues with Anton Leykin (parallel irreducible decomposition).

The focus of this thesis is the development of “parallel PHCpack”. For sparse polynomial systems, polyhedral methods give efficient homotopy algorithms. The polyhedral homotopy methods run in three stages:

1. compute the mixed volume;
2. solve a random coefficient start system;
3. track solution paths to solve the target system.

In this thesis, we parallelize the second stage in PHCpack (ACM TOMS Algorithm 795), a publicly available software package for solving polynomial system with homotopy methods.

## SUMMARY (Continued)

We use a static workload distribution algorithm and achieve a good speedup on the cyclic  $n$ -roots benchmark systems. Dynamic workload balancing leads to reduced wall times on large polynomial systems which arise in mechanism design.

# CHAPTER 1

## INTRODUCTION

Polynomial systems occur in a wide variety of application areas in science and engineering, see the case studies in [62, Chapter 9]. During the last two decades, homotopy methods have been proven reliable and efficient numerical algorithms for computing approximations to all isolated solutions of polynomial systems. In this chapter, we first introduce the history and related software of the homotopy continuation method, then we will define the problem and provide a motivation for our work. In Section 1.3, we will explain the Bernshtein Theorem. We will give the outline of the polyhedral homotopies in Section 1.4.

The history of homotopy continuation for polynomial systems can be roughly divided into two eras, each spanning about one decade. The first decade was focussed on applying Bézout's theorem for counting the solutions. Milestone publications are the introductory paper [41], the book [46], and the survey [79]. Publicly available software packages are CONSOL [46] and HOMPACK [81], recently upgraded to Fortran 90 [82]. During the last ten years, root-counting methods have developed to exploit the structure of a polynomial system. The novel methods are of a symbolic-numeric nature [16]. Progress in homotopy continuation for polynomial systems [42] benefited from the interaction between combinatorics, algebraic geometry and applied mathematics [65]. The polyhedral methods have brought homotopy continuation into the literature on computational algebraic geometry [13]. Recently, optimal homotopies were presented for computing linear subspace intersections in enumerative geometry, see [28] and [65].

Now we will mention related software dedicated to solving polynomial systems by homotopy continuation, four different packages that are publicly available are briefly mentioned. We also refer to a program for computing mixed volumes.

HOMPACK [52],[81] and [46] are written in Fortran 77. HOMPACK is a general package for homotopy continuation with a polynomial driver. It has been parallelized [2], [26] and extended with an end game [63]. A Fortran 90 version appeared recently [82]. The package POLSYS\_PLP [83] for constructing partitioned linear-product start systems is intended to be used in conjunction with HOMPACK90. Morgan, Sommese and Wampler [49], [50] and [51] developed technique to handle end-point singularities.

The computation of mixed volumes is a crucial step in the resolution of sparse polynomial systems. We can use PHCpack [71], or MixedVol [22], or PHoM [25] to compute mixed volumes.

### 1.1 Motivation: Solve large “sparse” polynomial systems

Almost all polynomial systems occurring in applications are *sparse*, i.e.: only relatively few monomials appear with nonzero coefficients.

**Definition 1.1.1** A *sparse polynomial*  $f(\mathbf{x}) \in \mathbb{C}[\mathbf{x}]$ , in  $n$  variables  $\mathbf{x} = (x_1, x_2, \dots, x_n)$  is written as

$$f(\mathbf{x}) = \sum_{\mathbf{a} \in A} c_{\mathbf{a}} \mathbf{x}^{\mathbf{a}}, \quad \text{with } c_{\mathbf{a}} \in \mathbb{C}^*, \quad \mathbb{C}^* = \mathbb{C} \setminus \{0\}, \quad \text{and } \mathbf{x}^{\mathbf{a}} = x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n} \quad (1.1)$$

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

**Definition 1.1.2** A sparse polynomial system  $F(\mathbf{x}) = \mathbf{0}$  is defined by a tuple of sparse polynomials,

$$F = (f_1, f_2, \dots, f_n), \quad f_k(\mathbf{x}) \in \mathbb{C}[\mathbf{x}], \quad k = 1, 2, \dots, n$$

The tuples  $\mathcal{A}$  and  $\mathcal{P}$  collect respectively the supports and Newton polytopes:

$$\begin{aligned} \mathcal{A} &= (A_1, A_2, \dots, A_n), A_j = \text{supp}(f_j) \\ \mathcal{P} &= (P_1, P_2, \dots, P_n), P_j = \text{conv}(A_j) \end{aligned}, \quad j = 1, 2, \dots, n. \quad (1.2)$$

**Example 1.1.1** A small example of a sparse polynomial system

$$\begin{aligned} F &= (f_1, f_2) & \mathcal{A} &= (A_1, A_2) \\ &= \begin{cases} x_1^3 x_2 + x_1 x_2^2 + 1 = 0 \\ x_1^4 + x_1 x_2 + 1 = 0 \end{cases} & A_1 &= \{(3, 1), (1, 2), (0, 0)\} \\ & & A_2 &= \{(4, 0), (1, 1), (0, 0)\} \end{aligned} \quad (1.3)$$

The polytopes of Example 1.1.1 is shown in Figure 1. For sparse systems, homotopies based on the degrees typically track many paths diverging to infinity. For the above example, the *total degree* is  $D=4*4=16$ , so based on the degrees, we will track 16 paths, however polyhedral homotopies will only track 8 paths, which means that another 8 paths will diverge to infinity.

For another example, in a benchmark system from mechanism design [67] [70], a degree-based homotopy used in [69] by the POLSYS\_GLP extension [68] of HOMPACK [81] [82] [83] leads to 9,216 paths, whereas polyhedral homotopies track the optimal number of 1,024 paths. The 1,024 for this system is the *mixed volume*. We view the mixed volume as the number of

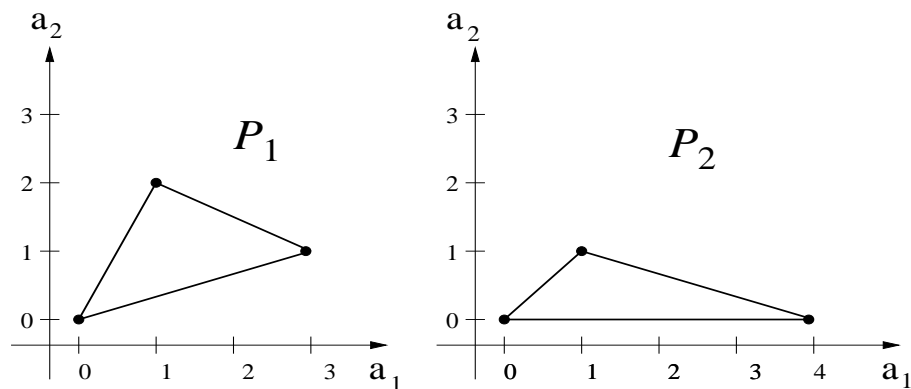


Figure 1. Newton polytopes  $(P_1, P_2)$  of polynomial system  $F$ .

isolated solutions of a system with randomly chosen complex coefficients. See Chapter 1.3 for details.

In this thesis, we will apply polyhedral homotopies to large “sparse” polynomial systems arising in mechanism design. These systems have a special structure, they are not so sparse, but their mixed volume is still less than degree bounds. We say that system  $F$  is *large* if the homotopy that we use to solve it requires more than 100,000 solutions to track. Although large does not always automatically imply “difficult”, numerical problems are more likely to occur. This thesis is concerned with three issues:

1. Workload Distribution:

The polynomial systems that we want to solve are all large systems with more than 100,000 paths to track, so how to track the paths more efficiently is very important. Homotopy methods to solve polynomial systems are “pleasingly parallel” because the solution paths

defined by the homotopy can be tracked independently. When we try to parallelize the polyhedral homotopy method, how to distribute the workload across processors in order to obtain the highest-possible execution speed becomes very important.

## 2. Numerical Stabilities:

When dimensions and degrees of the polynomial system grows, numerical stabilities may occur. We want to discover a numerically stable solver which will compute the roots and avoid numerical overflow or underflow.

## 3. Jumpstarting: Memory Use

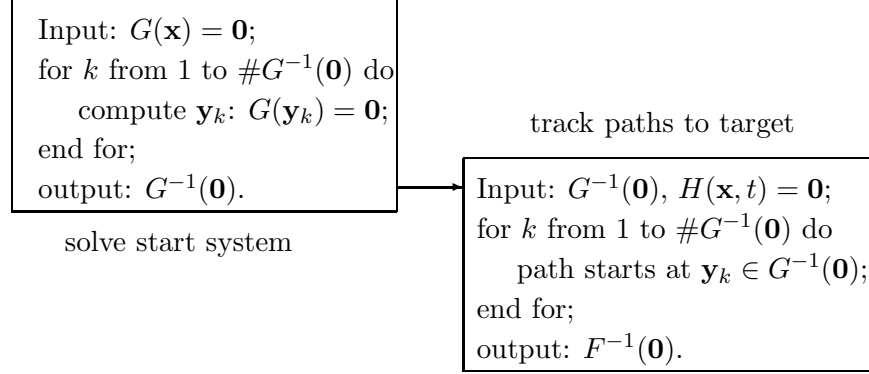
When we use polyhedral homotopies to solve a polynomial system, we will compute start solutions of the target system. Since we solve large polynomial systems, we will have more than 100,000 start solutions. For efficiency, it is undesirable to keep all solutions in the main memory.

## 1.2 Homotopy Continuation Method

Homotopy continuation methods are globally convergent and exhaustive solvers for computing numerical approximations to all isolated solutions of polynomial systems, except perhaps at the very end of the paths if the system to be solved has singular solutions.

**Definition 1.2.1** A homotopy  $H(\mathbf{x}, t) = 0$  is a family of systems

$$H(\mathbf{x}, t) = \gamma(1 - t)G(\mathbf{x}) + tF(\mathbf{x}) = \mathbf{0}, \quad \gamma \in \mathbb{C}, \quad t \in [0, 1]. \quad (1.4)$$



The homotopy is parameterized by a *continuation parameter*, denoted by  $t$ . The constant  $\gamma$  is a random constant number, also called the *accessibility constant*. The homotopy contains the *start system*  $G(\mathbf{x}) = H(\mathbf{x}, 0) = \mathbf{0}$ , its solutions are called *start solutions*, and the *target system*  $F(\mathbf{x}) = H(\mathbf{x}, 1) = \mathbf{0}$ , whose solutions are desired and *target solutions*.

The homotopy given in Equation 1.4 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. For example, we consider the problem of finding surfaces that contain a set of points generated by a displaced rigid body. The polynomial system defined by these reachable surfaces becomes a *natural-parameter homotopy*. 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.

**Definition 1.2.2** A *good homotopy*  $H(\mathbf{x}, t)$  should possess the following properties ( criteria taken from [42], [43] ):

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

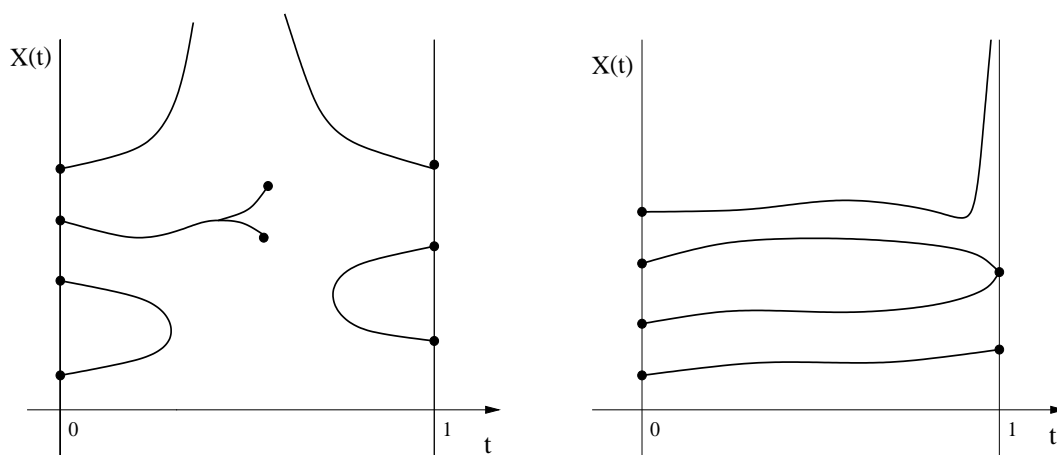


Figure 2. On the left, a bad behavior of the solution paths is pictured: paths that turn back, bifurcate, stop or diverge to infinity for some  $t, |t| < 1$ . A typical behavior is drawn on the right: paths can only diverge at the end, when  $t \approx 1$ .

2. The solution set of  $H(\mathbf{x}, t) = \mathbf{0}$  for all  $t : 0 \leq |t| \leq 1$ , contains a finite number of smooth paths, each can be parameterized by  $t$ .
3. Every isolated solution of  $H(\mathbf{x}, 1) = \mathbf{0}$  is reached by some path originated at a solution of  $H(\mathbf{x}, t) = \mathbf{0}$  for  $t = 0$ . If  $m$  is the multiplicity of an isolated solution  $x^*$ , then there are exactly  $m$  paths converging to  $x^*$ .

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

In Figure 2, the behavior of the solution paths is pictured, with on the left a bad and on the right a good homotopy.

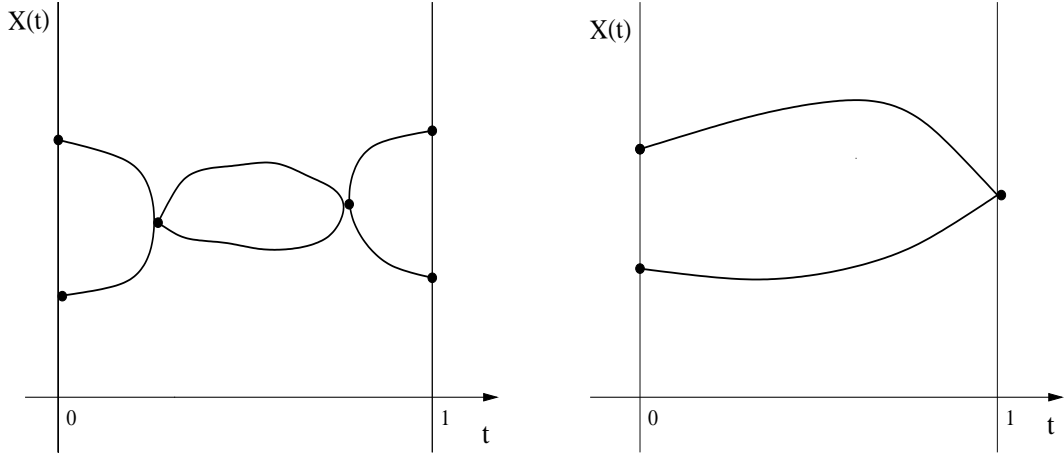


Figure 3. By a random choice of a complex  $\gamma$ , singularities will not occur for all  $t \in [0, 1]$  as on the left, but they may occur at the end, for  $t = 1$ .

The random complex choice of  $\gamma$  will ensure that all the roots miss the interval  $[0, 1)$ . A schematic (as in [46]) illustrating what cannot and what can happen is in Figure 3.

The same random constant  $\gamma$  ensures that all paths stay bounded for all  $t \in [0, 1)$ . By this we mean that no path diverges to infinity for some  $t \in [0, 1)$ . Equivalently, for all  $t \in [0, 1)$ , the system  $H(\mathbf{x}, t) = \mathbf{0}$  has no solutions at infinity ( see Figure 4 )

### 1.3 Bernshtein Theorem

Given a sparse polynomial  $F = (f_1, f_2, \dots, f_n)$  and the support  $\mathcal{A} = (A_1, A_2, \dots, A_n)$ , its convex hull,  $\mathcal{P} = (P_1, P_2, \dots, P_n)$ ,  $P_j = \text{conv}(A_j)$  are the *Newton polytopes* of  $F$ . We denote the volume of its convex hull by  $\text{Vol}(\mathcal{A})$ . Kushnirenko showed in [34] that a system  $F(\mathbf{x}) = \mathbf{0}$  whose polynomials all share the same support  $\mathcal{A}$  can have no more than  $\text{Vol}(\mathcal{A})$  isolated solutions

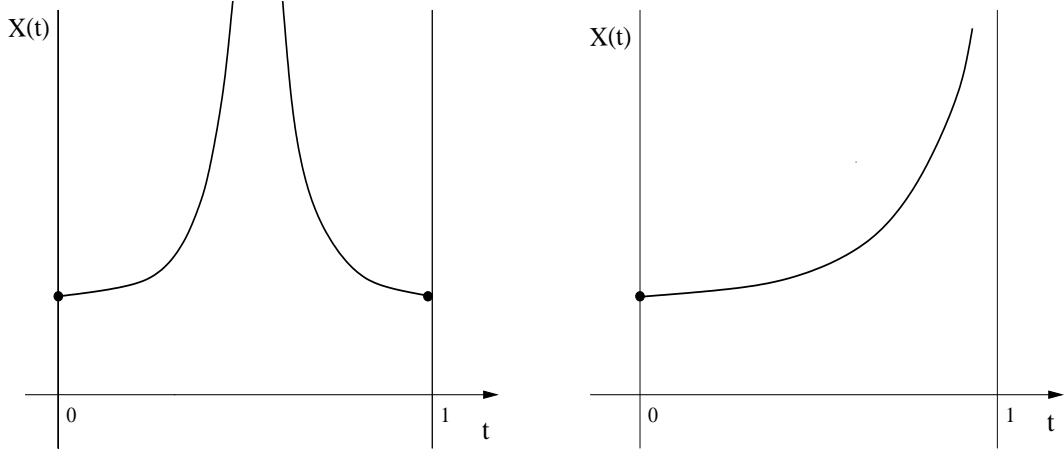


Figure 4. By a random choice of a complex  $\gamma$ , divergence will not occur for all  $t \in [0, 1]$  as on the left, but they may occur at the end, for  $t = 1$ .

in  $(\mathbb{C}^*)^n$ . Moreover, if the coefficients  $c_{\mathbf{a}}$  are generic, then  $\text{Vol}(A)$  is an exact root count. Kushnirenko's theorem was generalized to general polynomial systems by Bernshtein [3] and Khovanskii [32].

In this section, we will give the basic definitions of the mixed volumes. We will also give an example to explain how to compute the mixed volumes.

**Definition 1.3.1** The *mixed volume*  $V(\mathcal{P})$  of an  $n$ -tuple of polytopes  $\mathcal{P}$  is

$$V(\mathcal{P}) := \sum_{I \subset \{1, 2, \dots, n\}} (-1)^{n-\#I} \text{vol}\left(\sum_{i \in I} P_i\right) \quad (1.5)$$

where  $\text{vol}(P)$  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  $V(P, P, \dots, P) = n! \text{vol}(P)$ .

**Theorem 1.3.1** (*Bernshtein's theorem A*) *let  $F$  be a tuple of  $n$  sparse polynomials with Newton polytopes  $\mathcal{P}$ . Then the number of isolated solutions of  $F(\mathbf{x}) = \mathbf{0}$  in  $(\mathbb{C}^*)^n$  is bounded by the mixed volume  $V(\mathcal{P})$ . For almost all choices of the coefficients the system  $F(\mathbf{x}) = \mathbf{0}$  has exactly as many roots in  $(\mathbb{C}^*)^n$  as  $V(\mathcal{P})$ .*

**Example 1.3.1** Consider the following polynomial system

$$F(\mathbf{x}) = \begin{cases} c_{10} + c_{11}x_1x_2^2 + c_{12}x_1^3x_2 = 0 \\ c_{20} + c_{21}x_1x_2 + c_{22}x_1^4 = 0 \end{cases} \quad (1.6)$$

Figure 5 shows that the Newton polytopes of the system and their Minkowski sum.

Applying Equation 1.5, the BKK <sup>1</sup> bound can be computed as

$$V(P_1, P_2) = \text{vol}(P_1 + P_2) - \text{vol}(P_1) - \text{vol}(P_2) = 8 \quad (1.7)$$

From Theorem 1.3.1, we know that for almost all choices of the coefficients  $c_{ij} \in \mathbb{C}^*$ , the system  $F(\mathbf{x}) = \mathbf{0}$  has eight isolated solutions in  $(\mathbb{C}^*)^2$ .

Now we will give some definitions about face, since in Chapter 2, we will use these definitions.

---

<sup>1</sup>The mixed volume was nicknamed in [10] as the BKK bound to honor Bernshtein [3], Kushnirenko [34] and Khovanskii [32].

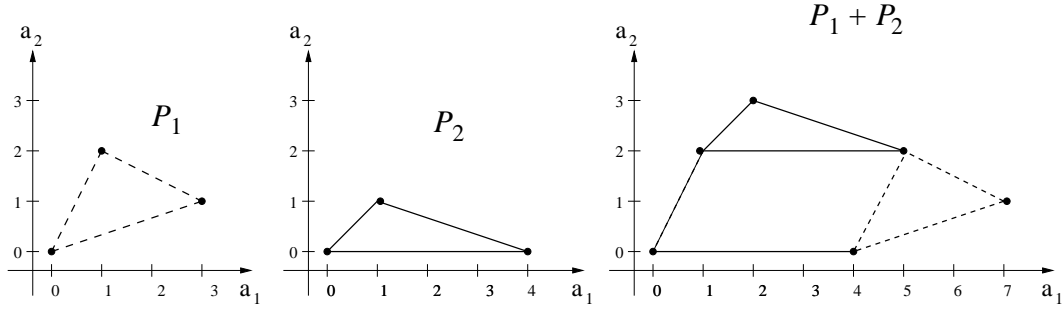


Figure 5. Newton polytopes  $(P_1, P_2)$  of  $F$ , with their Minkowski sum  $P_1 + P_2$ , constructed by placing  $P_2$  at the vertices of  $P_1$ .

**Definition 1.3.2** 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  $\partial_{\mathbf{v}}P = \text{conv}(\partial_{\mathbf{v}}A)$ .

$\mathbf{v}$  is called an *outer normal* to  $P$ .  $-\mathbf{v}$  is called an *inner normal* to  $P$  and we will substitute max by min in the definition of  $\partial_{\mathbf{v}}P$ .

**Definition 1.3.3** 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  $\mathbf{a}_0, \mathbf{a}_1, \dots, \mathbf{a}_d$  such that  $\mathbf{a}_0 - \mathbf{a}_j$ ,  $j = 1, 2, \dots, d$ , 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 1.3.4** Consider a sparse polynomial  $f(\mathbf{x})$  with support set  $A$ , we say  $\partial_{\mathbf{v}}f$  is a *face polynomial*.

$$f(\mathbf{x}) = \sum_{\mathbf{a} \in A} c_{\mathbf{a}} \mathbf{x}^{\mathbf{a}} \quad \text{and} \quad \partial_{\mathbf{v}}f(\mathbf{x}) = \sum_{\mathbf{a} \in \partial_{\mathbf{v}}A} c_{\mathbf{a}} \mathbf{x}^{\mathbf{a}} \quad (1.8)$$

**Definition 1.3.5** Given a sparse polynomial system  $F = (f_1, f_2, \dots, f_n)$  with support  $\mathcal{A} = (A_1, A_2, \dots, A_n)$ . Then *the face system*  $\partial_{\mathbf{v}}F$  of  $F$  is defined as the subsystem

$$\partial_{\mathbf{v}}F = (\partial_{\mathbf{v}}f_1, \partial_{\mathbf{v}}f_2, \dots, \partial_{\mathbf{v}}f_n) \quad (1.9)$$

$\partial_{\mathbf{v}}f_i$  is the face polynomial of  $f_i$ , for  $i = 1, 2, \dots, n$ .

#### 1.4 Outline of Polyhedral Homotopies

To solve a polynomial system  $F(\mathbf{x}) = \mathbf{0}$  using polyhedral homotopies, we distinguish three stages.

1. In the first stage, we will compute the mixed volume of the Newton polytopes  $\mathcal{P}$  spanned by the supports of  $F$ . When we compute the mixed volume, we ignore the particular coefficients of the polynomial system. We save the results in the mixed-cell configuration file, for details about this file please see Chapter 2.

2. In the second stage, we apply *polyhedral homotopies* to solve a system  $G(\mathbf{x}) = \mathbf{0}$  with the same sparse structure as the given system, but with random coefficients, using the results of the first stage.
3. In the third stage and final stage, we apply a plain linear homotopy to solve the given system  $F(\mathbf{x}) = \mathbf{0}$  using coefficient-parameter polynomial continuation [48] (related to the cheater’s homotopy [45]).

$$H(\mathbf{x}, t) = (1 - t)G(\mathbf{x}) + tF(\mathbf{x}) = \mathbf{0}, \quad \text{for } t \text{ from } 0 \text{ to } 1.$$

In Chapter 2, we will primarily focus on the second stage, and consider a mixed-cell configuration as given. These mixed-cell configurations may be computed using PHCpack [71], or MixedVol [22], or PHoM [25]. A parallel implementation of PHoM is described in [24]. While the third stage involves as many paths as the second stage, the computational cost can increase significantly because the polynomial system at the end of the homotopy is no longer generic. Conditions on genericity are given in [56].

## 1.5 Contributions of this Thesis

The contribution of this thesis is twofold. First – as announced in [77] – we parallelize the second stage in PHCpack. In the next chapter we will describe our two workload distribution algorithms in detail, one is static workload distribution and the other is dynamic workload distribution. Using dynamic algorithm we can solve large polynomial systems which arise in mechanism design. Second – as announced in [40] – we discovered a numerically stable solver

for the sparsest class of polynomial systems and propose to *jumpstart* homotopies. In Chapter 4, we will explain in detail the numerically stable solver which computes the magnitudes of the roots separately, avoiding numerical overflow or underflow. Also we will explain how to use *jumpstart* homotopy to avoid the storage of all start solutions in the main memory. In the last chapter, we will describe the numerical results.

## CHAPTER 2

# PARALLEL IMPLEMENTATION OF THE POLYHEDRAL HOMOTOPY METHODS

Since the solution paths defined by the homotopy can be tracked independently, therefore using homotopy continuation methods to solve polynomial systems (recently surveyed in [43] and [62]) are well suited for parallel implementation, as described in [2], [11; 12], [26], [53], and [54].

To solve a polynomial system using polyhedral homotopies, we distinguish three stages, as describe it in Chapter 1. In this chapter, we primarily focus on the second stage, and consider a mixed-cell configuration as given. We continue the development of parallel implementations of homotopy algorithms in PHCpack [71] – started in [76] with Pieri homotopies, followed by parallel decomposition methods in [37; 39].

To make this chapter self-contained, we will introduce some definitions first and give some examples. Then we will discuss different workload distributions. Our first parallel implementation of polyhedral homotopies uses a static workload distribution. This static distribution (as we experienced in [76]) is favorable when all solution paths require about the same computational cost, as could be expected from polyhedral homotopies solving generic polynomial systems. However, dynamic load balancing significantly improves the wall time for large mechanism design problems.

## 2.1 Regular Triangulations

In this chapter, for convenience we will work with *inner normals* and define the face  $\partial_{\mathbf{v}}P$  of the polytope  $P$  by  $\partial_{\mathbf{v}}P = \text{conv}(\partial_{\mathbf{v}}A)$ , 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 2.1.1** The *lower hull* of a polytope  $P$  consists of all facets  $\partial_{\mathbf{v}}P$  with  $v_n > 0$ .

**Definition 2.1.2** Given a support  $A$ , a *subdivision*  $S$  of  $A$  consists of a collection of cells  $S = \{C_1, C_2, \dots, C_m\}$ , with  $C_k \subset A, \forall k$ , satisfying

1.  $\dim(C_k) = n$
2.  $\text{conv}(C_l) \cap \text{conv}(C_k)$  is a proper face of both  $\text{conv}(C_l)$  and  $\text{conv}(C_k), l \neq k$
3.  $\cup_{k=1}^m \text{conv}(C_k) = \text{conv}(A)$

**Definition 2.1.3** A *lifting function*  $\omega$  lifts every point of a set  $A$ ,  $\omega : A \rightarrow \mathbb{R} : \mathbf{a} \mapsto \omega(\mathbf{a})$ . This yields the *lifted support*  $\hat{A} = \omega(A) = \{ \hat{\mathbf{a}} = (\mathbf{a}, \omega(\mathbf{a})) \mid \mathbf{a} \in A \}$ . The *lifted polytope* is denoted by  $\hat{P} = \text{conv}(\hat{A})$ .

For every lifting function a subdivision is induced by associating the cells with the projected facets  $\partial_{\mathbf{v}}\hat{P}$  of the lower hull of  $\hat{P}$ ,  $P = \text{conv}(A)$ . Every facet  $\hat{C} = \partial_{\mathbf{v}}\hat{A}$  is characterized by an *inner normal*  $\mathbf{v} \in \mathbb{R}^{n+1}$ ,  $v_{n+1} > 0$ , satisfying equalities  $\langle \hat{\mathbf{c}}_i, \mathbf{v} \rangle = \langle \hat{\mathbf{c}}_j, \mathbf{v} \rangle$ , for  $\hat{\mathbf{c}}_i, \hat{\mathbf{c}}_j \in \hat{C}$  and inequalities  $\langle \hat{\mathbf{a}}, \mathbf{v} \rangle > \langle \hat{\mathbf{c}}, \mathbf{v} \rangle$ , for  $\hat{\mathbf{c}} \in \hat{C}$  and  $\hat{\mathbf{a}} \in \hat{A} \setminus \hat{C}$ , where  $\langle \cdot, \cdot \rangle$  is the usual inner product.

**Definition 2.1.4** A subdivision  $S = \{C_1, C_2, \dots, C_n\}$  of  $A$  is called a *regular subdivision* if there exists a lifting function  $\omega$  which induces  $S$ . It will be denoted by  $S_{\omega}$ . The collection of

lifted cells  $\hat{C}$  will be denoted by  $\hat{S}$ . Note that in [29] a regular subdivision is called a *coherent* subdivision.

Given a subdivision  $S$  of a polytope  $P$  and support  $A$ , its volume can be computed by

$$\text{Vol}(A) = \sum_{C \in S} \text{Vol}(C) \quad (2.1)$$

The computation of the volumes of the cells is straightforward for special subdivisions.

**Definition 2.1.5** A subdivision  $S$  is called a *triangulation*, denoted by  $\Delta = \{C_1, C_2, \dots, C_m\}$  if  $\#C_j = n + 1, \forall j$ ,  $\text{conv}(C_j)$  is an  $n$ -dimensional simplex. A triangulation  $\Delta$  is said to be a *regular triangulation* if it can be induced by a lifting function  $\omega$ .

We denote a regular triangulation as  $\Delta_\omega$ . Note that not every triangulation is regular, see [35] for an example.

Each cell  $C$  of a triangulation spans a simplex, so we can denote  $C = \{\mathbf{c}_0, \mathbf{c}_1, \dots, \mathbf{c}_n\}$  and

$$\text{vol}(C) = \frac{1}{n!} |\det(\mathbf{c}_1 - \mathbf{c}_0, \dots, \mathbf{c}_n - \mathbf{c}_0)| \quad (2.2)$$

Therefore, a regular triangulation  $\Delta_\omega$  is a triangulation — i.e.: a collection of nonoverlapping simplices so that  $\text{Vol}(A) = \sum_{C \in \Delta_\omega} \text{Vol}(C)$  — where every cell  $C \in \Delta_\omega$  corresponds to a facet  $\hat{C}$  of the lower hull of the lifted support  $\hat{A} = \omega(A) = \{ \hat{\mathbf{a}} = (\mathbf{a}, \omega(\mathbf{a})) \mid \mathbf{a} \in A \}$ .

## 2.2 Regular simple Mixed Subdivision

Theorem 2.2.1 states a fundamental result from Minkowski, see [9]. This theorem will lead us to mixed subdivision which is needed to compute mixed volumes.

**Theorem 2.2.1** (Minkowski) *Let  $\mathcal{P} = (P_1, P_2, \dots, P_n)$  be a tuple of polytopes. Consider the positive linear combination of the polytopes in  $\mathcal{P}$  :  $P = \lambda_1 P_1 + \lambda_2 P_2 + \dots + \lambda_n P_n$ ,  $\lambda_i \geq 0, \forall i$ . The volume  $\text{vol}(P)$  of  $P$  is a homogeneous polynomial of degree  $n$  with as unknowns the  $\lambda_i$ 's and as coefficients the mixed volumes of all possible subsets of  $\mathcal{P}$ . More precisely,*

$$n! \text{vol}(\lambda_1 P_1 + \lambda_2 P_2 + \dots + \lambda_n P_n) = \sum_{i_1=1}^n \sum_{i_2=1}^n \dots \sum_{i_n=1}^n V(P_{i_1}, P_{i_2}, \dots, P_{i_n}) \lambda_{i_1} \lambda_{i_2} \dots \lambda_{i_n} \quad (2.3)$$

The mixed volume  $V(\mathcal{P})$  of  $\mathcal{P}$  is given by the coefficient of the monomial  $\lambda_1 \lambda_2 \dots \lambda_n$  in the expansion formula (Equation 2.3) of  $n! \text{vol}(P)$ .

We call the expansion formula (Equation 2.3) the *Minkowski polynomial* of  $\mathcal{P}$ . This polynomial can be computed efficiently by the Cayley trick, which we will explain later.

It is computationally advantageous to exploit the fact that several polynomials have the same support. Denote the number of different support  $r$ . Moreover, if all supports are equal, then we automatically arrive at the well-known case of one polytope.

**Definition 2.2.1** let  $\mathcal{A}$  be a tuple of point sets with corresponding tuple of polytopes  $\mathcal{P}$ . Denoted by  $\mathbf{k} = (k_1, k_2, \dots, k_r)$ , with  $k_i$  the number of times the same  $i$ th support occurs in  $\mathcal{A}$ . Assume that the sets in  $\mathcal{A}$  are ordered in the following way

$$\begin{aligned} \mathcal{A} &= ( \underbrace{A_1, \dots, A_1}_{k_1}, \underbrace{A_2, \dots, A_2}_{k_2}, \dots, \underbrace{A_r, \dots, A_r}_{k_r} ) \\ \mathcal{P} &= ( \underbrace{P_1, \dots, P_1}_{k_1}, \underbrace{P_2, \dots, P_2}_{k_2}, \dots, \underbrace{P_r, \dots, P_r}_{k_r} ) \end{aligned} \quad (2.4)$$

with  $\sum_{i=1}^r k_i = n$

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  $\mathcal{C} = (C_1, C_2, \dots, C_n)$ ,  $C_i \subset A_i$ , be a cell. The volume of  $C$  is written as  $\text{vol}(C) = \text{vol}(\text{conv}C)$

where the following conventions are used:

$$\begin{aligned} \text{conv}(C) &= \text{conv}(C_1 + C_2 + \dots + C_r) \subset \mathbb{R}^n \\ \text{type}(C) &= (\dim(C_1), \dim(C_2), \dots, \dim(C_r)) \in \mathbb{N}^r \end{aligned} \quad (2.5)$$

With  $\mathcal{C}$  we denote the cell as tuple of subsets of the supports, while  $C$  stands for the sum of all subsets in  $\mathcal{C}$ . We also denote  $A := \sum_{i=1}^r A_i$  and  $P := \sum_{i=1}^r P_i$ .

Given a tuple of points sets, based on a subdivision of the Minkowski sum, Huber and Sturmfels have extended Definition 2.1.2 in the following way, (see [29]):

**Definition 2.2.2** Assume  $\mathcal{A} = (A_1, A_2, \dots, A_r)$ . A *subdivision* of  $\mathcal{A}$  is a collection  $S = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_m\}$  of  $m$  cells  $\mathcal{C}_j = (C_{j1}, C_{j2}, \dots, C_{jr})$ ,  $C_{ji} \subset A_i$ , satisfying

1.  $\dim(C_j)=n$ , for  $j = 1, \dots, m$ ;
2.  $\text{conv}(C_j) \cap \text{conv}(C_k)$  is a proper common face of  $\text{conv}(C_j)$  and  $\text{conv}(C_k)$ ,  $j \neq k$ ;
3.  $\cup_{j=1}^m \text{conv}(C_j) = \text{conv}(A)$ .

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  $C_j \in S$ .

**Definition 2.2.3** 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  $\text{type}(C) = (k_1, k_2, \dots, k_r)$  are mixed, see [29]. For the computation of the mixed volume, it is sufficient that the collection of the mixed cells is fine mixed.

**Definition 2.2.4** A mixed subdivision is called a *simple mixed subdivision* if for all mixed cells

$$C_j : \sum_{i=1}^r (\#(C_{ji}) - 1) = n.$$

**Definition 2.2.5** A subdivision  $S$  of a tuple  $\mathcal{A} = (A_1, A_2, \dots, A_r)$  is called *regular* if there exists a tuple  $\omega$  of lifting functions,  $\omega = (\omega_1, \omega_2, \dots, \omega_r)$ , 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  $\widehat{\mathcal{A}} = (\widehat{A}_1, \widehat{A}_2, \dots, \widehat{A}_r)$  and the corresponding polytopes by  $\widehat{\mathcal{P}} = (\widehat{P}_1, \widehat{P}_2, \dots, \widehat{P}_r)$ . As before, a regular subdivision induced by a lifting  $\omega$

is denoted by  $S_\omega$  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 facet spanned by mixed cells are called *mixed facets*.

Given a mixed subdivision, it is sufficient to consider only the mixed cells, as shown in [29], to compute the mixed volume. Because we take the type of mixture into account, we extend the notation of  $V$  and  $\text{vol}$  with the additional argument  $\mathbf{k} \in \mathbb{N}^r$ .

$$V(\mathcal{P}, \mathbf{k}) = \sum_{\substack{\mathcal{C} \in S_\omega \\ \text{type}(\mathcal{C}) = \mathbf{k}}} k_1! \cdots k_r! \text{vol}(\mathcal{C}, \mathbf{k}) \quad (2.6)$$

A simple mixed cell  $\mathcal{C}$  spans a cube and can be denoted by  $\mathcal{C} = (C_1, C_2, \dots, C_r)$ , with  $C_i = \{\mathbf{c}_{0i}, \mathbf{c}_{1i}, \dots, \mathbf{c}_{k_i i}\}$ ,  $i = 1, 2, \dots, r$ . We can compute its volume by

$$\text{vol}(\mathcal{C}, \mathbf{k}) = \frac{1}{k_1! \cdots k_r!} | \det (\mathbf{c}_{11} - \mathbf{c}_{01}, \dots, \mathbf{c}_{1i} - \mathbf{c}_{0i}, \dots, \mathbf{c}_{k_i i} - \mathbf{c}_{0i}, \dots, \mathbf{c}_{k_r r} - \mathbf{c}_{0r}) | \quad (2.7)$$

Let's see the example, for Equation 1.3, we lifted the supports  $\widehat{A} = (\widehat{A}_1, \widehat{A}_2)$ , with

$$\begin{aligned} \widehat{A}_1 &= \{(3, 1, 1), (1, 2, 0), (0, 0, 0)\} \\ \widehat{A}_2 &= \{(4, 0, 0), (1, 1, 1), (0, 0, 0)\} \end{aligned} \quad (2.8)$$

Figure 6 shows the mixed subdivision of Figure 5 as induced by the lower hull of the sum of the lifted polytopes.

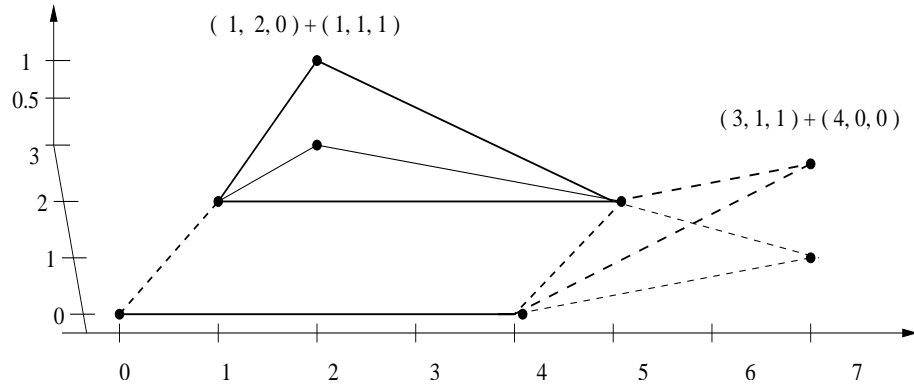


Figure 6. A mixed subdivision is regular if it is induced by a lifting

### 2.3 Polyhedral Homotopies

Polyhedral homotopies (proposed in [29] and [75]) provide efficient algorithms to solve systems with random coefficients, denoted by  $G(\mathbf{x}) = \mathbf{0}$ . To solve a given system  $F(\mathbf{x}) = \mathbf{0}$ , we then follow the solution paths defined by the linear homotopy  $H(\mathbf{x}, t) = (1-t)G(\mathbf{x}) + tF(\mathbf{x}) = \mathbf{0}$ , for  $t$  varying between 0 and 1. As shown in [48] (see also [45]) all isolated solutions of  $F(\mathbf{x}) = \mathbf{0}$  lie at the end of some solution path originated at a solution of  $G(\mathbf{x}) = \mathbf{0}$ .

Polyhedral homotopies are induced by *lifting functions*  $\omega : A \rightarrow \mathbb{R} : \mathbf{a} \mapsto \omega(\mathbf{a})$  applied to polynomials  $g$  of  $G$  in the above notation as  $\hat{g}(\mathbf{x}, t) = \sum_{\mathbf{a} \in A} c_{\mathbf{a}} \mathbf{x}^{\mathbf{a}} t^{\omega(\mathbf{a})}$ . Obviously, for  $t = 1$ , we have a polynomial, but at  $t = 0$ , we need coordinate transformations to start a polyhedral homotopy.

### 2.3.1 Coordinate Transformations

**Definition 2.3.1** Given  $\widehat{g}(\mathbf{x}, t) = \sum_{\mathbf{a} \in A} c_{\mathbf{a}} \mathbf{x}^{\mathbf{a}} t^{\omega(\mathbf{a})}$ . The right *coordinate transformations* are obtained as follows:

$$\text{replace } \begin{cases} x_i = y_i s^{v_i} \\ t = s^{v_{n+1}} \end{cases} \text{ to } \widehat{g}(\mathbf{x}, t) \quad (2.9)$$

Because the inner normal  $\mathbf{v}$  satisfies these equalities and inequalities,

$$\begin{aligned} \langle \widehat{\mathbf{c}}_i, \mathbf{v} \rangle &= \langle \widehat{\mathbf{c}}_j, \mathbf{v} \rangle, \quad \text{for } \widehat{\mathbf{c}}_i, \widehat{\mathbf{c}}_j \in \widehat{C} \\ \langle \widehat{\mathbf{a}}, \mathbf{v} \rangle &> \langle \widehat{\mathbf{c}}, \mathbf{v} \rangle, \quad \text{for } \widehat{\mathbf{c}} \in \widehat{C} \text{ and } \widehat{\mathbf{a}} \in \widehat{A} \setminus \widehat{C} \end{aligned} \quad (2.10)$$

This results in the homotopy composed of polynomials

$$\begin{aligned} \widehat{g}(\mathbf{y}, s) &= \widehat{g}|_C(\mathbf{y}, s) + \widehat{g}|_{A \setminus C}(\mathbf{y}, s) \\ \widehat{g}|_C(\mathbf{y}, s) &= \sum_{\mathbf{a} \in C} \mathbf{y}^{\mathbf{a}} s^{\langle \mathbf{a}, \mathbf{v} \rangle} \\ \widehat{g}|_{A \setminus C}(\mathbf{y}, s) &= \sum_{\mathbf{a} \in A \setminus C} \mathbf{y}^{\mathbf{a}} s^{\langle \mathbf{a}, \mathbf{v} \rangle} \end{aligned} \quad (2.11)$$

In particular: the equalities on  $\mathbf{v}$  ensure that the power of  $s$  in  $\widehat{g}|_C$  is the same, say  $p$ , and the inequalities on  $\mathbf{v}$  ensure that all powers of  $s$  in  $\widehat{g}|_{A \setminus C}$  are strictly larger than  $p$ . Therefore, we have a homotopy in  $s$ , starting at  $s = 0$  at a system so sparse that is trivial to solve, and ending at  $s = 1$  at the random coefficient system  $G(\mathbf{x}) = \mathbf{0}$ .

**Example 2.3.1** Consider the following polynomial system

$$G(\mathbf{x}) = \begin{cases} c_{22}x_1^2x_2^2 + c_{10}x_1 + c_{01}x_2 = 0 \\ c_{11}x_1x_2 + c_{10}x_1 + c_{01}x_2 + c_{00} = 0 \end{cases}, \quad \mathcal{A} = (A_1, A_2) \quad (2.12)$$

There's a generic lifting function  $\omega$  which defines the polyhedral homotopy

$$\widehat{G}(\mathbf{x}, t) = \begin{cases} c_{22}x_1^2x_2^2t^0 + c_{10}x_1t^0 + c_{01}x_2t^0 = 0 \\ c_{11}x_1x_2t + c_{10}x_1t + c_{01}x_2t + c_{00}t^0 = 0 \end{cases}, \quad \widehat{\mathcal{A}} = (\widehat{A}_1, \widehat{A}_2) \quad (2.13)$$

Suppose

$$\begin{aligned} \partial_{\mathbf{v}}\widehat{A}_1 &= \{(2, 2, 0), (1, 0, 0)\} \\ \partial_{\mathbf{v}}\widehat{A}_2 &= \{(1, 0, 1), (0, 0, 0)\} \end{aligned} \quad (2.14)$$

spans a subsystem of  $\widehat{G}$  whose supports spans a face on the lower hull of  $\widehat{P}$ . Then the inner normal  $\mathbf{v} = (v_1, v_2, v_3)$  must satisfy the following constraints,

$$\begin{cases} 2.v_1 + 2.v_2 + 0.v_3 = 1.v_1 + 0.v_2 + 0.v_3 < 0.v_1 + 1.v_2 + 0.v_3 \\ 1.v_1 + 0.v_2 + 1.v_3 = 0.v_1 + 0.v_2 + 0.v_3 < 1.v_1 + 1.v_2 + 1.v_3 \\ 1.v_1 + 0.v_2 + 1.v_3 = 0.v_1 + 0.v_2 + 0.v_3 < 0.v_1 + 1.v_2 + 1.v_3 \end{cases} \quad (2.15)$$

solve it and choose  $v_3 = 2$ , then we get an inner normal  $\mathbf{v} = (-2, 1, 2)$  defining a face  $\partial_{\mathbf{v}}\widehat{A}$  of the lower hull of  $\widehat{A}$ .

With this  $\mathbf{v}$ , we can define the following coordinate transformation

$$\begin{cases} x_1 &= y_1 s^{-2} \\ x_2 &= y_2 s \\ t &= s^2 \end{cases} \quad (2.16)$$

and apply it to  $\widehat{G}(\mathbf{x}, t)$ , we get

$$\widehat{G}(\mathbf{y}, s) = \begin{cases} c_{22}y_1^2y_2^2s^{-2} + c_{10}y_1s^{-2} + c_{01}y_2s = 0 \\ c_{11}y_1y_2s + c_{10}y_1s^0 + c_{01}y_2s^3 + c_{00}s^0 = 0 \end{cases} \quad (2.17)$$

After multiplying using the lowest power of  $s$ , we find

$$\widehat{G}(\mathbf{y}, s) = \begin{cases} c_{22}y_1^2y_2^2 + c_{10}y_1 + c_{01}y_2s^3 = 0 \\ c_{11}y_1y_2s + c_{10}y_1 + c_{01}y_2s^3 + c_{00} = 0 \end{cases} \quad (2.18)$$

For the inner normal  $\mathbf{v} = (-2, 1, 2)$ , we have a homotopy in  $s$ . When  $s=0$ , we have a binomial system

$$\widehat{G}(\mathbf{y}, s = 0) = \begin{cases} c_{22}y_1^2y_2^2 + c_{10}y_1 = 0 \\ c_{10}y_1 + c_{00} = 0 \end{cases} \quad (2.19)$$

It is easy to solve and we will get 2 start solutions. For Example 2.3.1, we also have another inner normal  $v = (1, -2, 1)$  which defines another homotopy and  $\widehat{G}(\mathbf{y}, s = 0) = \mathbf{0}$  will have 4 start solutions.

### 2.3.2 The Cayley Trick

Mixed volumes were defined by Minkowski who showed that the volume of a linear combination of polytopes is a homogeneous polynomial in the factors of the combination. The coefficients of this polynomial are mixed volumes. We will visualize this theorem on a simple example by the Cayley trick.

The Cayley trick ([23], Proposition 1.7, page 274) is a method to rewrite a certain resultant as a discriminant of one single polynomial with additional variables. The polyhedral version of this trick as in ([64], Lemma 5.2) is due to Bernd Sturmfels. See [27] for another application of this trick.

**Definition 2.3.2** The *Cayley embedding* places the sets  $A_i$  of  $\mathcal{A}$  on the vertices of the  $(n - 1)$ -dimensional standard basis simplex  $S$ . In particular:

$$\begin{aligned} \mathcal{E}(\mathcal{A}) &= \bigcup_{i=1}^n \mathcal{E}_i(A_i) \\ \mathcal{E}_i(A_i) &= \{(\mathbf{a}, \mathbf{e}_i) \mid \mathbf{a} \in A_i\}, \quad \mathbf{e}_i : i\text{th vertex of } S \end{aligned} \tag{2.20}$$

This means that  $\mathbf{e}_1$  is  $\mathbf{0}$  and  $\mathbf{e}_i$  is also zero except for 1 at position  $i - 1$ .

To solve systems  $F(\mathbf{x}) = \mathbf{0}$ ,  $F = (f_1, f_2, \dots, f_n)$  with different supports  $\mathcal{A} = (A_1, A_2, \dots, A_n)$ ,  $A_i$  is the support of  $f_i$ , we use regular mixed-cell configurations, its definition will be defined below.

**Definition 2.3.3** Cells in a regular triangulation of  $\mathcal{E}(\mathcal{A})$  which are spanned by exactly two points of every support  $A_i$  are called *mixed*. Extending our notation of  $\Delta_\omega$  to denote the

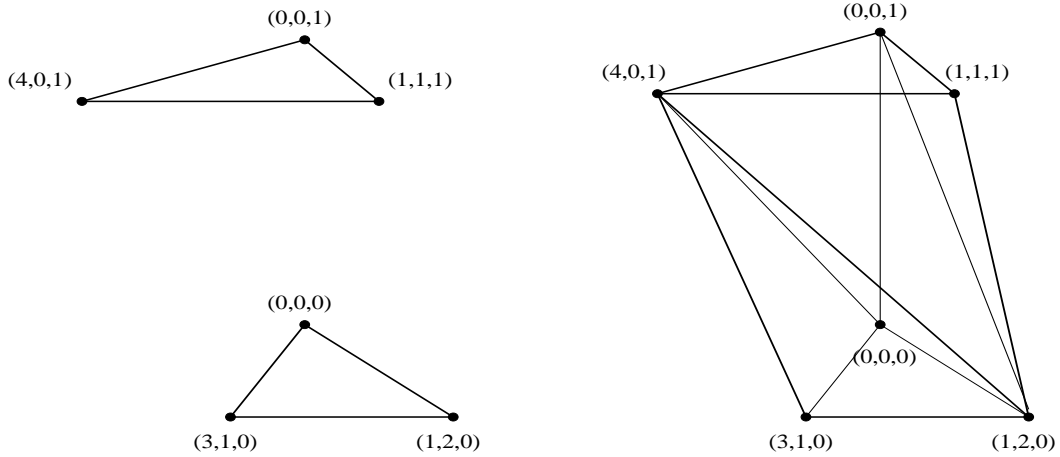


Figure 7. The Cayley polytope of two polygons. The first polygon is placed at the vertex  $(0,0,0)$ , the second polygon is placed at  $(0,0,1)$ .

collection of mixed cells of any regular triangulation of  $\mathcal{E}(\mathcal{A})$ , we then call  $\Delta_\omega$  a *regular mixed-cell configuration*, generalizing the volume to the *mixed volume*

$$\text{Vol}(\mathcal{A}) = \sum_{C \in \Delta_\omega} \text{Vol}(C) \quad (2.21)$$

Figure 7 illustrates Cayley embedding for Example 1.1.1. The Cayley polytope of our example is so simple that a triangulation is obvious see (Figure 8). As every simplex has four vertices, either the simplex has three vertices from the same polygon ( and the fourth one of the other polygon ), or the simplex has two vertices of each polygon. A simplex of the first type is called unmixed, a simplex of the second type is mixed. Imagine taking slices parallel to the base of the Cayley polytope. These slices produce scaled copies of the original polygons in the

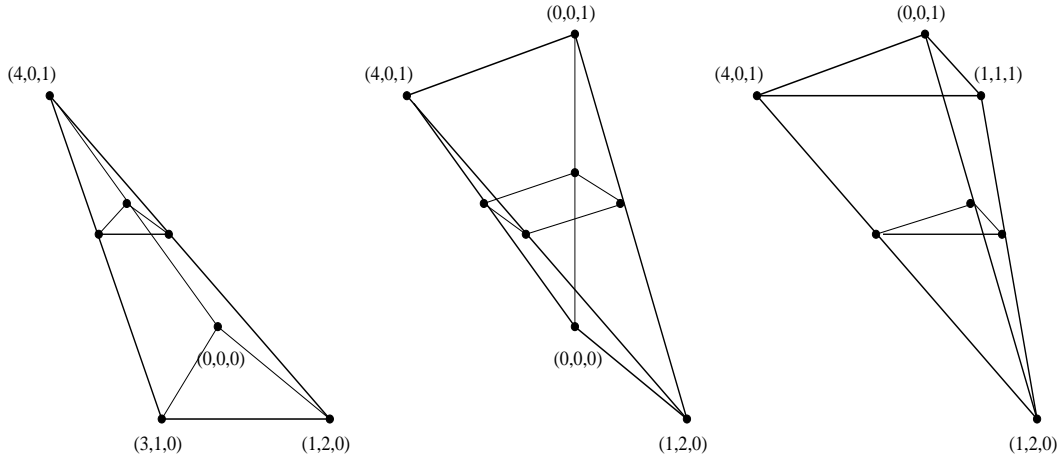


Figure 8. A triangulation of the Cayley polytope. The middle simplex is mixed, the other two simplices are unmixed.

unmixed simplices. In the mixed simplex we find one scaled edge from the first and another scaled edge from the second polygon, see Figure 8.

On Figure 9 we see in the cross section of the Cayley polytope a mixed subdivision of the convex combination  $\lambda_1 P_1 + \lambda_2 P_2$ ,  $\lambda_1 + \lambda_2 = 1$ ,  $\lambda_1 \geq 0$  and  $\lambda_2 \geq 0$ , where  $P_1$  defines the base and  $P_2$  is at the top of the polytope. The areas of the triangles in the cross section are  $\lambda_1^2 \times \text{area}(P_1)$  and  $\lambda_2^2 \times \text{area}(P_2)$ , as each side of the triangle is scaled by  $\lambda_1$  and  $\lambda_2$  respectively. The area of the cell in the subdivision spanned by one edge of  $P_1$  (scaled by  $\lambda_1$ ) and the other edge of  $P_2$  (scaled by  $\lambda_2$ ) is scaled by  $\lambda_1 \times \lambda_2$ , as we move the cross section.

For the relation to sparse elimination theory, we refer to [17], [23], and [64], see also [27] for geometric applications. Algorithm 2.3.1 follows [29].

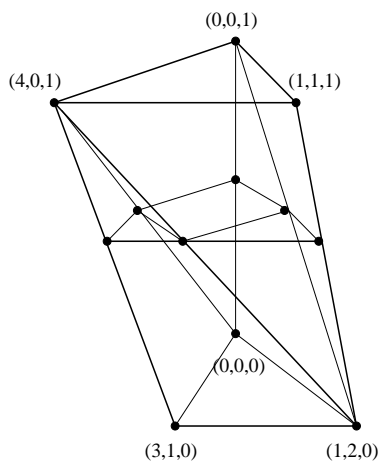


Figure 9. A mixed subdivision induced by a triangulation of the Cayley polytope.

The execution of the coordinate transformation to create the polyhedral homotopy has a cost linear in  $n$  and the number of monomials in the system. As  $\widehat{G}|_C(\mathbf{y}, 0) = \mathbf{0}$  has exactly  $n + 1$  distinct monomials, the cost of solving the start system is equivalent to solving one linear system. The most computationally intensive stage in Algorithm 2.3.1 is the path tracking, because tracking one solution path typically involves solving a couple of hundreds of linear systems in the Newton corrector.

In addition to leading to more solution paths, the difficulty with applying polyhedral homotopies is the height of the power of the continuation parameter, a problem first addressed in [74] by dynamic lifting, later in [21] by balancing, and by scaling in [33]. While solving large 12 dimensional systems from mechanical design (see the last section below), we discovered nu-

---

**Algorithm 2.3.1** *Polyhedral homotopies to solve a generic system.*

<b>Input:</b> $\Delta_\omega, G(\mathbf{x}) = \mathbf{0}$ .	mixed-cell config and generic system
<b>Output:</b> $G^{-1}(\mathbf{0})$ .	all solutions to $G(\mathbf{x}) = \mathbf{0}$
<b>for all</b> $C \in \Delta_\omega$ <b>do</b>	enumerate all $C$ with inner normals
create a polyhedral homotopy $\widehat{G}(\mathbf{y}, s) = \mathbf{0}$ ;	apply coordinate transformation
solve the start system $\widehat{G} _C(\mathbf{y}, 0) = \mathbf{0}$ ;	$\widehat{G} _C(\mathbf{y}, 0) = \mathbf{0}$ has $\text{Vol}(C)$ solutions
track $\mathbf{y}(s)$ : $\widehat{G}(\mathbf{y}(s), s) = \mathbf{0}$ , for $s$ from 0 to 1;	track $\text{Vol}(C)$ solution paths
<b>end for.</b>	

---

merical instabilities when solving semi-mixed start systems. In [40] we describe a numerically stable algorithm to resolve this issue.

## 2.4 Distribution of the Workload

Polyhedral homotopies are applied to solve generic systems  $G(\mathbf{x}) = \mathbf{0}$ . The “generic” means in practice that the coefficients of  $G$  are random points on the complex unit circle. Because of this randomness, all solution paths are expected to follow the same uniform behavior, and the tracking will require the same amount of computational work. Therefore, reducing the communication overhead to a minimum, a static workload assignment seems to be the best choice. Using the manager-worker<sup>1</sup> paradigm, Algorithm 2.4.1 summarizes our parallel version of the polyhedral homotopies.

---

<sup>1</sup>The terminology we use is less common than “master-slave”, but avoids the unpleasant historical connotations.



manager	worker1	worker2	worker3
Vol(cell1)=5	→ #paths(cell1) : 5		
Vol(cell2)=4	→ #paths(cell2) : 4		
Vol(cell3)=4	→ #paths(cell3) : 4		
Vol(cell4)=6	→ #paths(cell4) : 1	#paths(cell4) : 5	
Vol(cell5)=7	→ #paths(cell5) : 7		
Vol(cell6)=3	→ #paths(cell6) : 2		#paths(cell6) : 1
Vol(cell7)=4	→ #paths(cell7) : 4		
Vol(cell8)=8	→ #paths(cell8) : 8		
total#paths : 41	#paths to track : 14	#paths to track : 16	#paths to track : 11

Figure 10. The example to show how the manager assigns the cells (with respective volumes listed) and #paths to the workers in the static workload distribution.

---

**Algorithm 2.4.2** *Static distribution of cells executed by the manager.*

<b>Input:</b> $\Delta_\omega, V, p.$	mixed-cell configuration, volume, #processors
<b>Output:</b> $G^{-1}(\mathbf{0}).$	all solutions to $G(\mathbf{x}) = \mathbf{0}$

---

<b>for</b> $i$ <b>from</b> 1 <b>to</b> $p - 1$ <b>do</b>	the manager is processor 0
$load[i] := distribute(V, p, i);$	#paths for $i$ -th worker
send $load[i]$ to $i$ ;	message to $i$ -th worker
<b>end for</b> ;	$\lfloor V/(p - 1) \rfloor \leq load[i] \leq \lceil V/(p - 1) \rceil$
$i := 1$ ;	start at worker 1
<b>for each</b> $C \in \Delta_\omega$ <b>do</b>	$C$ defines $Vol(C)$ many paths
$\#paths := Vol(C);$	first index of start solution
$j := 1$ ;	
<b>while</b> $\#paths > 0$ <b>do</b>	
<b>if</b> $\#paths \leq load[i]$ <b>then</b>	$i$ -th worker can handle $C$
send $(C, j, j + \#paths - 1)$ to $i$ ;	message to $i$ -th worker
$load[i] := load[i] - \#paths;$	reduce load for $i$ -th worker
$\#paths := 0;$	done with cell $C$
<b>else</b>	too many paths for $i$ -th worker
send $(C, j, j + load[i] - 1)$ to $i$ ;	message to $i$ -th worker
$j := j + load[i];$	update index of start solution
$\#paths := \#paths - load[i];$	reduce #paths to track
$load[i] := 0;$	finished with $i$ -th worker
$i := i + 1$ ;	move to the next worker
<b>end if</b> ;	
<b>end while</b> ;	
<b>end for</b> .	

---

**Algorithm 2.4.3** *Static distribution of cells executed by all the workers.*

<b>Input:</b> $G(\mathbf{x}) = \mathbf{0}.$	generic system
<b>Output:</b> a subset of $G^{-1}(\mathbf{0}).$	as many solutions as load of the worker

---

$count := 0;$	initialize the counter
receive load from manager;	#paths the worker has to track
<b>while</b> $count \leq load$ <b>do</b>	as long as not finished
receive $(C, k, l);$	receive cell and indices to start solutions
create polyhedral homotopy $\widehat{G}_C(\mathbf{y}, s) = \mathbf{0};$	perform coordinate transformation
solve the start system $\widehat{G}_C(\mathbf{y}, 0) = \mathbf{0};$	one linear system to solve
track paths from solutions $k$ to $l$ ;	track $l - k + 1$ paths
send solutions to manager;	message to manager reporting results
$count := count + l - k + 1;$	update the counter
<b>end while</b> .	

---

The static workload distribution of the cells of the mixed-cell configuration  $\Delta_\omega$  is formalized in Algorithm 2.4.2 and Algorithm 2.4.3, executed respectively by the manager and each worker. The data the manager sends to each worker is the triplet  $(C, k, l)$ , with  $C \in \Delta_\omega$  a cell,  $k$  and  $l$  are indices to the first and last start solutions in the polyhedral homotopy defined by the cell  $C$ .

The distribute function in Algorithm 2.4.2 determines the workload for each worker. In case the total number of paths is not divisible by the number of workers, workers with a lower number will receive some extra paths. This bias in the workload distribution is justified because workers with lower number receive their load earlier than the rest. Algorithm 2.4.3 details the actions performed by every worker. Because solving a start system involves only the solution of a linear system, each worker solves the start system and tracks only the assigned paths.

## 2.5 Dynamic Workload Distribution

For very sparse or generic problems, a static workload distribution seems to be the best choice. However, for polynomials which are not so sparse as the mechanisms design problems, dynamic load balancing is needed.

The dynamic workload distribution of the cells of the mixed-cell configuration  $\Delta_\omega$  is formalized in Algorithm 2.5.1 and Algorithm 2.5.2, executed respectively by the manager and each worker. The data the manager sends to each worker is  $(C, index)$ , with  $C \in \Delta_\omega$  a cell,  $index$  is the index of the start solutions in the polyhedral homotopy defined by the cell  $C$ .

Algorithm 2.5.1 distributes the load according to the number of cells and number of processors. In case the number of cells is less than the number of processors, we will distribute

the load path by path, otherwise we will distribute the load cell by cell except for the last cell which will be distributed path by path.

Now we will use the same example which we showed in static load balancing, to state how the dynamic workload distribution algorithm 2.5.1 works. In the example, since the number of cells (which is 8) is larger than the number of processors (which is 4), therefore we will distribute the load cell by cell except for the last cell. The result is shown in Figure 11. From the figure we know that the path tracking will require a different amount of computational work. So at the end, the dynamic workload balancing will be  $41=14+16+11$ , this is different from the static workload balancing which is  $41=14+14+13$ .

## **2.6 Computational Results**

In this section we summarize our computational experiments on two typical benchmark applications. The first class of polynomial systems is a frequently used and widely known problem in computational algebraic geometry, the so-called cyclic  $n$ -roots problem. Our second example comes from mechanism design.

### **2.6.1 Hardware and Software**

We developed and tested our software using two Rocketcalc clusters, with four and eight processors respectively. Combined with the dual processor server machine, our configuration has a total of fourteen 2.4Ghz processors, connected by a 100Mbit Ethernet network.

We also ran our software on UIC's "argo", a cluster of sixty-eight PCs: including two master machines, sixty-four compute machines, and two file servers. The nodes are heterogeneous, some



---

**Algorithm 2.5.1** *Dynamic distribution of cells executed by the manager.*

<b>Input:</b> $\Delta_\omega, V, p.$	mixed-cell config, volume, #processors
<b>Output:</b> $G^{-1}(\mathbf{0}).$	all solutions to $G(\mathbf{x}) = \mathbf{0}$
<b>if</b> $\#\Delta_\omega \leq p$ <b>then</b>	distribute path by path
<b>for</b> $i$ <b>from</b> 1 <b>to</b> $p - 1$ <b>do</b>	the manager is processor 0
compute $(C, index);$	determine which path of which cell
send $(C, index)$ to $i;$	message to $i$ -th worker
<b>end for;</b>	
<b>for</b> $k$ <b>from</b> $p$ <b>to</b> $V + p - 1$ <b>do</b>	distribute the rest of the paths
compute $(C, index)$ from $k;$	determine which path of which cell
receive $(i, requestTag);$	$i$ -th worker requests a job
send $k$ to $i;$	$i$ -th worker expects a job/terminate
<b>if</b> $k \leq V$ <b>then</b>	
send $(C, index)$ to $i;$	send job to $i$ -th worker
<b>end if;</b>	
<b>end for;</b>	
<b>else</b>	distribute cell by cell
<b>for</b> $i$ <b>from</b> 1 <b>to</b> $p - 1$ <b>do</b>	the manager is processor 0
send $(C_i, -1)$ to $i;$	message to $i$ -th worker
<b>end for;</b>	
<b>for</b> $k$ <b>from</b> $p$ <b>to</b> $\#\Delta_\omega - 1 + V_l + p - 1$ <b>do</b>	$V_l$ is #paths of the last cell
receive $(i, requestTag);$	$i$ -th worker requests a job
<b>if</b> $k > \#\Delta_\omega + V_l$ <b>then</b>	all cells processed
$k := V + 1;$	$k$ signals termination
<b>end if;</b>	
send $k$ to $i;$	$i$ -th worker terminates according to $k$
<b>if</b> $k \leq \#\Delta_\omega$ <b>then</b>	
send $(C_k, -1)$ to $i;$	send $i$ -th worker a new cell
<b>else if</b> $k \leq \#\Delta_\omega + V_l$ <b>then</b>	
send $(C_l, k - \#\Delta_\omega + 1)$ to $i;$	send $i$ -th worker a new path
<b>end if;</b>	
<b>end for;</b>	
<b>end if.</b>	

---

**Algorithm 2.5.2** *Dynamic distribution of cells executed by all the workers.*

<b>Input:</b> $G(\mathbf{x}) = \mathbf{0}, V.$	generic system, total #paths
<b>Output:</b> a subset of $G^{-1}(\mathbf{0}).$	as many solutions as load of the worker
<b>do</b>	
receive $k;$	terminate according to $k$
<b>if</b> $k > V$ <b>then</b> exit loop; <b>end if;</b>	terminate loop
receive $(C, index);$	receive cell and indices to start solutions
create polyhedral homotopy $\widehat{G}_C(\mathbf{y}, s) = \mathbf{0};$	perform coordinate transformation
solve the start system $\widehat{G}_C(\mathbf{y}, 0) = \mathbf{0};$	one linear system to solve
track paths according to index;	track $Vol(C)$ paths or just one path
send solutions to manager;	message to manager reporting results
<b>end do.</b>	

---

processors run at 2.8Ghz, others at 3.0Ghz. Compared to our Rocketcalc clusters, the faster network connections of argo led to an improved speedup.

Our software is an extension of PHCpack [71]. Our main programs are written in C, using the MPI library, and call the path tracking and homotopy facilities in PHCpack.

### 2.6.2 Experimental Results on the cyclic $n$ -roots problem

The cyclic  $n$ -roots problem defines a family of polynomial systems widely used for benchmarking. The systems occur in Fourier analysis [5]. Progress on these systems is reported in [6], [7], [14], [18], [19], [20], and [44].

**Example 2.6.1** *The polynomial system of the cyclic 9-roots problem:*

$$\begin{cases} f_i = \sum_{j=0}^8 \prod_{k=1}^i x_{(k+j) \bmod 9} = \mathbf{0}, & i = 1, 2, \dots, 8 \\ f_9 = x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 - 1 = 0. \end{cases} \quad (2.22)$$

In Table I we list the computational results for our cluster configuration, for increasing values of the problem size  $n$ . As is typical for polynomial systems solving the number of solutions grows exponentially in the number of variables  $n$  in the system. Even for our relatively small cluster configuration, larger problems (in the range of several hundreds of thousands of solutions) are well within reach.

Table II shows the speedup on the cyclic 7-roots problem for an increasing number of workers. Because the manager does no path tracking, the time spent when using one worker corresponds to the sequential time. First we list the wall time and speedup for the static and

<b>Problem</b>	<b>#Paths</b>	<b>CPU Time</b>
cyclic 5-roots	70	0.13m
cyclic 6-roots	156	0.19m
cyclic 7-roots	924	0.30m
cyclic 8-roots	2,560	0.78m
cyclic 9-roots	11,016	3.64m
cyclic 10-roots	35,940	21.33m
cyclic 11-roots	184,756	2h 39m
cyclic 12-roots	500,352	24h 36m

TABLE I

WALL TIME FOR START SYSTEMS TO SOLVE THE CYCLIC  $N$ -ROOTS PROBLEMS, USING 13 WORKERS, WITH STATIC LOAD DISTRIBUTION.

dynamic distribution on our cluster. The last 2 columns list wall time and speedup for the dynamic distribution on argo.

Moreover, we observe a very good distribution of the workload, as the difference in time between the workers is minor. With 13 workers we obtain an almost tenfold speedup, which is acceptable. For dynamic workload distribution we get 11.3 speedup which is better than static. The dynamic workload balancing algorithm on argo achieves a close to optimal speedup. We also show the speedup in Figure 12.

### **2.6.3 Mechanism Design: the computation of reachable surfaces**

We now consider the problem coming from mechanism design. The problem is about finding surfaces that contain a set of points generated by a displaced rigid body. We focus on the surfaces reachable by the wrist center of an articulated serial chain, see [67] and [70].

#workers	Static versus Dynamic on our cluster				Dynamic on argo	
	Static	Speedup	Dynamic	Speedup	Dynamic	Speedup
1	50.7021	–	53.0707	–	29.2389	–
2	24.5172	2.1	25.3852	2.1	15.5455	1.9
3	18.3850	2.8	17.6367	3.0	10.8063	2.7
4	14.6994	3.4	12.4157	4.2	7.9660	3.7
5	11.6913	4.3	10.3054	5.1	6.2054	4.7
6	10.3779	4.9	9.3411	5.7	5.0996	5.7
7	9.6877	5.2	8.4180	6.3	4.2603	6.9
8	7.8157	6.5	7.4337	7.1	3.8528	7.6
9	7.5133	6.8	6.8029	7.8	3.6010	8.1
10	6.9154	7.3	5.7883	9.2	3.2075	9.1
11	6.5668	7.7	5.3014	10.0	2.8427	10.3
12	6.4407	7.9	4.8232	11.0	2.5873	11.3
13	5.1462	9.8	4.6894	11.3	2.3224	12.6

TABLE II

WALL TIME IN SECONDS FOR AN INCREASING NUMBER OF WORKERS TO SOLVE A START SYSTEM FOR THE CYCLIC 7-ROOTS PROBLEM.

There are five basic joints,

1. Revolute: denoted by  $R$ , allow pure rotation about.
2. prismatic: denoted by  $P$ , allow a linear slide along.
3. Cylindric: denoted by  $C$ , the sequence of a revolute and a prismatic joint constructed so that their axes are parallel.
4. Universal: denoted by  $T$ , the sequence of two revolute joints which have axes that intersect at right angles.

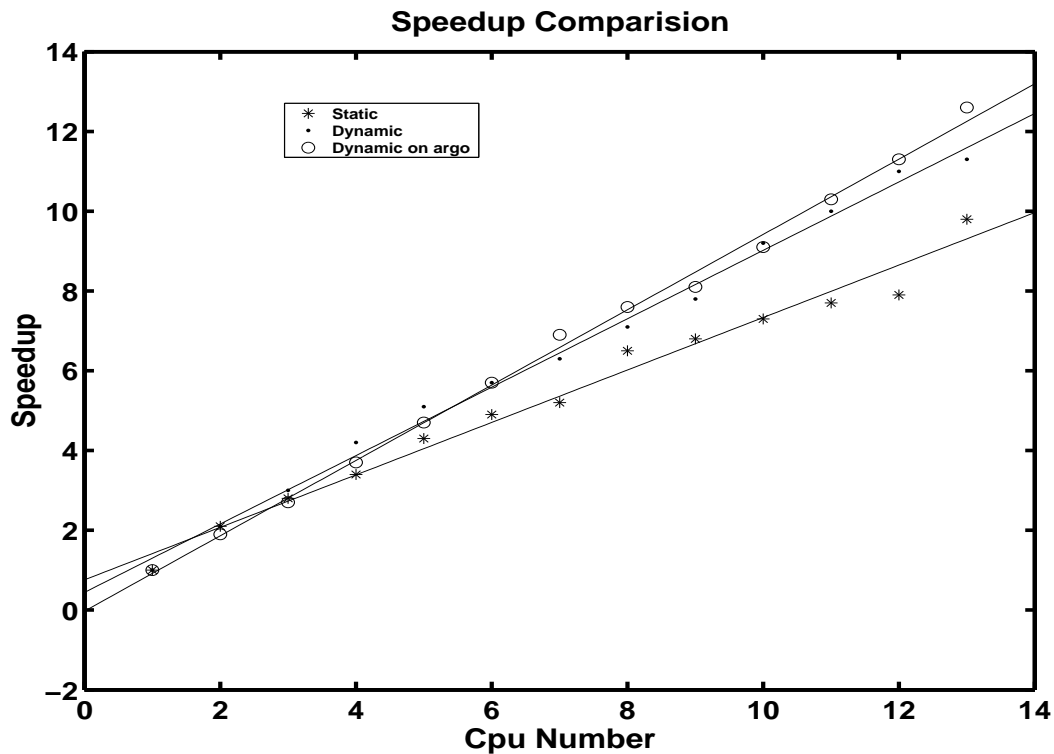


Figure 12. Speedup comparison for the cyclic 7-roots problem

5. Spherical: denoted by  $S$ , a three revolute chain with concurrent joint axes.

The combinations available for revolute and prismatic joints yield four basic chains:  $PPS$ ,  $RPS$ ,  $PRS$  and  $RRS$ . The reachable surfaces defined by these chains are the plane, the circular hyperboloid, the elliptic cylinder and the general torus. In [69] and [68] using results of [83], the POLSYS\_GLP extension of HOMPACK [81] (see also [82] and [83]) used a linear-product start systems to solve these type of systems.

These systems are more challenging to polyhedral homotopies than the cyclic  $n$ -roots problems, because they are not so sparse.

For example, an elliptic cylinder is generated by a circle that has its center swept along a line  $L(t) = \mathbf{B} + t\mathbf{S}_1$  circle maintains a constant direction  $\mathbf{S}_2$  at an angle  $\alpha$  relative to the direction  $\mathbf{S}_1$  of  $L(t)$ , see Figure 13. The major axis of the elliptic cross-section is the radius  $R$  of the circle and the minor axis is  $R \cos \alpha$ . This surface is generated by the wrist center of a  $PRS$  chain that has its  $P$ -joint aligned with the axis  $L(t)$  and its  $R$ -joint positioned so that its axis is along  $\mathbf{S}_2$ .

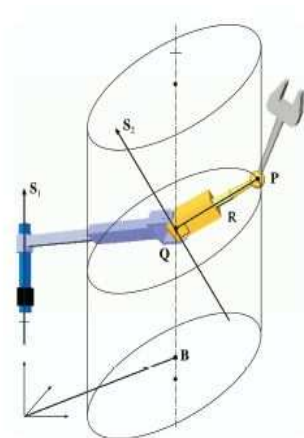


Figure 13. The elliptic cylinder reachable by a PRS serial chain.

Figure 4.4 of [66].

A point  $P$  on the elliptic cylinder must satisfy the corresponding polynomial system which has the total degree  $4^9 2^3 = 2,097,152$ .

$$E(\mathbf{z}) = \begin{cases} (\mathbf{P}^2 - (\mathbf{P}^2 \cdot \mathbf{K})\mathbf{S}_1 + \mathbf{Q})^2 - (\mathbf{P}^1 - (\mathbf{P}^1 \cdot \mathbf{K})\mathbf{S}_1 + \mathbf{Q})^2 = 0 \\ \vdots \\ (\mathbf{P}^{10} - (\mathbf{P}^{10} \cdot \mathbf{K})\mathbf{S}_1 + \mathbf{Q})^2 - (\mathbf{P}^9 - (\mathbf{P}^9 \cdot \mathbf{K})\mathbf{S}_1 + \mathbf{Q})^2 = 0 \\ \mathbf{S}_1 \cdot \mathbf{S}_1 - 1 = 0 \\ \mathbf{K} \cdot \mathbf{S}_1 - 1 = 0 \\ \mathbf{Q} \cdot \mathbf{K} = 0 \end{cases} \quad (2.23)$$

Nine equations of this example have 201 monomials, of which 102 are vertices and belong to the random coefficient start system. This makes the evaluation of the polynomials more expensive. The mixed volume of this system equals 125,888, which is about half of the bound 247,968 based on the degrees of the system, used in [69]. The semi-mixed nature of the system (i.e.: nine equations share the same support) is exploited by the dynamic lifting algorithm of [74]. While the computation of the mixed volume is not the main cost, we also ran the program MixedVol [22] to confirm this number.

The second example we test is the Circular Torus. The circular torus is generated by sweeping a circle around an axis so that its center traces a second circle. See Figure 14.

The third example we test is the General Torus. The general torus is defined by sweeping a circle that has a general orientation in space around an arbitrary axis. See Figure 15.



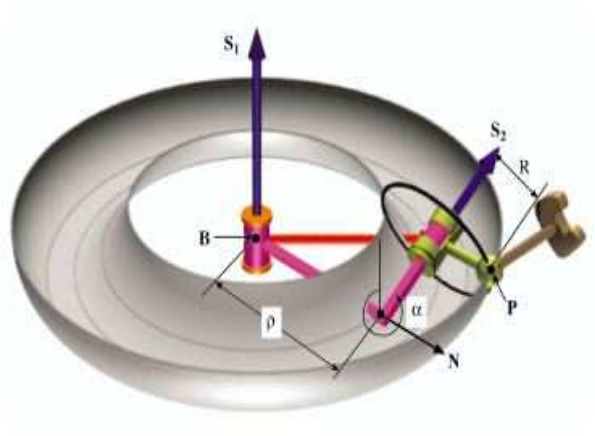


Figure 15. The general torus reachable by the wrist center of an RRS serial chain.  
Figure 4.8 of [66].

Surface	Bounds on #Solutions			dynamic load distribution	
	Total degree	LPD bound	Mixvol	our cluster	time on argo
elliptic cylinder	2,097,152	247,968	125,888	11h 33m	6h 12m
circular torus	2,097,152	868,352	474,112	7h 17m	4h 3m
general torus	4,194,304	448,702	226,512	14h 15m	6h 36m

TABLE III

WALL TIME FOR MECHANISM DESIGN PROBLEMS ON OUR CLUSTER AND ARGO.

## CHAPTER 3

# PARALLEL HOMOTOPY ALGORITHMS TO SOLVE POLYNOMIAL SYSTEMS

Homotopy continuation methods to compute numerical approximations to all isolated solutions of a polynomial system are known as “pleasingly parallel” because of their low communication overhead. Because so many important problems remain unsolved mainly due to their intrinsic computational complexity, it would be embarrassing not to develop parallel implementations of polynomial homotopy continuation methods.

In this chapter, we concern the development of “parallel PHCpack” which is a project started a couple of years ago, developed by my advisor Jan Verschelde and his student Yuesong Wang (parallel Pieri homotopy), and continued with Anton Leykin (parallel irreducible decomposition). Our motivation is to solve the large polynomial system which has more than 100,000 solutions, e.g.: polynomial systems coming from mechanism design. This chapter is concerned with 3 issues: efficiency, numerical stabilities and quality control. In Section 3.2, we introduce jumpstarting homotopies which will increase the performance of the software package PHCpack. In Section 3.3, we give a numerically stable solver for “Simplex” systems.

### 3.1 Motivation: we want to solve large systems

To solve a polynomial system  $F(\mathbf{x}) = \mathbf{0}$ , a homotopy  $H(\mathbf{x}, t) = \mathbf{0}$  connects  $F$  to a start system  $G(\mathbf{x}) = \mathbf{0}$  ( $G$  stands for “generic”, i.e.: all start solutions are regular), typically of the form

$$H(\mathbf{x}, t) = \gamma(1 - t)G(\mathbf{x}) + tF(\mathbf{x}) = \mathbf{0}, \quad \gamma \in \mathbb{C}, \quad (3.1)$$

where the random complex constant  $\gamma$  ensures with probability one that all solution of  $H(\mathbf{x}, t) = \mathbf{0}$  are regular for all  $t \in [0, 1)$ . Because of this regularity, predictor-corrector methods will track all solution paths defined by  $H(\mathbf{x}(t), t) = 0$ , as  $t$  moves from 0 to 1, starting at  $t = 0$  at the solution of  $G(\mathbf{x}) = \mathbf{0}$  and ending at  $t = 1$ , at approximate isolated solutions of  $F(\mathbf{x}) = \mathbf{0}$ .

We say that the system  $F$  is “*large*” if the homotopy we use to solve it requires more than 100,000 solutions to track. Although large does not always automatically imply “difficult”, numerical problems are more likely to occur. This chapter is concerned with three issues:

- For efficiency, it is undesirable to keep all solutions in the main memory.
- Numerical instabilities may occur as dimensions grow.
- Quality control on the computed solutions must be done fast.

Recent work produced two different software systems: PHoMpara [24] (a parallel version of PHoM [25]) and POLSYS\_GLP [68] (based upon HOMPACK [81]). The first software system uses polyhedral homotopies, while the second one applies linear-product start systems to solve polynomial systems. Regardless of the performance of these programs relative to PHCpack, PHCpack [71] offers both types of homotopies.

The parallel implementation of the path trackers in PHCpack started in a joint work with Yusong Wang [76] and yielded a parallel version of the Pieri homotopies [28], [30]. Parallel path tracking is discussed in [2], [11; 12], [26], [53], and [54]. Our computational experiments showed that distributing all path tracking jobs at the start performs well when all paths require the same amount of work. Otherwise, dynamic load balancing is needed to achieve an optimal performance.

The work in [37] reports on the parallel implementation of methods to decompose a positive dimensional solution set, using monodromy [58] and traces [59], as needed in a numerical irreducible decomposition [57]. The techniques presented in this thesis provide efficient homotopies to create *witness sets* of these positive dimensional solution sets see [61] and [62] for introductions to numerical algebraic geometry. The development of parallel polyhedral homotopies (described in [77]) will increase the capabilities of PHCpack to deal with solution sets of larger degrees.

The parallel software was developed using personal cluster computers from RocketCalc and ported to similar Beowulf clusters like UIC's supercomputer argo. Most recently, the parallel path tracking facilities of PHCpack were installed on NCSA's IBM pSeries 690 system running AIX 5.3. The use of MPI and a description of other interfaces to PHCpack can be found in [39].

In this chapter we present solutions to deal with the three issues raised above. To avoid the storage of all start solutions in the main memory, we propose to “jumpstart” homotopies, either by computing the roots whenever and wherever they are needed, or by reading the start solutions from file. For the sparsest class of polynomial systems, we discovered a numerically

stable solver which computes the magnitudes of the roots separately, avoiding numerical overflow or underflow. Thirdly, for efficient quality control of the results, the programs are allowed only one linear sweep through the files which contain the solutions.

As noted before, what we call “large” does not automatically imply “difficult”. The homotopies we consider in this thesis are *optimal* (a notion introduced in [28]): no solution path diverges to infinity, so the overall cost of the solver is polynomial in the output size.

### 3.2 Jumpstarting Homotopies

Homotopy continuation methods compute one solution at a time. Keeping all start solutions in main memory may decrease the overall performance, or even be impossible. Assuming a manager/worker protocol, our solution is the following:

1. The manager reads start solution from file “just in time”  
whenever a worker needs another path tracking job.
2. For total degree and linear-product start systems,  
it is simple to compute the solutions whenever needed.
3. As soon as worker reports the end of a solution path  
back to the manager, the solution is written to a file.

Solutions to total degree start systems can be computed very fast, faster than they can be retrieved from file. A lexicographical indexing scheme allows the manager to dictate only which node has to track which path. As all nodes know how to solve total degree start systems, they only need a number, reducing the communication overhead.

For example, a typical total degree start system may look like

$$G(x_1, x_2, x_3) = \begin{cases} x_1^4 - 1 = 0 \\ x_2^5 - 1 = 0 \\ x_3^3 - 1 = 0. \end{cases} \quad (3.2)$$

It has  $4 \times 5 \times 3 = 60$  solutions.

We can get the 25th solution via decomposition of 24 (we start counting from 0):  $24 = 1(5 \times 3) + 3(3) + 0$ . Let us verify this via lexicographic enumeration:

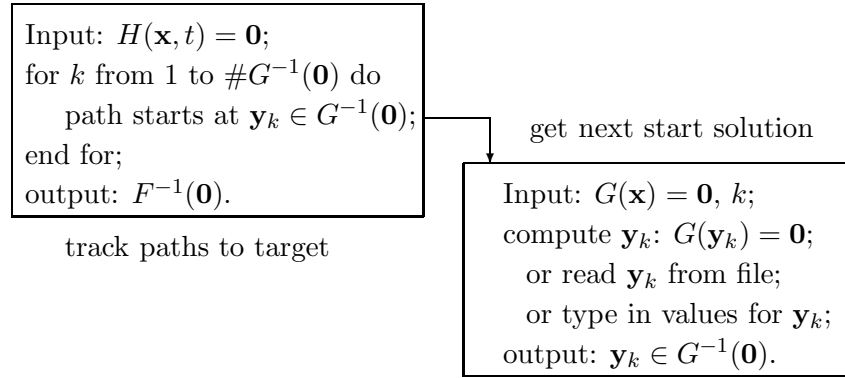
$$\begin{aligned} &000 \rightarrow 001 \rightarrow 002 \rightarrow 010 \rightarrow 011 \rightarrow 012 \rightarrow 020 \rightarrow 021 \rightarrow 022 \rightarrow 030 \rightarrow 031 \rightarrow 032 \rightarrow 040 \rightarrow 041 \rightarrow 042 \\ &100 \rightarrow 101 \rightarrow 102 \rightarrow 110 \rightarrow 111 \rightarrow 112 \rightarrow 120 \rightarrow 121 \rightarrow 122 \rightarrow \boxed{130} \rightarrow 131 \rightarrow 132 \rightarrow 140 \rightarrow 141 \rightarrow 142 \\ &200 \rightarrow 201 \rightarrow 202 \rightarrow 210 \rightarrow 211 \rightarrow 212 \rightarrow 220 \rightarrow 221 \rightarrow 222 \rightarrow 230 \rightarrow 231 \rightarrow 232 \rightarrow 240 \rightarrow 241 \rightarrow 242 \\ &300 \rightarrow 301 \rightarrow 302 \rightarrow 310 \rightarrow 311 \rightarrow 312 \rightarrow 320 \rightarrow 321 \rightarrow 322 \rightarrow 330 \rightarrow 331 \rightarrow 332 \rightarrow 340 \rightarrow 341 \rightarrow 342 \end{aligned} \quad (3.3)$$

Although examples for which the total degree homotopy is optimal are fairly rare, one interesting application appears in magnetism, posed by Shigetoshi Katsura [31], see also [8]. This application leads to a family of systems, which scales for up to any number of equations and variables. All equations in the systems are of degree two, except for one linear equation. The largest polynomial system in this family we considered has 21 equations and a total degree of  $2^{20} = 1,048,576$ . Recently, the `mpi2track` function in PHCpack tracked all 1,048,576 paths using a personal cluster of fourteen CPUs all running at a clockspeed of 2.4Ghz, in a traditional

manager/worker dynamic load distribution model. It took 32 hours and 44 minutes to complete the tracking, leading to an output file of 1.3Gb.

The program inversion picture is shown below. Whenever a worker finishes one path tracking job and needs another path to track, the manager will read a start solution from a file for this work just in time or the manager will get the start solution input by the user.

$$H(\mathbf{x}, t) = \gamma(1 - t)G(\mathbf{x}) + tF(\mathbf{x}) = \mathbf{0}, \quad \gamma \in \mathbb{C}, \quad t \in [0, 1].$$



While homotopies based on the total degree are rarely efficient, the sample principle of lexicographic enumeration of the start solutions applies to linear-product start systems. These start systems first occurred in [78], using multi-homogeneous homotopies [47] and were generalized in [72]. Compared to the total degree, homotopies using linear-product start systems typically follow far fewer solution paths than the total degree.

All equations in a linear-product start system are products of linear equations, of the form the system in (Equation 3.4). Every  $\cdots$  in (Equation 3.4) represents a linear equation with randomly chosen coefficients.

$$G(\mathbf{x}) = \begin{cases} (\dots) \cdot (\dots) \cdot (\dots) \cdot (\dots) = 0 \\ (\dots) \cdot (\dots) \cdot (\dots) \cdot (\dots) \cdot (\dots) = 0 \\ (\dots) \cdot (\dots) \cdot (\dots) = 0 \end{cases} \quad (3.4)$$

In [72], it was shown that the random choice of the coefficients of the linear factors of the products in the linear-product start systems implies that the maximal number of isolated solutions is attained. Moreover, if every monomial in the target system  $F(\mathbf{x}) = \mathbf{0}$  also occurs in the corresponding equation of the start system  $G(\mathbf{x}) = \mathbf{0}$ , then all isolated solutions of  $F(\mathbf{x}) = \mathbf{0}$  lie at the end of some solution path defined by a homotopy using a linear-product start system  $G(\mathbf{x}) = \mathbf{0}$ . Efficient implementations of these type of homotopies were described in [83] and [68].

Just like (Equation 3.2), the solution of the start system in (Equation 3.4) can be enumerated lexicographically. As the linear-product start system is stored on file in its product form, one does not need storing the start solutions on file. Moreover, any node in a parallel computer can solve for one particular solution. While the main motivation is to avoid to store the complete list of start solutions in the main memory, an additional advantage is a reduced communication overhead: instead of passing the start solution vector from manager to path tracking worker, the manager can simply pass out the label (or group of labels) to the nodes.

While there are as many candidates as the total degree, the number of start solutions (and the corresponding generalized Bézout number) is typically much less than the total degree. For efficiency – as the sequential root counting procedures in PHCpack already do – an incremental LU factorization of the coefficient matrices for each linear system leading to a start solution is

an effective technique to prune the tree of all possible combinations of factors in the products of  $G$ .

### 3.3 A Numerically Stable Solver for “Simplex” Systems

Homotopies implementing Bernshtein’s theorem [3] were described in [75]. What we now call *polyhedral homotopies* follows the more general treatment in [29]. In [43] these methods are explained in greater details. Bernshtein showed in [3] that the mixed volume of the Newton polytopes of the polynomial system bound the number of solutions (with all variables different from zero). For systems with randomly chosen coefficients, this bound is sharp. The first stage of a polyhedral homotopy method consists in the calculation of this mixed volume, see [25] and [22] for efficient programs to perform this task.

Polyhedral homotopies require in their second stage the solution of a polynomial system with random coefficients. Choosing all complex coefficients on the unit circle in the complex plane, naturally leads to a well-conditioned polynomial system. Despite this good choice of the coefficients, previous versions of our software failed for some large examples used for testing the parallel polyhedral homotopies [77].

Now we will explain what is the *binomial system*. Every equation in a binomial system has exactly two monomials with nonzero coefficients. In compact form, we denote a binomial system by  $\mathbf{x}^A = \mathbf{b}$ . The definition is as follows,

**Definition 3.3.1** A matrix  $A$  with integer coefficients and a vector  $\mathbf{b} \in (\mathbb{C}^*)^n$  define a *binomial system*, denoted as  $\mathbf{x}^A = \mathbf{b}$ . The columns of  $A$  define the exponent vectors of the equations in the binomial system, i.e.: for  $A = [\mathbf{a}_1 \mathbf{a}_2 \cdots \mathbf{a}_n]$ , we have

$$\begin{aligned} \mathbf{x}^A &= \mathbf{x}^{[\mathbf{a}_1 \ \mathbf{a}_2 \ \cdots \ \mathbf{a}_n]} = [\mathbf{x}^{\mathbf{a}_1} \ \mathbf{x}^{\mathbf{a}_2} \ \cdots \ \mathbf{x}^{\mathbf{a}_n}] \\ \mathbf{x}^A = \mathbf{b} &\Leftrightarrow [\mathbf{x}^{\mathbf{a}_1} = b_1 \ \mathbf{x}^{\mathbf{a}_2} = b_2 \ \cdots \ \mathbf{x}^{\mathbf{a}_n} = b_n] \end{aligned} \quad (3.5)$$

We solve the binomial system via the Hermite normal form of  $A$ , computing a unimodular matrix  $M$  with  $\det(M) = \pm 1$ , so that  $U$  is an upper triangular matrix with

$$MA = U, \quad |\det(U)| = |\det(MA)| = |\det(A)| \quad (3.6)$$

The unimodular transformation  $M$  defines a change of coordinates:

$$\mathbf{x} = \mathbf{z}^M \quad \Rightarrow \quad \mathbf{x}^A = \mathbf{z}^{MA} = \mathbf{z}^U \quad (3.7)$$

So we have reduced

$$\mathbf{x}^A = \mathbf{b} \quad \Rightarrow \quad \mathbf{z}^U = \mathbf{b} \quad (3.8)$$

For two variables, we write

$$[z_1 \ z_2] \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix} = [b_1 \ b_2]. \quad \begin{cases} z_1^{u_{11}} & = b_1 \\ z_1^{u_{12}} z_2^{u_{22}} & = b_2 \end{cases} \quad (3.9)$$

Forward substitution on the triangular system shows that there are exactly  $|\det(A)|$  distinct isolated solutions, and  $|b_k| = 1$  implies  $z_k = 1$ , for every solution component,  $k = 1, 2, \dots, n$ . So our binomial systems are numerically very well conditioned.

**Example 3.3.1** Consider the following binomial system:

$$\begin{cases} x_1^3 x_2^2 = 1 \\ x_1^5 x_2^1 = 1 \end{cases} \Leftrightarrow \mathbf{x}^A = \mathbf{b} : [x_1 \ x_2] \begin{bmatrix} 3 & 5 \\ 2 & 1 \end{bmatrix} = [1 \ 1]. \quad (3.10)$$

To make  $A$  upper triangular, we define a unimodular matrix  $M$  taking the greatest common divisor of 3 and 2:  $\gcd(3, 2) = 1 = (+1).3 + (-1).2$ . To eliminate the 2 in  $A$ , the second row in  $M$  will contain the coefficients of the linear combination of 3 and 2:  $(+2).3 + (-3).2 = 0$ . So we have

$$\begin{aligned} M &= \begin{bmatrix} 1 & -1 \\ -2 & 3 \end{bmatrix}, \quad \det(M) = 1 \\ MA &= \begin{bmatrix} 1 & -1 \\ -2 & 3 \end{bmatrix} \begin{bmatrix} 3 & 5 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 4 \\ 0 & -7 \end{bmatrix} = U \end{aligned} \quad (3.11)$$

The coordinate transformation is

$$\mathbf{x} = \mathbf{z}^M \Rightarrow [x_1 \ x_2] = [z_1 \ z_2] \begin{bmatrix} 1 & -1 \\ -2 & 3 \end{bmatrix} = [x_1 = z_1 z_2^{-2} \quad x_2 = z_1^{-1} z_2^3] \quad (3.12)$$

While doing the coordinate transformation on the system, we recognize the matrix multiplication  $MA$ :

$$\begin{cases} x_1^3 x_2^2 = (z_1 z_2^{-2})^3 (z_1^{-1} z_2^3)^2 = z_1^{1.3+(-2).3} z_2^{(-2).3+3.2} = z_1 \\ x_1^5 x_2^1 = (z_1 z_2^{-2})^5 (z_1^{-1} z_2^3)^1 = z_1^{1.5+(-1).3} z_2^{(-2).5+3.1} = z_1^4 z_2^{-7} \end{cases} \quad (3.13)$$

**Example 3.3.2** Consider for example the 12-dimensional polynomial system below in (Equation 3.14). It occurs as just one of the 11,417 start systems generated by polyhedral homotopies to create a random coefficient start system occurring in the design of a robot (see [67], [70], [69], [66]).

$$\begin{cases} b_1 x_5 x_8 + b_2 x_6 x_9 = 0 \\ b_3 x_2^2 + b_4 = 0 \\ b_5 x_1 x_4 + b_6 x_2 x_5 = 0 \\ c_1^{(k)} x_1 x_4 x_7 x_{12} + c_2^{(k)} x_1 x_6 x_{10}^2 + c_3^{(k)} x_2 x_4 x_8 x_{10} + c_4^{(k)} x_2 x_4 x_{11}^2 \\ + c_5^{(k)} x_2 x_6 x_8 x_{11} + c_6^{(k)} x_3 x_4 x_9 x_{10} + c_7^{(k)} x_4^2 x_{12}^2 + c_8^{(k)} x_3 x_6 \\ + c_9^{(k)} x_4^2 + c_{10}^{(k)} x_9 = 0, \quad k = 1, 2, \dots, 9 \end{cases} \quad (3.14)$$

The coefficients  $b_i$ ,  $i = 1, 2, \dots, 6$ , and  $c_j^{(k)}$ ,  $j = 1, 2, \dots, 10$ ,  $k = 1, 2, \dots, 9$  are random chosen complex numbers, where  $\text{t } |b_i| = 1$  and  $|c_j^{(k)}| = 1$ . Because of this good choice of coefficients, all solutions are well conditioned. Despite the high degrees, there are only one hundred isolated solutions, because of the sparsity of the system: only 13 distinct monomials after appropriate division. We call such a system *simplex system*.

**Definition 3.3.2** A *simplex system* is denoted by  $C\mathbf{x}^A = \mathbf{b}$ , where  $C$  is some coefficient matrix.

The natural approach to reduce this simplex system to a binomial system is via a LU-factorization on  $C$ . Assuming  $\det(C) \neq 0$ , we compute a lower and an upper triangular matrix  $L$  and  $U$  so that  $C = LU$  and solve two systems:

$$\begin{aligned} (1) \quad LU\mathbf{y} &= \mathbf{b} && \text{linear system} \\ (2) \quad \mathbf{x}^A &= \mathbf{y} && \text{binomial system} \end{aligned} \tag{3.15}$$

However, this is a numerically unstable algorithm! Even if all coefficients for  $C$  and  $\mathbf{b}$  are chosen to lie on the complex unit circle, varying magnitudes in the intermediate values for  $\mathbf{y}$  do occur. High powers in the range of 50 and over occur in the Hermite normal form for larger systems and magnify the imbalance between the magnitudes in  $\mathbf{y}$  up to the point where numerical underflow or overflow crashes the solver.

For Example 3.3.2, after appropriate division, we can get the coefficient matrix C

$$C = \begin{bmatrix} b_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & b_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & b_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_1^1 & c_2^1 & c_3^1 & c_4^1 & c_5^1 & c_6^1 & c_7^1 & c_8^1 & c_9^1 \\ 0 & 0 & 0 & c_1^2 & c_2^2 & c_3^2 & c_4^2 & c_5^2 & c_6^2 & c_7^2 & c_8^2 & c_9^2 \\ 0 & 0 & 0 & c_1^3 & c_2^3 & c_3^3 & c_4^3 & c_5^3 & c_6^3 & c_7^3 & c_8^3 & c_9^3 \\ 0 & 0 & 0 & c_1^4 & c_2^4 & c_3^4 & c_4^4 & c_5^4 & c_6^4 & c_7^4 & c_8^4 & c_9^4 \\ 0 & 0 & 0 & c_1^5 & c_2^5 & c_3^5 & c_4^5 & c_5^5 & c_6^5 & c_7^5 & c_8^5 & c_9^5 \\ 0 & 0 & 0 & c_1^6 & c_2^6 & c_3^6 & c_4^6 & c_5^6 & c_6^6 & c_7^6 & c_8^6 & c_9^6 \\ 0 & 0 & 0 & c_1^7 & c_2^7 & c_3^7 & c_4^7 & c_5^7 & c_6^7 & c_7^7 & c_8^7 & c_9^7 \\ 0 & 0 & 0 & c_1^8 & c_2^8 & c_3^8 & c_4^8 & c_5^8 & c_6^8 & c_7^8 & c_8^8 & c_9^8 \\ 0 & 0 & 0 & c_1^9 & c_2^9 & c_3^9 & c_4^9 & c_5^9 & c_6^9 & c_7^9 & c_8^9 & c_9^9 \end{bmatrix} \quad (3.16)$$

Given the order of  $\mathbf{x} = (x_5, x_8, x_6, x_9, x_2, x_1, x_4, x_{11}, x_{10}, x_3, x_7, x_{12})$ , the exponent matrix  $A$  is as follows. The first column of  $A$  corresponds to the first binomial equation  $b_1 x_5 x_8 x_6^{-1} x_9^{-1} = -b_2$ .

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 \\ 0 & 2 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 1 & 1 & 1 & 2 & 0 & 2 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 & 0 \end{bmatrix} \quad (3.17)$$

The matrix  $\mathbf{b}$  is

$$\mathbf{b} = [-b_2, -b_4, -b_6, -c_{10}^1, -c_{10}^2, -c_{10}^3, -c_{10}^4, -c_{10}^5, -c_{10}^6, -c_{10}^7, -c_{10}^8, -c_{10}^9]^T$$

Via the Hermite Normal Form to A, we get the unimodular transformation M

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -2 & 0 & -1 & -1 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 1 & 2 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & -1 & -2 & 0 & -1 & -1 & 0 & 0 & 1 & 0 & 0 \\ 7 & 2 & 3 & 6 & 0 & 3 & 4 & 1 & 0 & -4 & 0 & 0 \\ 13 & 5 & 5 & 13 & 0 & 6 & 7 & 3 & 1 & -8 & 0 & 0 \\ 11 & -2 & 7 & 2 & 0 & 3 & 8 & -3 & -4 & -4 & -2 & -7 \\ -42 & 6 & -26 & -10 & 0 & -12 & -30 & 10 & 14 & 16 & 7 & 25 \end{bmatrix} \quad (3.18)$$

The upper triangular matrix  $U$

$$U = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 2 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 & 0 & 1 & -1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & -7 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -16 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 14 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -50 \end{bmatrix} \quad (3.19)$$

$|\det U| = |\det A| = 100$ . High power like -50 occurs in the diagonal of the matrix  $U$ , and the intermediate values  $\mathbf{y}$  are not well conditioned, therefore the numerical instability will happen.

Our new solver separates the magnitudes of the solutions from their angles. Using the notations  $\mathbf{z} = |\mathbf{z}|\mathbf{e}_{\mathbf{z}}$ ,  $\mathbf{e}_{\mathbf{z}} = \exp(i\theta_{\mathbf{z}})$ ,  $\mathbf{y} = |\mathbf{y}|\mathbf{e}_{\mathbf{y}}$ ,  $\mathbf{e}_{\mathbf{y}} = \exp(i\theta_{\mathbf{y}})$ ,  $i = \sqrt{-1}$ , we rewrite the binomial system and solve

$$\mathbf{z}^U = \mathbf{y} : |\mathbf{z}|^U \mathbf{e}_{\mathbf{z}}^U = |\mathbf{y}|\mathbf{e}_{\mathbf{y}} \Leftrightarrow \begin{cases} \mathbf{e}_{\mathbf{z}}^U = \mathbf{e}_{\mathbf{y}} \\ |\mathbf{z}|^U = |\mathbf{y}| \end{cases} \quad (3.20)$$

The first binomial system  $\mathbf{e}_z^U = \mathbf{e}_y$  is well conditioned because all components of the right hand side vector have modulus one. To find the magnitudes  $|\mathbf{z}|$  we solve  $|\mathbf{z}|^U = |\mathbf{y}|$ , using a logarithmic scale, i.e.:  $U \log(|\mathbf{z}|) = \log(|\mathbf{y}|)$ . Even as the magnitude of the values  $\mathbf{y}$  may be extreme,  $\log(|\mathbf{y}|)$  will be modest in size.

Our new numerically stable fewnomial solver to solve  $C\mathbf{x}^A = \mathbf{b}$  executes the following steps:

1. LU factorization of  $C \rightarrow \mathbf{x}^A = \mathbf{y}$ , where  $C\mathbf{y} = \mathbf{b}$ .
2. Use Hermite normal form of  $A$ :  $MA = U$ ,  $\det(M) = \pm 1$ ,  
to solve binomial system  $\mathbf{e}_z^U = \mathbf{e}_y$ ,  $\mathbf{z} = |\mathbf{z}|\mathbf{e}_z$ ,  $\mathbf{y} = |\mathbf{y}|\mathbf{e}_y$ .
3. Solve upper triangular linear system  $U \log(|\mathbf{z}|) = \log(|\mathbf{y}|)$ .
4. Compute the magnitude of  $\mathbf{x} = \mathbf{z}^M$  via  $\log(|\mathbf{x}|) = M \log(|\mathbf{z}|)$ .
5. As  $|\mathbf{e}_z| = 1$ , let  $\mathbf{e}_x = \mathbf{e}_z^M$ .

Even as  $\mathbf{z}$  may be extreme, we deal with  $|\mathbf{z}|$  at a logarithmic scale and never raise small or large number to high powers. Only at the very end we calculate  $|\mathbf{x}| = 10^{\log(|\mathbf{x}|)}$  and  $\mathbf{x} = |\mathbf{x}|\mathbf{e}_x$ .

For more on parallel polyhedral homotopy methods, we refer to [77].

### 3.4 Scanning Solution Files into Frequency Tables

During runtime, we often want to monitor progress of a large path tracking job, and get an impression about the “quality” of the solutions which have been already computed, but again, we do not want store all solutions in the the main memory. For each solution at the end of a path, Newton’s method reports three floating-point numbers:

1. the magnitude of the last update to the solution vector;

2. an estimate for the inverse condition number of the Jacobian matrix at the solution;
3. the magnitude of the residual.

These three numbers determine the quality of a solution.

To determine the overall quality of the list of solutions, the program builds frequency tables, e.g.: counting #solutions with condition number between  $10^{k-1}$  and  $10^k$ , for some range of  $k$ . These frequency tables used to judge the quality of solution lists are a first step to employ the so-called endgames, eventually with some reruns of the paths at tighter tolerances. We refer to [62, Chapter 10] for an overview of endgames.

As the quality analysis of solution lists can already be done while the lists are still incomplete, remedial action or more computationally demanding endgames will lead to extra jobs to be distributed among the worker nodes.

### **3.5 Towards High Performance Continuation ...**

The polynomial systems we typically consider have a number  $n$  of variables which is relatively modest, averaging around 8 or 10. The nonlinearity results in a large number of solution paths, which we denote by  $R$  for the root count used in the homotopy. In this chapter we considered  $R$  in the range of 100,000 and higher. Because  $R \gg n$ , several issues must be addressed to improve the performance of parallel homotopies. In particular, we avoid storing all start solutions in the main memory by jumpstarting the homotopies. We discovered a numerical instability in the polyhedral homotopies which was not treated before and emphasized the need for fast quality control of large solution lists.

The “parallel PHCpack” effort has led to good speedups of running times on existing benchmark systems, essentially leaving the basic path tracking facilities and homotopy constructors intact, calling the routines in PHCpack in conjunction with message passing primitives. To solve polynomial systems which are too large to handle by one single computer, an internal reorganization of PHCpack is needed, in an effort to turn Polynomial Homotopy Continuation into High Performance Continuation.

## CHAPTER 4

### SOFTWARE AND APPLICATIONS

PHCpack [71] implements homotopy continuation methods to compute numerical approximations to all isolated solutions of a system of  $n$  polynomial equations in  $n$  unknowns.

The central concept in polynomial homotopy continuation is the root count because it determines the number of solution paths that need to be traced. Recent research has striven to develop sharp root counts that lead to homotopies with an optimal number of paths. The root count is also a vital instrument in validation numerical results. The computation of a root count is identified with the resolution of a generic system. In this sense, we call root-counting a symbolic computation mirroring this resolution.

For sparse polynomial systems, in general, we apply Bernshtein theorem [3] and count the number of roots by the mixed volume of the Newton polytopes. To compute the mixed volume, we can use four different lifting methods: implicit, static, dynamic, and symmetric lifting. Implicit lifting refers to the algorithms used in the proof of David Bernshtein [3]. The method in Huber and Sturmfels 1995 [29] is called static, to make the distinction with dynamic lifting, an algorithm that has been developed in Verschelde, Gatermann, and Cools 1996 [74] to construct regular triangulations of polytopes incrementally with low lifting values. Symmetric lifting was presented in Verschelde and Gatermann 1995 [73]. To construct regular subdivisions, both integer and floating-point lifting functions are available in PHCpack and elaborated with recursion. The Cayley trick [23] defines in its polyhedral version a polytope whose volume

equals the mixed volume of the considered configuration of polytopes. In PHCpack, this trick is implemented by means of dynamic lifting. When many polynomials share the same exponents, this method is more efficient than static lifting.

In this chapter, we will introduce the MixedVol [22] software package and explain how to integrate it into our PHCpack. We will also introduce a C library interface to PHCpack in Section 4.2.

#### 4.1 Integrate MixedVol into PHCpack

Our motivation is to apply polyhedral homotopies to large “sparse” polynomial systems arising in mechanism design. We already know that the first step for polyhedral homotopy methods is to compute the mixed volume of the polynomial system. For these large polynomial systems, how to compute the mixed volume fast is very important.

MixedVol [22] is a software package for mixed volume computation developed by T.Gao, T.Y.Li and M.Wu. It takes the special structure of the semimixed supports of the systems into account. By numerical results, it can dramatically speed up the mixed volume computation, especially when the systems are unmixed. Even when applied to fully mixed systems, it is also very efficient. MixedVol is written in C++. We want to integrate it into PHCpack.

PHCpack is written in Ada and it can call C functions by building a portable interface to the Ada routines in PHCpack with C functions because the language Ada has the *pragma import* construction to call routines from other languages. Since MixedVol is written in C++ and PHCpack is not compatible with C++, therefore we translate MixedVol from C++ code

to C code first, then we construct an interface between Ada and MixedVol C code, finally the translated Ada code was finished.

Now let's see how to call this translated Ada code to compute the mixed volume in PHCpack.

We will use an example, cyclic 4-roots problem to show how it works.

phc -m

Is the system on a file ? (y/n/i=info) **y**

Reading the name of the input file.

Give a string of characters : **cyclic4**

Reading the name of the output file.

Give a string of characters : **cyclic4.out**

MENU with available Lifting Strategies (0 is default) :

- 0. Static lifting : lift points and prune lower hull.
- 1. Implicit lifting : based on recursive formula.
- 2. Dynamic lifting : incrementally add the points.
- 3. Symmetric lifting : points in same orbit get same lifting.
- 4. MixedVol Algorithm : a faster mixed volume computation.

Type 0, 1, 2, 3, or 4 to select, eventually preceded by i for info: **4**

Do you wish the mixed-cell configuration on separate file ? (y/n) **y**

Reading the name of the file to write the cells ...

Give a string of characters : **cyclic4.fsub**

The input file cyclic4 and mixed-cell configuration file cyclic4.fsub is in Appendix A and B.

## 4.2 A C Interface to PHCpack

PHCpack is written in Ada, while the parallel program to solve start system  $G(\mathbf{x}) = \mathbf{0}$  of the given target system  $F(\mathbf{x}) = \mathbf{0}$  is written in C and MPI. MPI is a library specification for message-passing, proposed as a standard by a broadly based committee of vendors, implementors, and users, it is used for parallel programming. In the C function, one can call functions of the MPI library by including the head file “mpi.h”.

So the main problem now is how to connect C functions with PHCpack. One way is to use **pragma import** and **pragma export** statements which can associate the PHCpack function name with C function name and support conversions for C integers, doubles and arrays of these C types. In PHCpack, if we want to call a function from C, we can build a portable interface in PHCpack for this C function.

An example code fragment of an Ada call to a C function follows:

### **ADA SIDE:**

```
function CPROC(input : in integer) return integer;

pragma import(C, CPROC, "cproc");

result : integer;

...

result:=CPROC(123);  – Put answer in variable “result”
```

### **C SIDE:**

```
int cproc(int input)
{
```

```

        int tmp;          // Accumulate the result here.
        ...              // Do some processing here.
    return tmp;         // Return the result.
}

```

This type of interface is efficient. C program prepares input, then calls Ada program. After computations, Call C program to process results. This type of interface requires either a programmer knowledgeable of both C and Ada, or two programmers collaborating with each other.

For a programmer not familiar with Ada, we want to find a third interface to implement parallel programs using MPI in a good way. We construct an Ada procedure to operate as a gateway interface to PHCpack. In our program, we create an Ada procedure named “**use\_c2phc**” which consists of bundles containers and `c_to_phcpack` as follows:

1. `use_syscon` : Ada gateway to the systems container. We can read and write polynomial system and put in container, get the dimension of the polynomial system, return the number of terms in the  $i$ -th polynomial, return the coefficient (real and imaginary part) and the exponent vector, and so on.
2. `use_solcon` : Ada gateway to the solutions container. We can read solutions from file and put in container, write solutions in the container, return the dimension of the solution vectors, return the multiplicity  $m$  of the solution and the solution as an array, appends the solution with data to the container, and so on.

3. `use_celcon` : Ada gateway to the cells container. We can read mixed-cell configuration from file and initializes the container, writes the mixed-cell configuration in the container, returns the number of cells in the container, returns the dimension of the lifted points in the container, returns the number of different supports and the number of occurrences of support, returns the inner normal for the cell, returns the number of points of each support in the cell, and so on.
4. `c_to_phcpack` : Ada routine as gateway to the machine. We can read and write target polynomial system, read and write start polynomial system, write start solutions, solve by homotopy continuation, write the target solutions, define the output file, and so on.

After constructing `use_c2phc`, the gateway interface to PHCPack, then we will construct a C library on top of it by calling the Ada procedure in it.

Our main parallel programs deliberately are written in C, using MPI. We call library functions by add the gateway routine below to the beginning of the program.

```
extern void adainit( void );
extern int _ada_use_c2phc( int task, int *a, int *b, double *c );
extern void adafinal ( void );
```

(4.1)

The C interface uses PHCPack as a state machine:

1. Feed data into machine and select methods.
2. Compute with given data and selected methods.

3. Extract the results from the machine.

The C user is unaware of the data structure and algorithms.

Given the mixed-cell configuration, we implement the parallel homotopy algorithm to solve polynomial start system by two versions. One is static workload distribution, implemented by `mpi2cell_s.c`. Another is dynamic workload distribution, implemented by `mpi2cell_d.c`. These two programs are coded by calling the C library constructed by above explanation. They are available in version 2.3 of PHCpack. The source code and executable files for this release are available at <http://www.math.uic.edu/~jan/download.html>.

We developed and tested our software using our small cluster with 14 processors. For example, if we want to solve the start system of the cyclic4-roots problem, the input will be mixed-cell configuration (Appendix B) and target system (Appendix A) of cyclic4, the output will be the start solutions of the cyclic4. See Appendix C for command process.

### **4.3 Applications**

The cyclic n-roots problem defines a family of polynomial systems widely used for benchmarking. In Chapter 2, we run our software on the cyclic n-roots problem with our small cluster. We can see that even for our relatively small cluster, larger problem with dimension 12 can be solved well. We ran our software on the mechanism design problem, although these polynomial systems are not so sparse, we can still solve them with our dynamic workload distribution algorithm.

#### 4.4 Conclusions and future work

Our software is an extension of PHCpack. We parallelize the second stage in PHCpack. Our main program is written in C, use the MPI library, and call the path tracking and homotopy facilities in PHCpack. Our motivation is to solve large polynomial systems from mechanism design. We use a static workload distribution algorithm and achieve a good speed up on the cyclic n-roots benchmark systems. Dynamic workload balancing leads to reduced wall times on large polynomial systems which arise in mechanism design.

We also resolve some issues. We use jumpstart homotopies to avoid the storage of all start solutions in main memory. We developed a numerically stable solver for the sparsest class of polynomial systems.

In the thesis of Eric Lee [36], he describes the 3R 5P problem, the geometric design problem of the 3R serial-link robot manipulators with five precision specified. In this problem, five spatial positions and orientations are defined and the dimensions of the geometric parameters of the 3-R manipulator are computed so that the manipulator will be able to place its end-effector at these pre-specified locations. Eric Lee solves this problem by using an interval analysis method.

## APPENDICES

## Appendix A

### CYCLIC 4-ROOTS PROBLEM:

Polynomial system: cyclic4

4

$$x_1 + x_2 + x_3 + x_4;$$

$$x_1*x_2 + x_2*x_3 + x_3*x_4 + x_4*x_1;$$

$$x_1*x_2*x_3 + x_1*x_2*x_4 + x_1*x_3*x_4 + x_2*x_3*x_4;$$

$$x_1*x_2*x_3*x_4 - 1;$$

## Appendix B

## MIXED-CELL CONFIGURATION FILE: CYCLIC4.FSUB

4

4

1 1 1 1

4

8.30428735739752E-01

1.25589423033218E+00

-2.13765813416692E+00

-1.47776315895736E+00

1.00000000000000E+00

2

0.0 0.0 1.0 0.0 4.84834821701439E+00

0.0 0.0 0.0 1.0 4.18845324180483E+00

2

0.0 1.0 1.0 0.0 1.15463130183268E+00

0.0 0.0 1.0 1.0 3.88828869112222E+00

2

1.0 0.0 1.0 1.0 2.66795040744727E+00

0.0 1.0 1.0 1.0 2.24248491285485E+00

## Appendix B (Continued)

2

1.0 1.0 1.0 1.0 3.43832386771139E+00

0.0 0.0 0.0 0.0 1.90922554065903E+00

0

-7.46860192505113E-02

3.50779475341914E-01

1.04292736902969E-01

-1.90948452004673E+00

1.00000000000000E+00

2

1.0 0.0 0.0 0.0 2.35365474100861E+00

0.0 0.0 0.0 1.0 4.18845324180483E+00

2

1.0 1.0 0.0 0.0 1.33361005798616E+00

0.0 1.0 1.0 0.0 1.15463130183268E+00

2

1.0 0.0 1.0 1.0 2.66795040744727E+00

0.0 1.0 1.0 1.0 2.24248491285485E+00

2

1.0 1.0 1.0 1.0 3.43832386771139E+00

0.0 0.0 0.0 0.0 1.90922554065903E+00

## Appendix B (Continued)

0

-6.21674699066986E-01

1.94158704110495E-01

1.35489086776734E+00

-2.45647319986321E+00

1.00000000000000E+00

2

1.0 0.0 0.0 0.0 2.35365474100861E+00

0.0 0.0 0.0 1.0 4.18845324180483E+00

2

1.0 1.0 0.0 0.0 1.33361005798616E+00

1.0 0.0 0.0 1.0 3.98424196195986E+00

2

1.0 1.0 0.0 1.0 3.82868257110411E+00

1.0 0.0 1.0 1.0 2.66795040744727E+00

2

1.0 1.0 1.0 1.0 3.43832386771139E+00

0.0 0.0 0.0 0.0 1.90922554065903E+00

0

-6.03008615133823E-01

-2.12457541102757E+00

## Appendix B (Continued)

-4.24029858980342E-01

1.62251555808937E+00

1.00000000000000E+00

2

1.0 0.0 0.0 0.0 2.35365474100861E+00

0.0 1.0 0.0 0.0 3.87522153690235E+00

2

1.0 1.0 0.0 0.0 1.33361005798616E+00

0.0 1.0 1.0 0.0 1.15463130183268E+00

2

1.0 1.0 1.0 0.0 4.46800908607804E+00

0.0 1.0 1.0 1.0 2.24248491285485E+00

2

1.0 1.0 1.0 1.0 3.43832386771139E+00

0.0 0.0 0.0 0.0 1.90922554065903E+00

## Appendix C

### COMMAND PROCESS FOR RUNNING MPI2CELL\_D.C ON CYCLIC4-ROOTS PROBLEM

```
[yan@idefix bin]$ mpirun -np 4 mpi2cell_d
```

```
Reading the file name for a mixed-cell configuration.
```

```
Give a string of characters : cyclic4.fsub
```

```
Reading a system to initialize the symbol table...
```

```
Reading the target system...
```

```
Give a string of characters : cyclic4
```

```
Reading the name of the output file...
```

```
Give a string of characters: cyclic4.out
```

```
The total #solutions is 16.
```

```
writing random coefficient system and its solutions to file
```

## CITED LITERATURE

1. Allgower, E. L. and Georg, K.: Introduction to Numerical Continuation Methods, volume 45 of Classics in Applied Mathematics. SIAM, 2003.
2. Allison, D. C. S., Chakraborty, A., and Watson, L. T.: Granularity issues for solving polynomial systems via globally convergent algorithms on a hypercube. J. of Supercomputing, 3:5–20, 1989.
3. Bernshtein, D.: The number of roots of a system of equations. Functional Anal. Appl., 9(3):183–185, 1975. Translated from *Functional. Anal. i Prilozhen.*, 9(3):1–4, 1975.
4. Björck, G.: Functions of modulus one on  $Z_p$  whose Fourier transforms have constant modulus. In Proceedings of the Alfred Haar Memorial Conference, Budapest, eds, J. Szabados and K. Tandori, volume 49 of Colloquia Mathematica Societatis János Bolyai, pages 193–197. North Holland, 1985.
5. Björck, G.: Functions of modulus one on  $Z_n$  whose Fourier transforms have constant modulus, and “cyclic n-roots”. In Recent Advances in Fourier Analysis and its Applications, eds, J. Byrnes and J. Byrnes, volume 315 of NATO Adv. Sci. Inst. Ser. C: Math. Phys. Sci., pages 131–140. Kluwer, 1989.
6. Björck, G. and Fröberg, R.: 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.
7. Björck, G. and Fröberg, R.: Methods to “divide out” certain solutions from systems of algebraic equations, applied to find all cyclic 8-roots. In Analysis, Algebra and Computers in Math. research, eds, M. Gyllenberg and L. Persson, volume 564 of Lecture Notes in Mathematics, pages 57–70. Dekker, 1994.
8. Boege, W., Gebauer, R., and Kredel, H.: Some examples for solving systems of algebraic equations by calculating groebner bases. J. Symbolic Computation, 2:83–98, 1986.
9. Bonnesen, T. and Fenchel, W.: Theorie der konvexen körper. Springer-Verlag, Berlin Heidelberg New York, 1934.

10. Canny, J. and Rojas, J. M.: 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 (ISSAC 1991), ed. S. M. Watt, pages 96–101. ACM, 1991.
11. Chakraborty, A., Allison, D. C. S., Ribbens, C. J., and Watson, L. T.: Note on unit tangent vector computation for homotopy curve tracking on a hypercube. Parallel Computing, 17(12):1385–1395, 1991.
12. Chakraborty, A., Allison, D. C. S., Ribbens, C. J., and Watson, L. T.: The parallel complexity of embedding algorithms for the solution of systems of nonlinear equations. IEEE Transactions on Parallel and Distributed Systems, 4(4):458–465, 1993.
13. Cox, D., Little, J., and O’shea, D.: Using algebraic geometry. Volume 185 of Graduate Texts in Mathematics. Springer-Verlag, New York.
14. Dai, Y., Kim, S., and Kojima, M.: Computing all nonsingular solutions of cyclic- $n$  polynomial using polyhedral homotopy continuation methods. J. Comput. Appl. Math., 152(1-2):83–97, 2003.
15. Davenport, J. H.: Looking at a set of equations. Technical report 87-06, School of Mathematical Sciences, University of Bath, 1987. Available at <http://www.bath.ac.uk/~masjhd/>.
16. Emiris, I. Z.: Symbolic-numeric algebra for polynomials. In A. Kent and J. Williams Eds., Encyclopedia of Computer Science and Technology, 39:261-281. Marcel Dekker.
17. Emiris, I. Z. and Canny, J. F.: Efficient incremental algorithms for the sparse resultant and the mixed volume. J. Symbolic Computation, 20(2):117–149, 1995.
18. Faugère, J. C.: A new efficient algorithm for computing Gröbner bases ( $F_4$ ). Journal of Pure and Applied Algebra, 139(1-3):61–88, 1999. Proceedings of MEGA’98, 22–27 June 1998, Saint-Malo, France.
19. Gao, T. and Li, T. Y.: Mixed volume computation via linear programming. Taiwan J. of Math., 4:599–619, 2000.
20. Gao, T. and Li, T. Y.: Mixed volume computation for semi-mixed systems. Discrete Comput. Geom., 29(2):257–277, 2003.

21. Gao, T., Li, T. Y., Verschelde, J., and Wu, M.: Balancing the lifting values to improve the numerical stability of polyhedral homotopy continuation methods. Appl. Math. Comput., 114:233–247, 2000.
22. Gao, T., Li, T. Y., and Wu, M.: Algorithm 846: MixedVol: A software package for mixed volume computation. ACM Trans. Math. Softw., 31(4):555–560, 2005.
23. Gel'fand, I. M., Kapranov, M. M., and Zelevinsky, A. V.: Discriminants, Resultants and Multidimensional Determinants. Boston, Birkhäuser, 1994.
24. Gunji, T., Kim, S., Fujisawa, K., and Kojima, M.: PHoMpara – parallel implementation of the Polyhedral Homotopy continuation Method for polynomial systems. Computing, 77(4):387–411, 2006.
25. Gunji, T., Kim, S., Kojima, M., Takeda, A., Fujisawa, K., and Mizutani, T.: PHoM – a polyhedral homotopy continuation method for polynomial systems. Computing, 73(4):55–77, 2004.
26. Harimoto, S. and Watson, L. T.: The granularity of homotopy algorithms for polynomial systems of equations. In Parallel processing for scientific computing, ed. G. Rodrigue, pages 115–120. SIAM, 1989.
27. Huber, B., Rambau, J., and Santos, F.: The Cayley trick, lifting subdivisions and the Bohne-Dress theorem on zonotopal tilings. J. Eur. Math. Soc., 2(2):179–198, 2000.
28. Huber, B., Sottile, F., and Sturmfels, B.: Numerical Schubert calculus. J. of Symbolic Computation, 26(6):767–788, 1998.
29. Huber, B. and Sturmfels, B.: A polyhedral method for solving sparse polynomial systems. Math. Comp., 64(212):1541–1555, 1995.
30. Huber, B. and Verschelde, J.: Pieri homotopies for problems in enumerative geometry applied to pole placement in linear systems control. SIAM J. Control Optim., 38(4):1265–1287, 2000.
31. Katsura, S.: Users posing problems to posso. In the PoSSO Newsletter, no. 2, July 1994, eds, L. Gonzelez-Vega and T. Recio.

32. Khovanskii, A. G.: Newton polyhedra and the genus of complete intersections. Functional Anal. Appl., 12(1):38–46, 1978. Translated from *Functional. Anal. i Prilozhen.*, 12(1),51–61,1978.
33. Kim, S. and Kojima, M.: Numerical stability of path tracing in polynomial homotopy continuation methods. Computing, 73:329–348, 2004.
34. Kushnirenko, A. G.: Newton Polytopes and the Bézout Theorem. Functional Anal. Appl., 10(3):233–235, 1976. Translated from *Functional. Anal. i Prilozhen.*, 10(3),82–83,1976.
35. Lee, C. W.: Regular triangulations for convex polytopes. In Applied Geometry and Discrete Mathematics - The Victor Klee Festschrift, eds, P. Gritzmann and B. Sturmfels, volume 4 of DIMACS series, pages 443–456. AMS, Providence, R. I., 1991.
36. Lee, E.: Computational geometric design of spatial robot manipulators. PhD thesis, the State University of New Jersey, Rutgers, 2004.
37. Leykin, A. and Verschelde, J.: Decomposing solution sets of polynomial systems: a new parallel monodromy breakup algorithm. Accepted for publication in *The International Journal of Computational Science and Engineering* (special issue dedicated to HPSEC'05).
38. Leykin, A. and Verschelde, J.: PHCmaple: A Maple interface to the numerical homotopy algorithms in PHCpack. In Proceedings of the Tenth International Conference on Applications of Computer Algebra (ACA'2004), ed. Q.-N. Tran, pages 139–147, 2004.
39. Leykin, A. and Verschelde, J.: Factoring solution sets of polynomial systems in parallel. In Proceedings of the 2005 International Conference on Parallel Processing Workshops. 14-17 June 2005. Oslo, Norway. High Performance Scientific and Engineering Computing, eds, T. Skeie and C.-S. Yang, pages 173–180. IEEE Computer Society, 2005.
40. Leykin, A., Verschelde, J., and Zhuang, Y.: Parallel homotopy algorithms to solve polynomial systems. In Proceedings of ICMS 2006, LNCS 4151, eds, A. Iglesias and N. Takayama, pages 225–234. Springer-Verlag, 2006.
41. Li, T. Y.: Solving polynomial systems. The Mathematical Intelligencer, 9(3):33–39, 1987.

42. Li, T. Y.: Numerical solution of multivariate polynomial systems by homotopy continuation methods. Acta Numerica, 6:399–436, 1997.
43. Li, T. Y.: Numerical solution of polynomial systems by homotopy continuation methods. In Handbook of Numerical Analysis. Volume XI. Special Volume: Foundations of Computational Mathematics, ed. F. Cucker, pages 209–304. North-Holland, 2003.
44. Li, T. Y. and Li, X.: Finding mixed cells in the mixed volume computation. Found. Comput. Math., 1(2):161–181, 2001.
45. Li, T. Y., Sauer, T., and Yorke, J. A.: The cheater’s homotopy: an efficient procedure for solving systems of polynomial equations. SIAM J. Numer. Anal., 26(5):1241–1251, 1989.
46. Morgan, A.: Solving polynomial systems using continuation for engineering and scientific problem. Prentice-Hall, 1987.
47. Morgan, A. P. and Sommese, A. J.: A homotopy for solving general polynomial systems that respects m-homogeneous structures. Appl. Math. Comput., 24(2):101–113, 1987.
48. Morgan, A. P. and Sommese, A. J.: Coefficient-parameter polynomial continuation. Appl. Math. Comput., 29(2):123–160, 1989. Errata: *Appl. Math. Comput.* 51:207(1992).
49. Morgan, A. P., Sommese, A. J., and Wampler, C. W.: Computing singular solutions to nonlinear analytic systems. Numer. Math., 58(7):669–684, 1991.
50. Morgan, A. P., Sommese, A. J., and Wampler, C. W.: Computing singular solutions to polynomial systems. Adv. Appl. math., 13(3):305–327, 1992.
51. Morgan, A. P., Sommese, A. J., and Wampler, C. W.: A power series method for computing singular solutions to nonlinear analytic systems. Numer. Math., 63:391–409, 1992.
52. Morgan, A. P., Sommese, A. J., and Watson, L. T.: Finding all isolated solutions to polynomial software using hompack. ACM Trans. Math. Softw., 15(2):93–122, 1989.
53. Morgan, A. P. and Watson, L. T.: A globally convergent parallel algorithm for zeros of polynomial systems. Nonlinear Analysis, 13(11):1339–1350, 1989.
54. Pelz, W. and Watson, L. T.: Message length effects for solving polynomial systems on a hypercube. Parallel Computing, 10(2):161–176, 1989.

55. Rojas, J. M.: A convex geometric approach to counting the roots of a polynomial system. Theoret. Comput. Sci., 133(1):105–140, 1994.
56. Rojas, J. M.: Toric intersection theory for affine root counting. Journal of Pure and Applied Algebra, 136(1):67–100, 1999.
57. Sommese, A. J., Verschelde, J., and Wampler, C. W.: Numerical decomposition of the solution sets of polynomial systems into irreducible components. SIAM J. Numer. Anal., 38(6):2022–2046, 2001.
58. Sommese, A. J., Verschelde, J., and Wampler, C. W.: Using monodromy to decompose solution sets of polynomial systems into irreducible components. In C. Ciliberto, F. Hirzebruch, R. Miranda and M. Teicher, editors, Application of Algebraic Geometry to Coding Theory, Physics and Computation, pages 297–315. Kluwer Academic, 2001. Proceedings of a NATO Conference, February 25–March 1, 2001, Eilat, Israel.
59. Sommese, A. J., Verschelde, J., and Wampler, C. W.: Symmetric functions applied to decomposing solution sets of polynomial systems. SIAM J. Numer. Anal., 40(6):2026–2046, 2002.
60. Sommese, A. J., Verschelde, J., and Wampler, C. W.: Numerical irreducible decomposition using PHCpack. In Algebra, Geometry, and Software Systems, eds, M. Joswig and N. Takayama, pages 109–130. Springer–Verlag, 2003.
61. Sommese, A. J., Verschelde, J., and Wampler, C. W.: Introduction to numerical algebraic geometry. In A. Dickenstein and I.Z. Emiris, editors, Solving polynomial Equations. Foundations, Algorithms and Applications, volume 14 of Algorithms and Computation in Mathematics, pages 301–337. Springer–Verlag, 2005.
62. Sommese, A. J. and Wampler, C. W.: The Numerical solution of systems of polynomials arising in engineering and science. World Scientific, 2005.
63. Sosonkina, M., Watson, L. T., and Stewart, D. E.: Note on the end game in homotopy zero curve tracking. ACM Trans. Math. Softw., 22(3):281–287, 1996.
64. Sturmfels, B.: On the Newton polytope of the resultant. Journal of Algebraic Combinatorics, 3(2):207–236, 1994.

65. Sturmfels, B.: Polynomial equations and convex polytopes. Amer. Math. Monthly, 105:907–922, 1998.
66. Su, H.-J.: Computer-aided constrained robot design using mechanism synthesis theory. PhD thesis, University of California, Irvine, 2004.
67. Su, H.-J. and McCarthy, J. M.: Kinematic synthesis of RPS serial chains. In the Proceedings of the ASME Design Engineering Technical Conferences (CDROM), Chicago, IL, Sep 2-6, 2003.
68. Su, H.-J., McCarthy, J. M., Sosonkina, M., and Watson, L. T.: Algorithm 857: POLSYS\_GLP: A parallel general linear product homotopy code for solving polynomial systems of equations. ACM Trans. Math. Softw., 32(4):561–597, 2006.
69. Su, H.-J., McCarthy, J. M., and Watson, L. T.: Generalized linear product homotopy algorithms and the computation of reachable surfaces. ASME Journal of Information and Computer Sciences in Engineering, 4(3):226–234, 2004.
70. Su, H.-J., Wampler, C. W., and McCarthy, J. M.: Geometric design of cylindric PRS serial chains. ASME Journal of Mechanical Design, 126(2):269–277, 2004.
71. Verschelde, J.: Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. ACM Trans. Math. Softw., 25(2):251–276, 1999. Software available at <http://www.math.uic.edu/~jan>.
72. Verschelde, J. and Cools, R.: Symbolic homotopy construction. Applicable Algebra in Engineering, Communication and Computing, 4(3):169–183, 1993.
73. Verschelde, J. and Gatermann, K.: Symmetric newton polytopes for solving sparse polynomial system. Adv. Appl. Math., 16(1):95–127, 1995.
74. Verschelde, J., Gatermann, K., and Cools, R.: Mixed-volume computation by dynamic lifting applied to polynomial system solving. Discrete Comput. Geom., 16(1):69–112, 1996.
75. Verschelde, J., Verlinden, P., and Cools, R.: Homotopies exploiting Newton polytopes for solving sparse polynomial systems. SIAM J. Numer. Anal., 31(3):915–930, 1994.
76. Verschelde, J. and Wang, Y.: Computing feedback laws for linear systems with a parallel Pieri homotopy. In Proceedings of the 2004 International Conference on Parallel

Processing Workshops, 15-18 August 2004, Montreal, Quebec, Canada. High Performance Scientific and Engineering Computing, ed. Y. Yang, pages 222–229. IEEE Computer Society, 2004.

77. Verschelde, J. and Zhuang, Y.: Parallel implementation of the polyhedral homotopy method. In Proceedings of the 2006 International Conference on Parallel Processing Workshops. 14-18 August 2006. Columbus, Ohio. High Performance Scientific and Engineering Computing., eds, T. Pinkston and F. Ozguner, pages 481–488. IEEE Computer Society, 2006.
78. Wampler, C. W., Morgan, A. P., and Sommese, A. J.: Numerical continuation methods for solving polynomial systems arising in kinematics. SIAM J. of Mechanical Design, 112(1):59–68, 1990.
79. Watson, L. T.: Numerical linear algebra aspects of globally convergent homotopy methods. SIAM Rev., 28(4):529–545, 1986.
80. Watson, L. T.: Probability-one homotopies in computational science. J. Comput. Appl. Math., 140(1&2):785–807, 2002.
81. Watson, L. T., Billups, S. C., and Morgan, A. P.: Algorithm 652: HOMPACT: a suite of codes for globally convergent homotopy algorithms. ACM Trans. Math. Softw., 13(3):281–310, 1987.
82. Watson, L. T., Sosonkina, M., Melville, R. C., Morgan, A. P., and Walker, H. F.: HOMPACT90: A suite of Fortran 90 codes for globally convergent homotopy algorithms. ACM Trans. Math. Softw., 23(4):514–549, 1997.
83. Wise, S. M., Sommese, A. J., and Watson, L. T.: Algorithm 801: POLSYS\_PLP: a partitioned linear product homotopy code for solving polynomial systems of equations. ACM Trans. Math. Softw., 26(1):176–200, 2000.

## VITA

NAME: Yan Zhuang

### EDUCATION:

- \* B.S., Computer Science, Shanghai University, China, 1994
- \* M.S., Mathematical Computer Science, University of Illinois at Chicago, Chicago, Illinois, 2003.
- \* Ph.D., Mathematical Computer Science, University of Illinois at Chicago, Chicago, Illinois, 2007.

### EXPERIENCE:

- \* Research Assistant, Department of Psychiatry, University of Illinois at Chicago, 2002-2003.
- \* Research Assistant, Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago, 2004-2006.
- \* Teaching Assistant, Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago, 2007.

### PUBLICATIONS:

- [1] Parallel Implementation of the Polyhedral Homotopy

Method (with Jan Verschelde). Proceedings of the 2006 International Conference on Parallel Processing Workshops, 14-18 Augustus 2006, Columbus, Ohio. High Performance Scientific and Engineering Computing, pages 481-488, IEEE Computer Society, 2006.

- [2] Parallel Homotopy Algorithms to Solve Polynomial Systems (with Anton Leykin, Jan Verschelde). Proceedings of ICMS 2006, LNCS 4151, pages 225-234, Springer-Verlag, 2006.

#### PRESENTATIONS:

- \* Parallel Implementation of Polyhedral Homotopy Methods,  
AMS National Meetings, January 5-8, 2007, New Orleans, LA.
- \* Parallel Implementation of Polyhedral Homotopy Methods.  
International Conference on Parallel Processing, Columbus, Ohio,  
14-18 August 2006.
- \* Parallel Homotopy Algorithms to Solve Polynomial Systems,  
AMS special session on Numerical Solution of Polynomial Systems,  
University of Notre Dame, 8-9 April 2006.

#### POSTERS:

- \* Parallel Implementation of the Polyhedral Homotopy Method at workshop  
on Software for Algebraic Geometry, Institute for Mathematics and its

Applications, Minnesota, 23-27 October 2006.

- \* IMGEM-Interactive Multiple Gene Expression Maps at Chicago Chapter of the Society for Neuroscience 2003 Scientific Meeting.

PROFESSIONAL MEMBERSHIP:

- \* American Mathematical Society.