This is to certify that the
dissertation entitled

ALGORITHMS FOR SOLVING POLYNOMIAL SYSTEMS
BY HOMOTOPY CONTINUATION METHOD AND ITS
PARALLELIZATION

presented by

Chih-Hsiung Tsai

has been accepted towards fulfillment
of the requirements for the

Ph.D. degree in Mathematics

Tien-Yien Li
Major Professor's Signature

7/31/08
Date

ALGORITHMS FOR SOLVING POLYNOMIAL SYSTEMS BY HOMOTOPY CONTINUATION METHOD AND ITS PARALLELIZATION

By

Chih-Hsiung Tsai

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Mathematics

2008

ABSTRACT

ALGORITHMS FOR SOLVING POLYNOMIAL SYSTEMS BY HOMOTOPY CONTINUATION METHOD AND ITS PARALLELIZATION

By

Chih-Hsiung Tsai

HOM4PS-2.0 is a software package in FORTRAN 90 which implements the polyhedral homotopy continuation method for solving polynomial systems. It updates its original version HOM4PS in three key aspects: (1) New method for finding mixed cells, (2) New idea of following homotopy paths, (3) New way of dealing with the curve jumping.

The parallel version of HOM4PS-2.0, named HOM4PS-2.0para, parallelizes the three main stages in HOM4PS-2.0. Excellent scalability in the numerical results shows that the parallelization of the homotopy method always provides a great amount of extra computing resources to help solve polynomial systems of larger size which would be very difficult to deal with otherwise.

To my parents

ACKNOWLEDGMENTS

I would like to give my immense gratitude to my dissertation advisor, Professor Tien-Yien Li, for his constant encouragement and support during my graduate study at Michigan State University. His knowledge, insights and enthusiasm are invaluable.

I would also like to express my thank to Professor Chichia Chiu, Professor Jeanne Wald, Professor Guowei Wei , and Professor Zhengfang Zhou for their time and valuable advice.

I am grateful to my classmates Jules, Tianran, Alice and Ying for their help and useful comments. Last, but not least, I want to thank Jamie for her support during these years.

TABLE OF CONTENTS

LIST OF TABLES

Chapter 1

Introduction

Let $P(x) = 0$ be a system of n polynomial equations in n unknowns. Denoting $P(x) = (p_1(x), \ldots, p_n(x))$ with $x = (x_1, \ldots, x_n)$, we want to find all isolated solutions of

$$\begin{aligned} p_1(x_1, \ldots, x_n) &= 0, \\ &\vdots \\ p_n(x_1, \ldots, x_n) &= 0, \end{aligned}$$

in $\mathbb{C}^n$.

The problem of solving systems of polynomial equations arises in many scientific and engineering applications. Reliable and efficient numerical approaches to solve this problem have long been desired. One such approach is the homotopy continuation method proposed in 1977 by Garcia and Zangwill [11] and Drexler [7] independently.

Since then, this approach has undergone quite a bit of development. In the beginning, the employment of linear homotopies was standard. Then in 1995, Huber and Sturmfels [15] suggested the use of polyhedral (nonlinear) homotopies based on Bernshtein's combinatorial root count [1] yielding drastic improvements over the *classical linear homotopy method* for solving sparse polynomial systems. Ultimately this is because Bernshtein's root count is a much tighter bound on the number of solutions than the Bézout number. The software package HOM4PS developed by T. Y. Li and T. Gao since 1999 implemented this approach.

Huber and Sturmfels [15] also recommended the use of the combined polyhedral-linear homotopies, but these were impractical due to difficulties with efficiency and stability. In 2007, T. Y. Li and C. H. Tsai, utilizing exponential-logarithmic transformations, were able to successfully implement combined polyhedral-linear homotopies, and, released with T. L. Lee an update to HOM4PS, called HOM4PS-2.0. Its parallel version HOM4PS-2.0para which utilizes multiprocessors soon followed. These latest two packages show considerable speed-up and increased ability to handle larger systems.

This thesis presents the details of the strategies used to implement the homotopy continuation method in the codes HOM4PS-2.0 and HOM4PS-2.0para. A sketch of the theoretical underpinnings and historical development of the homotopy continuation method will be presented. Techniques used to overcome computational difficulties in implementing the polyhedral continuation method will be outlined.

In Chapter 2, we first review the basics of the polyhedral homotopy continuation method; this includes, the definition of the mixed volume of the support of a polynomial system as well as Bernshtein's theorem concerning root counts. Next, a practical method of computing mixed volumes, using the mixed cells of the support, which leads to the construction of the corresponding polyhedral homotopies, will be discussed. The details of finding the mixed cells will be left to Chapter 3.

In Chapter 3, we will present the latest optimizing strategies used in HOM4PS-2.0, which considerably upgrades its original version HOM4PS. There are three major steps in HOM4PS-2.0:

- **Mixed cell computations**
- **Following homotopy paths**
- **Dealing with 'Curve Jumping'.**

Sections 3.1-3.3 take up the treatment of these issues. Other miscellaneous aspects of HOM4Ps-2.0 are listed in Section 3.4. These include evaluating polynomials and derivatives, scaling coefficients, and determining criteria for path convergence and divergence, i.e., the end game. Numerical results of HOM4PS-2.0 are shown and discussed in Section 3.5. These results clearly show that HOM4PS-2.0 is very efficient and powerful.

As the size of the polynomial system becomes larger, more computing resources are required. And a natural way to allocate extra computing resources efficiently is to perform independent tasks simultaneously in parallel. Since each isolated zero is computed independently of all the others in the homotopy continuation method, it provides a natural environment for the parallelization. In Chapter 4, we present a parallel implementation of HOM4PS-2.0, HOM4PS-2.0para, in which each of the three main stages above are parallelized using multiple processors. A simple master-workers parallel computation environment is used, in which one master CPU communicates with all worker CPUs using the Message Passing Interface, MPI [34], to dynamically allocate tasks among the latter. The first four sections of Chapter 4 describe respectively, the parallel implementations of the mixed cells computation, the curve tracings of the polyhedral-linear homotopies, the curve tracings of the optional classical linear homotopy, and the checking for possible curve jumping. In Section 4.5, numerical results of HOM4PS-2.0para on the bench mark systems, eco-n [30], katsura-n [3], noon-n [35], reimer-n [37], and cyclic-n [2], are summarized. The near perfect speed-ups in the results clearly demonstrate that our dynamic allocation algorithms effectively utilize the processing capacities of a parallel architecture to implement the polyhedral-linear homotopy continuation method for solving previously unmanageable polynomial systems of larger size. Conclusions and directions for future work are given in Chapter 5.

Chapter 2

Polyhedral Homotopy Method

For sparse polynomial systems, Bernshtein's combinatorial root count gives a much tighter bound on the number of isolated solutions than the Bézout number. It thus provides a starting point for an alternative approach to the classical linear homotopy continuation method which will reduce the number of homotopy paths to be traced. In this chapter, we first review Bernshtein's theorem on the exact root count in $(\mathbb{C}^*)^n$ for a generic system, where $\mathbb{C}^* = \mathbb{C}\backslash\{0\}$. Next, we discuss an important extension by Li and Wang [26] for the root count in $\mathbb{C}^n$, followed by the description of the actual procedure that uses Bernshtein's root count to find all isolated solutions, i.e., Huber and Sturmfel's polyhedral homotopy method.

2.1 Bernshtein's Theorem

Let $P(x) = (p_1(x), \ldots, p_n(x))$ be a system of n polynomials in n unknowns with $x = (x_1, \ldots, x_n)$, where

$$p_j(x) = \sum_{a \in S_j} c^*_{j,a} x^a, \qquad j = 1, \ldots, n. \tag{2.1}$$

Here $x^a = x_1^{a_1} \cdots x_n^{a_n}$ where $a = (a_1, \ldots, a_n)$ with coefficient $c^*_{j,a} \in \mathbb{C}^* = \mathbb{C} \backslash \{0\}$. The index set S_j, denotes the *support* of $p_j(x)$, is the set of all monomial powers of $p_j(x)$.

The *convex hull* $K_j = conv(S_j)$ in $\mathbb{R}^n$ of S_j is the *Newton polytope* of $p_j(x)$ and $S = (S_1, \ldots, S_n)$ is called the *support* of the system $P(x)$. For non-negative real variables $\lambda_1, \ldots, \lambda_n$, write the *Minkowski sum*

$$\lambda_1 K_1 + \cdots + \lambda_n K_n = \{\lambda_1 \gamma_1 + \cdots + \lambda_n \gamma_n | \gamma_j \in K_j, j = 1, \ldots, n\}.$$

The n-dimensional volume $Vol_n(\lambda_1 K_1 + \cdots + \lambda_n K_n)$ is known to be a homogeneous polynomial of degree n in the variables $\lambda_1, \ldots, \lambda_n$. The coefficient of the monomial $\lambda_1 \lambda_2 \cdots \lambda_n$ in this homogeneous polynomial is called the *mixed volume* of the polytopes $K_1, \ldots, K_n$, denoted by $\mathcal{M}(K_1, \ldots, K_n)$. Alternatively, it is called the mixed volume of the support of the system $P(x) = (p_1(x), \ldots, p_n(x))$, denoted by $\mathcal{M}(S_1, \ldots, S_n)$, or just the mixed volume of $P(x)$ for short.

Now, let the coefficient $c_{j,a}$ of each monomial in the original system $P(x) =$

$(p_1(x), \ldots, p_n(x))$ in (2.1) be considered as a variable in its own right, resulting in the associated system $P(c,x) = (p_1(c,x), \ldots, p_n(c,x))$ where

$$p_j(c,x) = \sum_{a \in S_j} c_{j,a} x^a, \qquad j = 1, \ldots, n. \tag{2.2}$$

The original system $P(x) = P(c^*, x)$ is simply the expanded system $P(c,x)$ evaluated at a set of specified values of coefficients $c^* = (c^*_{j,a})$.

Recall that the root count of a polynomial system is simply the total number of isolated zeros, counting multiplicities.

Lemma 2.1 [14] For polynomial systems $P(c,x)$ in (2.2), there exists a polynomial system $G(c) = (g_1(c), \ldots, g_l(c))$ in the variables $c = (c_{j,a})$ for $a \in S_j$ and $j = 1, \ldots, n$ such that for those coefficients $c^* = (c^*_{j,a})$ for which $G(c^*) \neq 0$, the root count in $(\mathbb{C}^*)^n$ of the corresponding polynomial system in (2.2) is a fixed number. And the root count in $(\mathbb{C}^*)^n$ of any other polynomial system in (2.2) is bounded above by this number.

Remark 2.2 Since the zeros of the polynomial system $G(c)$ in the above lemma form an algebraic set, its complement is open and dense with full measure. Thus, a polynomial system $P(c^*, x)$ in (2.2) with $G(c^*) \neq 0$ is said to be in *general position.*

Theorem 2.3 [1] The root count in $(\mathbb{C}^*)^n$ of a polynomial system $P(x) = (p_1(x), \ldots, p_n(x))$ in general position equals the mixed volume of its support.

The above root count is, in practice, a much tighter bound than the Bézout bound [31] and is also known as the "BKK bound" after its inventors, Bernshtein [1], Kushnirenko [18], and Khovanskii [16]. An extension of the Bernshtein Theorem that counts the roots in $\mathbb{C}^n$ as opposed to $(\mathbb{C}^*)^n$ was provided by Li and Wang [26].

Theorem 2.4 The root count in $\mathbb{C}^n$ of a polynomial system $P(x) = (p_1(x), \ldots, p_n(x))$ with support $S = (S_1, \ldots, S_n)$ is bounded above by the mixed volume $\mathcal{M}(S_1 \bigcup\{\mathbf{0}\}, \ldots, S_n \bigcup\{\mathbf{0}\})$.

In other words, simply augment each polynomial in the original system $P(x)$ by adding a constant term, if one is missing, namely adding the point $\mathbf{0} = (0, \ldots, 0)$ to each support S_j. Then, the mixed volume of the augmented supports, which is the root count in $(\mathbb{C}^*)^n$ of the augmented polynomial system in general position, is the upper bound for the root count in $\mathbb{C}^n$ of the original system $P(x)$. Hence, if the constant term were already present in each polynomial in the original system $P(x) = 0$, i.e., its support $S_j = S_j \bigcup\{\mathbf{0}\}$, then the upper bound for the root count in $(\mathbb{C}^*)^n$ would be the same as that in $\mathbb{C}^n$. Furthermore, for a generic system with these supports (with all constant terms included),

$$\#\text{ of solns in } (\mathbb{C}^*)^n = \mathcal{M}(S_1, \ldots, S_n) = \#\text{ of solns in } \mathbb{C}^n.$$

So no isolated solution of a system in general position in which all constant terms are present has a zero component.

2.2 Polyhedral Homotopy

The earliest form of the homotopy continuation method for solving polynomial system $P(x) = (p_1(x), \ldots, p_n(x)) = 0$ consisted of first finding an easily solved system $Q(x) = (q_1(x), \ldots, q_n(x)) = 0$, next defining a linear homotopy $H : \mathbb{C}^n \times [0,1] \to \mathbb{C}^n$ by

$$H(x,t) = (1-t)Q(x) + tP(x), \tag{2.3}$$

and then numerically following the curves in the real variable t which make up the solution set of $H(x,t) = 0$. Here the homotopy $H(x,t)$ is linear in t. More generally, homotopies may be constructed to be nonlinear in t.

In order for the continuation method to work, the homotopy $H(x,t)$ must be chosen so that the following three properties hold:

- Property 1 (*Triviality*) The solution points of the starting system $H(x,0) = Q(x) = 0$ are known.
- Property 2 (*Smoothness*) The solution set of $H(x,t) = 0$ for $0 \leq t < 1$ consists of a finite number of smooth paths, each parameterized by t in $[0,1)$.
- Property 3 (*Accessibility*) Every isolated solution of the original target system $H(x,1) = P(x) = 0$ can be reached by some path originating at $t = 0$; i.e. at some solution point of the starting system $H(x,0) = Q(x) = 0$.

If all three properties hold, then all solutions of the original problem $P(x) = 0$ can be obtained by tracing the solution paths of $H(x,t) = 0$, each starting from an initial starting point which is a solution of the known system $Q(x) = 0$. Note that while divergence of a path to infinity *in the middle of the interval* (as $t \to t_0 < 1$) is precluded by the smoothness property, it is still consistent with the above properties that a solution path of $H(x,t) = 0$ may diverge to infinity as the parameter t approaches 1. Thus, some of the paths emanating from the roots of $Q(x)$, may be extraneous, and diverge to infinity as $t \to 1$. Since in practice it is not known ahead of time which path is extraneous, each path must be followed until it is clear that it diverges or a solution point of $P(x) = 0$ is reached. Divergent paths thus represent wasted computation effort.

Now we outline Huber and Sturmfel's polyhedral homotopy method to find all isolated solutions in $\mathbb{C}^n$ of the given polynomial system $P(x) = (p_1(x), \dots, p_n(x))$ with supports $S = (S_1, \dots, S_n)$. First, all polynomials without constant term are augmented with the constant monomial $x^0 = x_1^0 \cdots x_n^0 = 1$, resulting in the corresponding augmented supports $S_j^{'} = S_j \bigcup \{0\}$. A generic system $Q(x) = (q_1(x), \dots, q_n(x))$ with the augmented support $S^{'} = (S_1^{'}, \dots, S_n^{'})$ is constructed with randomly chosen coefficients. By Theorem 2.4, the root count of $Q(x)$ in $\mathbb{C}^n$ is the mixed volume $\mathcal{M}(S_1^{'}, \dots, S_n^{'})$ of the augmented supports.

The first step in Huber and Sturmfel's polyhedral homotopy method is to solve the system $Q(x) = 0$ in general position by means of polyhedral homotopies which will be discussed below in detail. It will be shown that the number of paths needed to be followed in this step is $\mathcal{M}(S_1^{'}, \dots, S_n^{'})$ which is the root count of $Q(x)$ in $\mathbb{C}^n$, and no paths diverge.

The second step in the polyhedral homotopy method is to solve the original system $P(x) = 0$ by means of the linear homotopy $H(x,t) = (1-t)\gamma Q(x) + t P(x)$, $t \in [0,1]$ with generically chosen complex γ, also known as the so-called cheater's homotopy [27] (or the coefficient-parameter continuation [32]). In this step, the linear homotopy paths of $H(x,t) = 0$ are followed from their starting points (which are the solutions of $Q(x) = 0$ found in the first step). It is known that all isolated solutions of $P(x) = 0$ can be attained, which was our original goal. Here, some paths may diverge. Since the number of starting points, and hence the number of paths, is the root count of $Q(x)$, the number of paths needed to be followed in this step is also $\mathcal{M}(S_1', \ldots, S_n')$. Hence the total number of paths needed to be followed in both steps is $2\mathcal{M}(S_1', \ldots, S_n')$.

We begin our discussion of how to solve the system $Q(x) = 0$ using polyhedral homotopies. Writing

$$Q(x) = \begin{cases} q_1(x) = \sum_{a \in S_1'} \bar{c}_{1,a} x^a, \\ \quad \vdots \\ q_n(x) = \sum_{a \in S_n'} \bar{c}_{n,a} x^a, \end{cases} \tag{2.4}$$

note that all coefficients $\bar{c} = (\bar{c}_{j,a})$ for $a \in S_j'$ and $j = 1, \ldots, n$ are chosen at random to ensure that $Q(x)$ is in general position.

To construct a nonlinear homotopy with $Q(x)$ as the target system, first define a *generic lifting* $\omega = (\omega_1, \ldots, \omega_n)$ on the support $S' = (S_1', \ldots, S_n')$ of $Q(x)$ where each $\omega_j : S_j^{'} \to \mathbb{R}, j = 1, \ldots, n$ is a random real-valued function on $S_j^{'}$. Each ω_j lifts $S_j^{'}$ to its graph $\hat{S}_j^{'} = \{\hat{a} = (a, \omega_j(a)) \mid a \in S_j^{'}\}$.

Define the *polyhedral homotopy* $\hat{Q} : \mathbb{C}^n \times [0,1] \to \mathbb{C}^n$ by adding a power of t given by the above random lifting to each term in $Q(x)$ to obtain $\hat{Q}(x,t) = (\hat{q}_1(x,t), \ldots, \hat{q}_n(x,t))$ where

$$\hat{Q}(x,t) = \begin{cases} \hat{q}_1(x,t) & = \sum_{a \in S_1^{'}} \bar{c}_{1,a} x^a t^{w_1(a)}, \\ \quad \vdots & \\ \hat{q}_n(x,t) & = \sum_{a \in S_n^{'}} \bar{c}_{n,a} x^a t^{w_n(a)}. \end{cases} \tag{2.5}$$

In order to apply the continuation method, we must demonstrate that $\hat{Q}(x,t)$ satisfies all three of the desired properties given in the beginning of this subsection (triviality, smoothness, and accessibility). Clearly the target system is $\hat{Q}(x,1) = Q(x) = 0$. It can be shown that for any fixed $t_0 > 0$, the system $\hat{Q}(x,t_0)$ is in general position having all nonzero constant terms present which implies that it has mixed volume $k := \mathcal{M}(S_1^{'}, \ldots, S_n^{'})$ isolated solutions in $(\mathbb{C}^*)^n$. Then the entire zero set of the homotopy $Q(x,t)$ consists of k distinct homotopy paths $x^{(1)}(t), \ldots, x^{(k)}(t)$ in $(\mathbb{C}^*)^n$ ending at the distinct solutions of $Q(x) = 0$. And again, since each $\hat{q}_j(x,t)$ has nonzero constant term for all $j = 1, \ldots, n$, by a standard application of the generalized Sard's Theorem [5], each of the k solution curves of $\hat{Q}(x,t)$ is smooth with no bifurcations. Therefore, Property 2 (smoothness) holds.

However, at $t = 0$, the start system $\hat{Q}(x,0) \equiv 0$ which means that the starting points $x^{(1)}(0), \ldots, x^{(k)}(0)$ of the corresponding homotopy curves $x^{(1)}(t), \ldots, x^{(k)}(t)$ can not be identified. So the polyhedral homotopy $\hat{Q}(x,t)$ does not satisfy Property 1 (triviality) and Property 3 (accessibility). So while every isolated solution of the target system may be reached by a homotopy curve of $\hat{Q}(x,t) = 0$, it is not clear where it originates from. This difficulty is resolved by the following technique.

We first find all vectors $\hat{\alpha} = (\alpha, 1) \in \mathbb{R}^{n+1}$ matched with a collection of pairs of points $\left(\{a_1, a_1'\}, \ldots, \{a_n, a_n'\}\right)$ from the support $S' = (S_1', \ldots, S_n')$ of $Q(x)$, which together satisfy the following conditions

- Each pair of points is chosen from each support,

  i.e. $\{a_1, a_1'\} \subset S_1', \ldots, \{a_n, a_n'\} \subset S_n'$, so that $\{a_1 - a_1', \ldots, a_n - a_n'\}$ is linearly independent in $\mathbb{R}^n$, and

- Taking $\langle \cdot, \cdot \rangle$ to be the usual inner product in Euclidean space and $\hat{a} = (a, \omega_j(a)) \in \mathbb{R}^{n+1}$ where $a \in S_j'$ and $\omega_j : S' \to \mathbb{R}$ is the corresponding lifting, for $j = 1, \ldots, n$,

$$\begin{aligned} \langle \hat{a}_j, \hat{\alpha} \rangle &= \langle \hat{a}_j', \hat{\alpha} \rangle \\ \langle \hat{a}, \hat{\alpha} \rangle &> \langle \hat{a}_j, \hat{\alpha} \rangle, \qquad \text{for all } a \in S_j' \setminus \{a_j, a_j'\}. \end{aligned} \tag{2.6}$$

Geometrically, $\hat{\alpha} = (\alpha, 1)$ represents a linear functional, or alternatively, the normal of parallel hyperplanes in $\mathbb{R}^{n+1}$, which supports the convex hull of the graph $\hat{S}_j'$ of S_j', at exactly two points $\{a_j, a_j'\} \subset S_j'$ for $j = 1, \ldots, n$. Such a collection of pairs of points $\left(\{a_1, a_1'\}, \ldots, \{a_n, a_n'\}\right)$ satisfying the above conditions is called a *mixed cell* of $S' = (S'_1, \ldots, S'_n)$, and its associated vector α is called its *inner normal.* Finding all mixed cells and their associated inner normals α is one of the key steps of the polyhedral homotopy method. We will treat the actual mixed cells computation in Chapter 3 and Chapter 4.

Having found such a mixed cell $\left(\{a_1, a'_1\}, \ldots, \{a_n, a'_n\}\right)$ together with its associated inner normal α where $\alpha = (\alpha_1, \ldots, \alpha_n)$, define a change of variables between $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ by $x = yt^{\alpha}$ or alternatively, $y = t^{-\alpha}x$ where

$$\begin{array}{rclcrcl} x_1 & = & t^{\alpha_1}y_1, & & y_1 & = & t^{-\alpha_1}x_1, \\ & \vdots & & \text{or} & & \vdots & \\ x_n & = & t^{\alpha_n}y_n, & & y_n & = & t^{-\alpha_n}x_n. \end{array} \tag{2.7}$$

With this change of variables, any monomial will be transformed as follows:

$$\begin{aligned} x^a &= x_1^{a_1} \ldots x_n^{a_n} \\ &= (y_1 t^{\alpha_1})^{a_1} \ldots (y_n t^{\alpha_n})^{a_n} \\ &= y_1^{a_1} \ldots y_n^{a_n} t^{\alpha_1 a_1 + \cdots + \alpha_n a_n} \\ &= y^a t^{\langle a, \alpha \rangle}. \end{aligned}$$

Thus each $\hat{q}_j(x,t), j = 1, \ldots, n$ of the homotopy $\hat{Q}(x,t)$ in (2.2 5) becomes

$$\begin{aligned}
\hat{q}_j(yt^{\alpha},t) &= \sum_{a\in S_j'} \bar{c}_{j,a} y^a t^{\langle a,\alpha\rangle} t^{\omega_j(a)} \\
&= \sum_{a\in S_j'} \bar{c}_{j,a} y^a t^{\langle (a,\omega_j(a)),(\alpha,1)\rangle} \\
&= \sum_{a\in S_j'} \bar{c}_{j,a} y^a t^{\langle \hat{a},\hat{\alpha}\rangle} \quad \text{with } \hat{a}=(a,\omega_j(a)) \text{ and } \hat{\alpha}=(\alpha,1).
\end{aligned}$$

For $j = 1,\ldots,n$, let

$$\beta_j = \min_{a\in S_j'} \langle \hat{a},\hat{\alpha}\rangle.$$

By the conditions defining the mixed cell $\left(\{a_1,a_1'\},\ldots,\{a_n,a_n'\}\right)$, we have $\beta_j = \langle \hat{a}_j,\hat{\alpha}\rangle = \langle \hat{a}_j',\hat{\alpha}\rangle$ for $j = 1,\ldots,n$. Now define $H^{\alpha}(y,t) = (h_1^{\alpha}(y,t),\ldots,h_n^{\alpha}(y,t))$ by " factoring " out the lowest powers of t in each of the $\hat{q}_j(yt^{\alpha},t)$.

$$\begin{aligned}
h_j^{\alpha}(y,t) &= t^{-\beta_j}\hat{q}_j(yt^{\alpha},t) = \textstyle\sum_{a\in S_j'} \bar{c}_{j,a} y^a t^{\langle \hat{a},\hat{\alpha}\rangle-\beta_j} \\
&= \sum_{\substack{a\in S_j' \\ \langle \hat{\alpha},\hat{a}\rangle=\beta_j}} \bar{c}_{j,a} y^a + \sum_{\substack{a\in S_j' \\ \langle \hat{\alpha},\hat{a}\rangle>\beta_j}} \bar{c}_{j,a} y^a t^{\langle \hat{a},\hat{\alpha}\rangle-\beta_j} \\
&= \bar{c}_{j,a_j} y^{a_j} + \bar{c}_{j,a_j'} y^{a_j'} + \sum_{\substack{a\in S_j' \\ \langle \hat{\alpha},\hat{a}\rangle>\beta_j}} \bar{c}_{j,a} y^a t^{\langle \hat{a},\hat{\alpha}\rangle-\beta_j}.
\end{aligned} \tag{2.8}$$

Since exactly two inner products equal the minimum β_j for each j, our new homotopy $H^{\alpha}(y,t)$ has exactly two terms without t.

This homotopy retains most of the properties of the previous homotopy $\hat{Q}(x,t) =$

0; in particular Property 2 (smoothness) still holds, because both $H^{\alpha}(y,t)$ and $\hat{Q}(x,t)$ have the same support and are in general position for fixed $t_0 > 0$. The entire solution set of $H^{\alpha}(y,t) = 0$ consists of k distinct smooth homotopy paths $y^{(1)}(t), \ldots, y^{(k)}(t)$ in $(\mathbb{C}^*)^n$ which are transformed versions of the solutions $x^{(1)}(t), \ldots, x^{(k)}(t)$ of $\hat{Q}(x,t) = 0$. However, by the transformation (2.7), we have $x = y$ at $t = 1$. Hence the solutions to the target system $H^{\alpha}(y,1) = \hat{Q}(y,1) = Q(y) = 0$ are exactly the same as the solutions to the target system $\hat{Q}(x,1) = Q(x) = 0$. Moreover, we have an easily solvable start system, known as the *binomial system*,

$$H^{\alpha}(y,0) = \begin{cases} h_1^{\alpha}(y,0) = \sum_{a \in S_1'} \bar{c}_{1,a} y^a = \bar{c}_{1,a_1} y^{a_1} + \bar{c}_{1,a_1'} y^{a_1'} = 0, \\ \vdots \\ h_n^{\alpha}(y,0) = \sum_{a \in S_n'} \bar{c}_{n,a} y^a = \bar{c}_{n,a_n} y^{a_n} + \bar{c}_{n,a_n'} y^{a_n'} = 0. \end{cases} \tag{2.9}$$

An algorithm to solve binomial systems is laid out in Section 4.3. Thus Property 1 (triviality) is now satisfied. Since the homotopy curves originating from the solutions of (2.9) can be traced to reach a partial set of isolated zeros of $Q(y)$ at $t = 1$, Property 3 (accessibility) partially holds.

Proposition 2.5 [22] The binomial system

$$\begin{aligned} \bar{c}_{1,a_1} y^{a_1} + \bar{c}_{1,a_1'} y^{a_1'} &= 0, \\ &\vdots \\ \bar{c}_{n,a_n} y^{a_n} + \bar{c}_{n,a_n'} y^{a_n'} &= 0. \end{aligned} \tag{2.10}$$

has

$$k_\alpha := |\det(a_1 - a_1', \ldots, a_n - a_n')|$$

nonsingular isolated solutions in $(\mathbb{C}^*)^n$.

The number k_α is called the *volume* of the mixed cell $(\{a_1, a_1'\}, \ldots, \{a_n, a_n'\})$. The existence of such mixed cell $\left(\{a_1, a_1'\}, \ldots, \{a_n, a_n'\}\right)$ along with its corresponding inner normal $\alpha \in \mathbb{R}^n$ for which condition (2.6) holds and hence for which $H^\alpha(y, 0) = 0$ is a binomial system is guaranteed by the following proposition.

Proposition 2.6 For all generically chosen real functions $\omega_j : S_j' \to \mathbb{R}, j = 1, \ldots, n$ there exists $\alpha \in \mathbb{R}^n$, for which the start system $H^\alpha(y, 0) = 0$ of the homotopy $H^\alpha(y, t) = 0$ in (2.8) is a binomial system having a nonempty set of nonsingular solutions in $(\mathbb{C}^*)^n$, i.e., $k_\alpha \neq 0$ in (2.10).

This proposition was proved implicitly in [15] and [22] in the context of combinatorial geometry. In [15], Huber and Sturmfels proved that

$$M(S_1', \ldots, S_n') = \sum_\alpha k_\alpha,$$

i.e., the mixed volume of $Q(y)$, the total number of isolated zeros of $Q(y)$, is the sum of the volumes $k_\alpha' s$ of the mixed cells corresponding to all possible $\alpha' s$ for which $H^\alpha(y, 0) = 0$ is a binomial system. This gives an alternative approach to Bernshtein's Theorem, different from the original paper [1].

Clearly, different inner normals $\alpha \in \mathbb{R}^n$ induce different homotopies $H^{\alpha}(y,t) = 0$ in (2.8). Furthermore, by examining the Puiseux series expansions of the homotopy solution curves, we conclude that any two sets of isolated zeros of $Q(y)$ are disjoint, if they are reached by corresponding sets of solution paths belonging to two distinct homotopies.

Therefore, every isolated zero of $Q(y)$ can be obtained by tracing a solution curve of a homotopy $H^{\alpha}(y,t) = 0$ induced by some $\alpha \in \mathbb{R}^n$ given by Proposition 2.6.

Chapter 3

HOM4PS-2.0

Based on the polyhedral homotopies given in Section 2.2, the software package HOM4PS, developed over the years by a group led by T. Y. Li at Michigan State University, implemented this approach for approximating all the isolated zeros of $P(x) = (p_1(x), \dots, p_n(x))$. HOM4PS-2.0 is a new version in FORTRAN 90 which updates HOM4PS with remarkable speed-ups. In this chapter, we will explain the details of the three main stages in HOM4PS-2.0 that account for the major improvement over HOM4PS: mixed cell computation, following homotopy paths, and checking for curve jumping.

3.1 Mixed cell computations

When the polyhedral homotopy is employed to find all isolated zeros of $P(x) = (p_1(x), \dots, p_n(x))$, the process of locating all the *mixed cells* during the mixed volume computation plays a crucially important role [22–24]: The mixed volume determines the number of solution paths which need to be traced and the mixed cells provide starting points of the solution paths.

For polynomial system $P(x) = (p_1(x), \ldots, p_n(x))$ with support $S = (S_1, \ldots, S_n)$, let $\omega_j : S_j \to \mathbb{R}$ be a random lifting function on S_j which lifts S_j to its graph $\hat{S}_j = \{\hat{a} = (a, \omega_j(a)) : a \in S_j\} \subset \mathbb{R}^{n+1}$ for $j = 1, \ldots, n$. Here we assume that $S_j = S_j \bigcup \{0\}$, i.e., all polynomials $p_j(x)$ in the system have constant terms. See Section 2.2.

Recall that a collection of pairs $\{a_1, a_1'\} \subset S_1, \ldots, \{a_n, a_n'\} \subset S_n$ is called a mixed cell if there exists $\hat{\alpha} = (\alpha, 1) \in \mathbb{R}^{n+1}$ with $\alpha \in \mathbb{R}^n$ such that

$$\left\langle \hat{a}_j, \hat{\alpha} \right\rangle = \langle \hat{a}_j', \hat{\alpha} \rangle < \langle \hat{a}, \hat{\alpha} \rangle \quad \text{for } a \in S_j \backslash \left\{a_j, a_j'\right\}, \; j = 1, \ldots, n,$$

and α is called the inner normal of this mixed cell. For $1 \le i \le n$, $\hat{\mathrm{e}} = \{\hat{a}, \hat{a}'\} \subset \hat{S}_i$, is called *a lower edge of* $\hat{S}_i$ if there exists $\hat{\alpha} = (\alpha, 1) \in \mathbb{R}^{n+1}$ for which the following relations hold:

$$\langle \hat{a}, \hat{\alpha} \rangle = \left\langle \hat{a}', \hat{\alpha} \right\rangle \le \langle \hat{b}, \hat{\alpha} \rangle \;\; \forall \; b \in S_i \setminus \{a, a'\}.$$

Denote the set of all lower edges of $\hat{S}_i$ by $L(\hat{S}_i)$. For k distinct integers $\{i_1, \ldots, i_k\} \subset \{1, \ldots, n\}$,

$$\hat{E}_k = (\hat{\mathrm{e}}_{i_1}, \ldots, \hat{\mathrm{e}}_{i_k}), \; 1 \le k \le n, \;\; \text{where } \hat{\mathrm{e}}_{i_j} = \{\hat{a}_{i_j}, \hat{a}_{i_j}'\} \subset \hat{S}_{i_j} \;\text{ for } j = 1, \ldots, k,$$

is called a *level-k subface* of $\hat{S} = \left(\hat{S}_1, \ldots, \hat{S}_n\right)$ (or simply "level-k subface") if there exists $\hat{\alpha} = (\alpha, 1) \in \mathbb{R}^{n+1}$ such that for each $j = 1, \ldots, k$

$$\langle \hat{a}_{i_j}, \hat{\alpha} \rangle = \langle \hat{a}'_{i_j}, \hat{\alpha} \rangle \leq \langle \hat{a}, \hat{\alpha} \rangle \quad \forall \ a \in S_{i_j} \backslash \left\{ a_{i_j}, a'_{i_j} \right\}.$$

For a level-k subface $\hat{E}_k = (\hat{e}_{i_1}, \ldots, \hat{e}_{i_k})$ of $\hat{S} = (\hat{S}_1, \ldots, \hat{S}_n)$ with $1 \leq k < n$ where $\hat{e}_{i_j} = \{\hat{a}_{i_j}, \hat{a}'_{i_j}\} \in L(\hat{S}_{i_j})$ for $j = 1, \ldots, k$, we say that the lower edge $\hat{e}_{i_{k+1}} = \{\hat{a}_{i_{k+1}}, \hat{a}'_{i_{k+1}}\} \in L(\hat{S}_{i_{k+1}})$ for certain $i_{k+1} \in \{1, 2, \ldots, n\} \backslash \{i_1, \cdots, i_k\}$ *extends* the level-k subface $\hat{E}_k$ if $\hat{E}_{k+1} = (\hat{e}_{i_1}, \ldots, \hat{e}_{i_{k+1}})$ is a level-$(k+1)$ subface of $\hat{S} = (\hat{S}_1, \ldots, \hat{S}_n)$. We say $\hat{E}_k$ is *extendible* in such situations.

A main strategy for finding mixed cells is the extension of level-k subfaces $\hat{E}_k$ of $\hat{S} = \left(\hat{S}_1, \ldots, \hat{S}_n\right)$ starting from $k = 1$ and an extendible $\hat{E}_k$ when $k = n - 1$ yields mixed cells of $S = (S_1, \ldots, S_n)$ induced by elements in $\hat{S}_{i_n}$. In [8–10, 25], the *order* of this extension i_1, $i_2, \ldots$ is predetermined and fixed, that is, one always starts from extending lower edge $\hat{e}_1$ of $\hat{S}_1$ to a level-2 subface $(\hat{e}_1, \hat{e}_2)$ with $\hat{e}_2 \in L(\hat{S}_2)$, then extend $(\hat{e}_1, \hat{e}_2)$ to a level-3 subface $(\hat{e}_1, \hat{e}_2, \hat{e}_3)$ with $\hat{e}_3 \in L(\hat{S}_3)$ ... etc. Software package **MixedVol** [10] for the mixed cells computation that was adopted in the original HOM4PS was developed along this line of approach.

In [29], the novel idea of *dynamic enumeration* of all mixed cells emerged where *dynamic* means that the order of the subface extension will not stay fixed. To extend a particular level-k subface

$$\hat{E}_k = (\hat{\mathrm{e}}_{i_1}, \ldots, \hat{\mathrm{e}}_{i_k}),\ 1 \le k < n, \quad \text{where } \hat{\mathrm{e}}_{i_j} = \{\hat{a}_{i_j}, \hat{a}'_{i_j}\} \in L(\hat{S}_{i_j}) \quad \text{for } j = 1, \ldots, k,$$

one searches among $M := \{\hat{S}_l : l \in \{1, \ldots, n\} \backslash \{i_1, \ldots, i_k\}\}$ for $\hat{S}_{i_{k+1}}$ that has minimal number of suitable points where only lower edges consisting of points among them can possibly extend $\hat{E}_k$ to a level-$(k+1)$ subface. The main strategy suggested in [29] for finding such $\hat{S}_{i_{k+1}}$ is the removal of those points in each $\hat{S}_l \in M$ which have no chances to be part of a lower edge in $L(\hat{S}_l)$ that can extend $\hat{E}_k$, and select the one with minimal remaining points as $\hat{S}_{i_{k+1}}$.

For level-k subface $\hat{E}_k = (\hat{\mathrm{e}}_{i_1}, \ldots, \hat{\mathrm{e}}_{i_k})$ where $\hat{\mathrm{e}}_{i_j} = \{\hat{a}_{i_j}, \hat{a}'_{i_j}\} \in L(\hat{S}_{i_j})$ for $j = 1, \ldots, k$, let $Q := \{i_1, \ldots, i_k\} \subset \{1, \ldots, n\}$. For a fixed $l \in \{1, \ldots, n\} \backslash Q$, let $\hat{b}_v$ be a particular point in $\hat{S}_l$, and consider the set of constraints:

$$\begin{aligned} \langle \hat{a}_\mu, \hat{\alpha} \rangle &= \left\langle \hat{a}'_\mu, \hat{\alpha} \right\rangle \quad \forall\ \mu \in Q \\ &\le \langle \hat{a},\ \hat{\alpha} \rangle \quad \forall\, \hat{a} \in \hat{S}_\mu \backslash \left\{ \hat{a}_\mu, \hat{a}'_\mu \right\} \\ \langle \hat{b}_v, \hat{\alpha} \rangle &\le \langle \hat{b}, \hat{\alpha} \rangle \quad \forall\ \hat{b} \in \hat{S}_l \backslash \left\{ \hat{b}_v \right\} \end{aligned} \tag{3.1}$$

in terms of unknowns $\hat{\alpha} = (\alpha, 1) \in \mathbb{R}^{n+1}$. Apparently, when the set of constraints in (3.1) is infeasible, then there is no $\hat{b}_\mu$ in $\hat{S}_l$ such that $\{\hat{b}_v, \hat{b}_\mu\}$ can extend $\hat{E}_k$ to become a level-$(k+1)$ subface, and therefore $\hat{b}_v$ can be safely removed from $\hat{S}_l$. For the feasibility of the set of constraints in (3 1), consider the linear programming

(LP) problem

$$\text{(P)} \qquad\qquad \text{Max} \quad \langle \mathbf{r}, \alpha \rangle$$
$$\text{Subject to} \quad (3.1)$$

where $\mathbf{r} \in \mathbb{R}^n$ is any fixed vector. By the duality theorem, the feasibility of the inequalities in (3.1) can be determined by the boundedness of the duality of the LP problem in (P), which can be checked by standard techniques in the textbooks.

The superiority of the resulting software **DEMiCs-0.95** in computing mixed cells with the dynamic enumeration was reported in [29]. Soon after, this idea was embedded in the original **MixedVol** and a new code **MixedVol-2.0** was developed which improves the speed of **DEMiCs-0.95** by a substantial margin as shown in [20]. As mentioned before, employing the polyhedral homotopy method for solving polynomial systems, locating all mixed cells always plays a critically important role [22–24]. The new adoption of **MixedVol-2.0** for the mixed cell computation in HOM4PS-2.0 is certainly one of the main factors accountable for its considerable speed-up.

3.2 Following homotopy paths

3.2.1 Constructing the polyhedral-linear homotopy

For polynomial system $P(x) = (p_1(x), \dots, p_n(x))$ with

$$p_j(x) = \sum_{a \in S_j} c_{j,a} x^a, \qquad j = 1, \dots, n,$$

let $Q(x) = (q_1(x), \dots, q_n(x)) = 0$ be a polynomial system having the same monomials as $P(x)$ but with randomly chosen coefficients, i.e., $q_j(x) = \sum_{a \in S_j} \bar{c}_{j,a} x^a$ where $\bar{c}_{ja}$ are randomly chosen complex numbers. In HOM4PS, this system, as elaborated in Chapter 2, is first solved by using a polyhedral homotopy $\hat{Q}(x,t) = (\hat{q}_1(x,t), \dots, \hat{q}_n(x,t))$, $t \in [0,1]$ with a random lifting given by $\omega = (\omega_1, \dots, \omega_n)$, $\omega_j : S_j \to \mathbb{R}$; i.e., $\hat{q}_j(x,t) = \sum_{a \in S_j} \bar{c}_{j,a} x^a t^{\omega_j(a)}$ for $j = 1, \dots, n$. Then a linear homotopy $H(x,t) = (1-t)\gamma Q(x) + t P(x)$, $t \in [0,1]$ with generically chosen complex γ, is constructed to reach all isolated solutions of $P(x) = 0$ by following the smooth solution paths of $H(x,t) = 0$ emanating from the solutions to $Q(x) = 0$ found above. The number of paths needed to be followed in each step is the mixed volume of the polynomial system.

In HOM4PS-2.0, these two steps were combined in one step by considering the polyhedral-linear homotopy

$$H(x,t) = (h_1(x,t), \dots, h_n(x,t)), \; x = (x_1, \dots, x_n), \; t \in [0,1]$$

where

$$h_j(x,t) = \sum_{a \in S_j} ((1-t)\bar{c}_{j,a} + tc_{j,a})x^a t^{w_j(a)}, \quad j = 1, \ldots, n.$$

Note that $H(x,1) = P(x)$. For a given mixed cell $C = (\{a_{11}, a_{12}\}, \ldots, \{a_{n1}, a_{n2}\})$ with inner normal $\alpha \in \mathbb{R}^n$, where $\{a_{j1}, a_{j2}\} \subset S_j$ for each $j = 1, \ldots, n$, after applying the change of variables $x = yt^\alpha$ where $y = (y_1, \ldots, y_n)$ and $x_j = y_j t^{\alpha_j}$ for $j = 1, \ldots, n$, and keeping the variable x in place of y, we reach the homotopy $\tilde{H}(x,t) = (\tilde{h}_1(x,t), \ldots, \tilde{h}_n(x,t))$, $t \in [0,1]$, where for $j = 1, \ldots, n$

$$\tilde{h}_j(x,t) = \sum_{a \in S_j} [(1-t)\bar{c}_{j,a} + tc_{j,a}]x^a t^{\langle \hat{a}, \hat{\alpha} \rangle}, \quad \text{with } \hat{a} = (a, w_j(a)) \text{ for } a \in S_j.$$

Letting

$$\beta_j = \min_{a \in S_j} \langle \hat{a}, \hat{\alpha} \rangle \quad \text{for } j = 1, \ldots, n$$

and "factoring out the lowest power of t" yields

$$\hat{H}(x,t) = (\hat{h}_1(x,t), \ldots, \hat{h}_n(x,t)), \quad \text{where} \quad t \in [0,1] \text{ and}$$

$$\hat{h}_j(x,t) = \sum_{a \in S_j} [(1-t)\bar{c}_{j,a} + tc_{j,a}]x^a t^{(\langle \hat{a}, \hat{\alpha} \rangle - \beta_j)}, \quad \text{for } j = 1, \ldots, n. \tag{3.1}$$

Note that we still have $\hat{H}(x,1) = P(x)$, because $x = y$ at $t = 1$ In following the solution paths of $\hat{H}(x,t) = 0$ by the prediction-correction method, the first step of the predictor at $t = 0$ cannot be taken if a power of t in $\hat{H}(x,t)$ is less than one, since $\hat{H}_t(x,t)$ would then be undefined at $t = 0$. If the minimum power of t in

(3.1) is, say, $t^{0.01}$, then changing variables with $T = t^{0.01}$ would solve the immediate problem. But it would reduce numerical stability and computational efficiency if large powers of t, such as $t^{1,000}$, were also contained in $\hat{H}(x,t)$. Then the tangent vector $\dot{x} = \hat{H}_x^{-1} * \hat{H}_t$ would contain the terms in the order of $100,000 * t^{99,999}$ which, if evaluated at any $t \in [0,1)$, would give 0. Close to 1, however, the tangent vector would become extremely steep, and step sizes for following the homotopy path would have to be correspondingly minuscule. Actually, while these sorts of problems already exist when "the polyhedral step" and "the linear step" are split as implemented in HOM4PS, they become multiply amplified when the combined polyhedral-linear homotopy is used. Ironically, notwithstanding the number of paths needed to be followed was cut in half by combining the polyhedral and linear steps, the difference between the computing times of the two approaches is almost negligible most of the time.

In HOM4PS-2.0, we address this problem by applying the transformation $s = \ln t$ in (3.1), resulting in the homotopy $\bar{H}(x,s) = (\bar{h}_1(x,s), \dots, \bar{h}_n(x,s))$, $s \in (-\infty, 0]$, where

$$\bar{h}_j(x,s) = \sum_{a \in S_j} [(1-e^s)\bar{c}_{j,a} + e^s c_{j,a}] x^a e^{s*(\langle \hat{a}, \hat{\alpha} \rangle - \beta_j)} \quad \text{for } j = 1, \dots, n.$$

Recall that mixed cell $C = (\{a_{11}, a_{12}\}, \{a_{21}, a_{22}\}, \dots, \{a_{n1}, a_{n2}\})$ with inner normal $\alpha \in \mathbb{R}^n$ satisfies the relations

$$\begin{aligned}\left\langle \hat{a}_{j1}, \hat{\alpha} \right\rangle &= \left\langle \hat{a}_{j2}, \hat{\alpha} \right\rangle, \\ \left\langle \hat{a}_{j1}, \hat{\alpha} \right\rangle &< \langle \hat{a}, \hat{\alpha} \rangle \quad \forall\, \hat{a} \in \hat{S}_j \setminus \{\hat{a}_{j1}, \hat{a}_{j2}\}, \quad j = 1, \dots, n.\end{aligned}$$

So, in each $\bar{h}_j(x,s)$, there are exactly two powers of e equal to 0, namely, for each $j = 1, \dots, n$,

$$\beta_j = \min_{a \in S_j} \langle \hat{a}, \hat{\alpha} \rangle = \left\langle \hat{a}_{j1}, \hat{\alpha} \right\rangle = \left\langle \hat{a}_{j2}, \hat{\alpha} \right\rangle.$$

Therefore at $s = -\infty$, $\bar{H}(x, -\infty)$ becomes a binomial system,

$$\begin{aligned}\bar{c}_{11} x^{a_{11}} + \bar{c}_{12} x^{a_{12}} &= 0, \\ &\vdots \\ \bar{c}_{n1} x^{a_{n1}} + \bar{c}_{n2} x^{a_{n2}} &= 0,\end{aligned}$$

having $|\det(a_{11} - a_{12}, \dots, a_{n1} - a_{n2})|$ number of nonsingular isolated solutions which provide the starting points for following the solution paths of $\bar{H}(x,s) = 0$ from $s = -\infty$ to 0. We will detail the standard procedure for solving binomial systems in Section 4.3. As we can see here, the number of paths associated to the cell is the volume of the cell. If we repeat the same process with all the mixed cells, then the total number of the homotopy paths needed to be followed equals

$$\sum_{C:\text{mixed cell}} Vol(C) = \text{mixed volume of the system} .$$

Since $\bar{H}(x,s) = \hat{H}(x,e^s)$, thus for $\bar{H}(x(s),s) = 0$ we have

$$\frac{d\bar{H}}{ds}(x(s),s) = \frac{d}{ds}\hat{H}(x,e^s) = \hat{H}_x\frac{dx}{ds} + \hat{H}_t e^s = 0. \tag{3.2}$$

It follows that $\frac{dx}{ds}\,|_{s=-\infty} = 0$ and the values of $x(s)$ stay close to invariant for large negative s. Thus, to keep $\bar{H}(x,s) \approx 0$, we choose s_0 so that terms $e^{s_0 * (\langle \hat{a},\hat{\alpha}\rangle - \beta_j)}$ for $a \in S_j \backslash \{a_{j1}, a_{j2}\}$ are negligible for all $j = 1,\ldots,n$, say on the order of 10^{-8}, as our starting s value for following the solution paths of $\bar{H}(x,s) = 0$. Since $s \in (-\infty, 0]$ the dominant or largest term with base e and exponent $s * (\langle \hat{a},\hat{\alpha}\rangle - \beta_j)$ in the polynomial

$$\bar{h}_j(x,s) = \sum_{a\in S_j} [(1-e^s)\bar{c}_{j,a} + e^s c_{j,a}]x^a e^{s*(\langle \hat{a},\hat{\alpha}\rangle - \beta_j)}$$

for all $j = 1,\ldots,n$ is given by

$$\mu := \exp\left(s * \left[\min_{a\in S_j\backslash\{a_{j1},a_{j2}\}} (\langle \hat{a},\hat{\alpha}\rangle - \beta_j)\right]\right), \quad \text{for } j = 1,\ldots,n.$$

Setting $\mu = 10^{-8}$ and solving for s, yields

$$s_0 = \frac{-8\ln 10}{\min_{\substack{a\in S_j\backslash\{a_{j1},a_{j2}\}\\ j\in\{1,2,\ldots,n\}}} (\langle \hat{a},\hat{\alpha}\rangle - \beta_j)}.$$

Tracing solution paths $x(s)$ of $\bar{H}(x,s) = 0$ from s_0 will reach isolated zeros of $P(x)$ when s reaches 0. As this is quite different from following paths in [0,1], one must move more aggressively, especially when the magnitude of s_0 is very large. From (3.2),

$$\frac{dx}{ds} = -\hat{H}_x^{-1}\,\hat{H}_t\,e^s, \quad \text{with } \hat{H}(x,t) \text{ as in (3.1) and } t = e^s.$$

So $\dfrac{dx}{ds}$ is small for large negative s_0, which justifies the adoption of a large step size at s_0, say $\delta_0 = \dfrac{-s_0}{3}$, making $s_1 = s_0 + \delta_0$. It is then followed by a standard prediction-correction algorithm at s_1 for tracing homotopy paths. In general, at s_k, we choose initial step size $\delta_k = \min\{\delta_{k-1}, \dfrac{-s_k}{3}\}$, namely, the step size remains the same as that of the previous stage, but not excessively large - not over a third of the remaining distance. When two consecutive points on the path are available along with their tangents to the path, we use the cubic Hermite interpolation rather than the Euler method as our predictor, followed by the Newton corrector. As usual, step sizes are adjusted by the chosen tolerance parameters. For instance, normally it takes no more than three iterations for the Newton corrector to converge within the desired accuracy. Therefore if the number of Newton's iterations for the correction is > 3, the step size will be cut in half to minimize the possibility that the beginning predictor estimate was too far away from the curve. On the other hand, if in two consecutive stages the step sizes were not cut, we assume the curve is flat at this moment and will take the initial step size to be the minimum of doubling the previous step size and a third of the remaining distance to 0.

As mentioned before, combining linear and nonlinear homotopies to reduce the number of solution paths needed to be followed in the polyhedral homotopy method by half was originally suggested back in 1995 [15]. However, this idea was not successfully implemented in HOM4PS because of the involved stability and efficiency problems. Addressing those difficulties by the transformation $s = ln\, t$ and parameterizing the solution paths by $s \in (-\infty, 0]$ in HOM4PS-2.0 as shown above, a substantial improvement in algorithmic efficiency and stability has been achieved as evidenced by the results of intensive numerical experiments. This combination strategy is particularly important when the polyhedral homotopies are used to solve large problems where mixed volumes of the systems are more than millions.

3.2.2 Classical linear homotopy

For some polynomial systems, such as katsura-n [3], noon-n [35] and reimer-n [37], the mixed volume of each system is the same or nearly the same as its total degree (or the Bézout number). In such situations, instead of employing polyhedral homotopies, they should be solved directly by following the total degree number of solution paths of the classical linear homotopy $H(x,t) = (1-t)\gamma Q(x) + t\,P(x) = 0$ with generic $\gamma \in \mathbb{C}$ where $Q(x) = (q_1(x), \ldots, q_n(x))$ with $x = (x_1, \ldots, x_n)$,

$$\begin{aligned} q_1 &= a_1 x_1^{d_1} - b_1, & d_1 &= \text{degree of } p_1(x), \\ &\vdots \\ q_n &= a_n x_n^{d_n} - b_n, & d_n &= \text{degree of } p_n(x) \end{aligned}$$

and randomly chosen $(a_1, \ldots, a_n, b_1, \ldots, b_n) \in \mathbb{C}^{2n}$. In this way, by avoiding polyhedral homotopies, very costly mixed cell computations for large systems are averted. Moreover, the $s = \ln t$ transformation for the powers of t used in the polyhedral-linear combination is unnecessary for a homotopy straightly linear in the parameter t. For large systems, this can also reduce a considerable amount of cpu time. We also implemented the algorithm of the classical linear homotopy for solving polynomial systems and made it an option in HOM4PS-2.0.

3.3 On curve jumping

After following all the homotopy paths, we need to check if there are paths accidentally jumping to other paths during the process of paths tracing. This may result in missing zeros. Following two very close homotopy paths may increase the chance of curve jumping that can occur at each stage of the prediction and correction. When Euler's method, or the cubic Hermite interpolation, is applied to predict a beginning estimate for the Newton correction of one homotopy path, the resulting point may become too close to the other homotopy path. Thus the correction sequence will converge to a point that is not on the desired path. Subsequently, continued with the prediction-correction procedure, we are in effect following the second path which will also be traversed independently beginning with its own starting point. From the numerical results, it looks as though two different curves each with different starting points both reach the same solution. On the other hand, reaching the same solution may not indicate the occurrence of curve jumping because of possible singular solutions. For example, when solving cyclic-8 [2] and cyclic-9 [2] systems by the homotopy method, both have more than one curve leading to the same singular solution that lies on a positive dimensional solution component.

Before dealing with the curve jumping, we wish, in the first place, to minimize the chance for curve jumping to happen during the curve tracing procedure. In HOM4PS-2.0, a more sophisticated selection for the tolerance parameters is designed for dynamically determining step size during the curve following. For instance, for Newton correction at $s = s_k$, we consider any two consecutive iteration points $x^{(m)}$ and $x^{(m+1)}$ too far apart from each other if the relative error $\dfrac{\| x^{(m+1)} - x^{(m)} \|}{\| x^{(m)} \|} >$ 10%. In this situation, we will repeat the prediction-correction process with the step size being cut in half. In addition, as mentioned before, if more than 3 steps of Newton's iterations were required to converge within the desired accuracy, we yet again cut the step size in half to minimize the possibility that the beginning predictor estimate was too far away from the curve. As shown in Table 3.4 in Section 3.5.1, these collective treatments greatly decrease the occurrences of curve jumping. In fact, they never appear in most of the systems we solved.

To check if curve jumping occurs, we must verify *all* the attained solutions to see if there are two solutions that are very close to each other. We may, for instance, consider two solutions to be *numerically identical* if the relative error of these two solutions is less than a chosen parameter $\epsilon_0 > 0$. In HOM4PS, this task was done in a quite straightforward manner, essentially each pair of solution points were compared. This will naturally become very costly when the number of solutions is big, say 100,000. Then there are $100,000 * 99,999/2$ solution pairs, and the relative error of each pairs must all be computed.

In HOM4PS-2.0, we divide all the solutions into different groups and only check solution pairs within the same group for closeness. For each isolated solution point $z = (z_1, \dots, z_n) \in \mathbb{C}^n$ we will focus on the imaginary part of its first component $z_1 = A_1 + B_1 i$ where A_1, $B_1 \in \mathbb{R}$. The decimal representation of B_1 always has a positive or negative sign associated with it and the solutions will be divided into groups that are characterized by this sign as well as the chosen and fixed k-th digit and $(k+1)$-th digit after the decimal point of the decimal representation of B_1. Each digit place has ten possibilities $\{0, 1, 2, \dots, 9\}$. So in total, the solution set is divided into 200 groups, and each group of solution points is characterized by having the same sign, the same k-th digit b_k and $(k+1)$-th digit b_{k+1} of the decimal representation of the imaginary part of its first component. Obviously, to locate solution pairs that are numerically identical, one only needs to compare solution pairs within the same group.

Along the same line, the number of groups can certainly be increased if one wishes to deal with even bigger solution set. Our numerical results show that this technique for the curve jumping detection chopped off a big amount of computing time of HOM4PS especially when solving big systems.

Now, after two (or more) numerically identical solutions are detected, call it $\bar{x}$, we first check the smallest singular value σ of the Jacobian matrix $P_x(x)$ evaluated at $\bar{x}$. When σ is bigger than a chosen threshold $\epsilon_1 > 0$, then the solution $\bar{x}$ will be considered isolated and nonsingular. The curve jumping clearly occurs. In such cases, we retrace the two associated curves with smaller step sizes. When $\sigma < \epsilon_1$, and the solution $\bar{x}$ is isolated, we will infer that the two curves reach the same solution $\bar{x}$ and curve jumping does not occur. However, we can not rule out the possibility of the curve jumping if the solution $\bar{x}$ is a *nonsingular* point lying on a higher dimensional solution component Z. A point $z \in Z$ is called nonsingular if

$$rank_{\mathbb{C}} \frac{\partial (p_1, \ldots, p_n)}{\partial (x_1, \ldots, x_n)} (z) = n - dim\ Z. \tag{3.1}$$

To differentiate those cases, the algorithm developed in [19] is used to determine the dimension of the solution component to which the solution $\bar{x}$ belongs first, followed by checking the rank condition of $\bar{x}$ in (3.1). For the rank revealing of the Jacobian, we use the scheme developed in [28] rather than calculating the whole SVD (Singular Value Decomposition) of the matrix. When $\bar{x}$ is singular, it commonly attracts more than one different solution curves as in solving cyclic-4 and cyclic-8 systems [19]. Curve jumping is only allowed to exist if $\bar{x}$ is nonsingular.

3.4 Miscellaneous aspects of HOM4PS-2.0

3.4.1 Evaluating polynomials and derivatives

The prediction-correction process for following the homotopy paths of $\bar{H}(x,s) = (\bar{h}_1(x,s),\ldots,\bar{h}_n(x,s)) = 0$ where for $j = 1,\ldots,n$,

$$\bar{h}_j(x,s) = \sum_{a\in S_j} [(1-e^s)\bar{c}_{j,a} + e^s c_{j,a}]x^a e^{s*(\langle\hat{a},\hat{\alpha}\rangle-\beta_j)} \qquad \text{and} \quad x^a = x_1^{a_1}\ldots\, x_n^{a_n},$$

requires the computation of $\bar{H}(x,s), \bar{H}_s(x,s)$, and the matrix $\bar{H}_x(x,s)$ for fixed s. What is essentially involved in evaluating $\bar{H}(x,s)$, $\bar{H}_s(x,s)$ and $\bar{H}_x(x,s)$ at a given point $x = (x_1,\ldots,x_n)$ are the evaluations of multivariate polynomials and their partial derivatives. In HOM4PS, a multivariate polynomial $g(x_1,\ldots,x_n)$ was evaluated via Horner's rule for univariate polynomials. By factoring out a variable, say x_1, $g(x_1,\ldots,x_n)$ becomes a polynomial in x_1 with coefficients in $\mathbb{C}[x_2,\ldots,x_n]$. By the same principle, those coefficients, as polynomials in one less variable, were evaluated by factoring out another variable. This may continue until the variables are exhausted.

If multivariate polynomials are evaluated in this manner, the repeated computation of the same powers of some variables seems inevitable. For instance, for the system $P(x_1, x_2, x_3) = (p_1(x_1, x_2, x_3), p_2(x_1, x_2, x_3), p_3(x_1, x_2, x_3))$ where

$$\begin{aligned} p_1 &= 2 * x_1^6 + 3 * x_2^4 + 5 * x_3^5 - 1 \\ p_2 &= 3 * x_1^6 * x_2^4 + 2 * x_1^6 * x_3^5 + 4 * x_2^4 * x_3^5 - 5 \\ p_3 &= 5 * x_1^6 * x_2^4 * x_3^5 - 7, \end{aligned} \tag{3.1}$$

x_1^6, x_2^4, and x_3^5 appear in all p_1, p_2, and p_3. When the above rule is applied, those quantities must repeatedly be computed.

For $j = 1, \ldots, n$, let

$$\begin{aligned} MaxDeg(j) &= \max\{a_j | (a_1, \ldots, a_n) \in S_1 \cup \cdots \cup S_n\} \\ &= \text{maximum power of the variable } x_j \text{ appearing in the entire} \\ &\quad\ \text{polynomial system} \end{aligned}$$

and

$$\begin{aligned} M &= \max_{j=1,\ldots,n} MaxDeg(j) \\ &= \text{the largest power of all the variables in the entire polynomial system.} \end{aligned}$$

In HOM4PS-2.0, a table T of size $n \times M$ is established to store all possible powers of x_j, $j = 1, \ldots, n$ which may appear in $\bar{H}(x, s)$, $\bar{H}_s(x, s)$, and $\bar{H}_x(x, s)$.

The first row of T stores the monomials $x_1, x_1^2, \ldots, x_1^{MaxDeg(1)}$, the second row stores the monomials $x_2, x_2^2, \ldots, x_2^{MaxDeg(2)}$, and similarly, the last row stores the

x_1	x_1^2	$\dots$	$x_1^{MaxDeg(1)}$
x_2	x_2^2	$\dots$	$x_2^{MaxDeg(2)}$
$\vdots$	$\vdots$	$\dots$	$\vdots$
x_n	x_n^2	$\dots$	$x_n^{Maxdeg(n)}$

Table 3.1: T

monomials $x_n, x_n^2, \dots, x_n^{MaxDeg(n)}$. Since $\bar{H}_x(x,s)$ contains the partial derivatives of $x^a = x_1^{a1} \cdots x_n^{an}$ that appear in $\bar{H}(x,s)$, and since $\bar{H}_s(x,s)$ contains the same $x^a = x_1^{a1} \cdots x_n^{an}$ as $\bar{H}(x,s)$, table T stores all possible powers of x_j appearing in all of $\bar{H}(x,s)$, $\bar{H}_s(x,s)$, and $\bar{H}_x(x,s)$. Those powers that are involved in any monomial evaluations, no matter how often they appear, will never be repeatedly computed.

For the system in (3.1), we have $MaxDeg(1) = 6$, $MaxDeg(2) = 4$, $MaxDeg(3) = 5$ and hence $M = 6$. The 3×6 table T is given as follows:

x_1	x_1^2	x_1^3	x_1^4	x_1^5	x_1^6
x_2	x_2^2	x_2^3	x_2^4		
x_3	x_3^2	x_3^3	x_3^4	x_3^5	

Table 3.2: T $(n = 3,\ M = 6)$

The value of a monomial evaluated at a point $x = (x_1, \dots, x_n)$ can easily be obtained from table T. For example, for $n = 3$, the quantity of $x_1^2 x_2^3 x_3$ is $T(1,2) * T(2,3) * T(3,1)$.

3.4.2 Scaling of the coefficients

Certain polynomial systems, such as cohn 2 and cohn 3 [6], have large coefficients. Large magnitudes in coefficients will result in large magnitudes in tangent vectors of the homotopy paths. This will affect the efficiency of the curve tracing because smaller step sizes need to be taken to follow the homotopy paths. The idea of scaling the system to reduce the magnitudes of the coefficients of the polynomials first appeared in [30]. We shall illustrate our scaling method by way of an example.

Example Consider the following system of two equations in two unknowns

$$\begin{aligned} 8000\,x_1^2x_2^2 - 2000\,x_1 + 1 &= 0 & (3.2)\\ 5000\,x_1x_2 - 30 &= 0. & (3.3)\end{aligned}$$

To scale the variables, let $x_1 = 10^{c_1}z_1$ and $x_2 = 10^{c_2}z_2$, and to scale the equations, multiply (3.2) by 10^{c_3} and multiply (3.3) by 10^{c_4}. This gives

$$\begin{aligned} 10^{c_3}(8000 * 10^{2c_1+2c_2}\,z_1^2z_2^2 - 2000 * 10^{c_1}z_1 + 1) &= 0\\ 10^{c_4}(5000 * 10^{c_1+c_2}\,z_1z_2 - 30) &= 0.\end{aligned}$$

Or,

$$\begin{aligned} 10^{E_1}z_1^2z_2^2 - 10^{E_2}z_1 + 10^{E_3} &= 0\\ 10^{E_4}z_1z_2 - 10^{E_5} &= 0\end{aligned}$$

where

$$\begin{aligned}
E_1 &= 2c_1 + 2c_2 + c_3 + \log_{10}(8000) \\
E_2 &= c_1 + c_3 + \log_{10}(2000) \\
E_3 &= c_3 \\
E_4 &= c_1 + c_2 + c_4 + \log_{10}(5000) \\
E_5 &= c_4 + \log_{10}(30).
\end{aligned}$$

To have the numerical stability afforded by coefficients centered about unity, we want each E_i to be close to 0. Furthermore, to reduce variability among the magnitude of the coefficients in each equation, we want the difference between each pair of $E_i's$ in an equation to be close to 0. Thus, setting

$$\begin{aligned}
r_1 &\equiv E_1^2 + E_2^2 + E_3^2 + E_4^2 + E_5^2 \\
r_2 &\equiv [(E_1 - E_2)^2 + (E_2 - E_3)^2 + (E_1 - E_3)^2] + [(E_4 - E_5)^2],
\end{aligned}$$

we wish to minimize $r = r_1 + r_2$. More explicitly,

$$\begin{aligned}
r = &(2c_1 + 2c_2 + c_3 + \log(8000))^2 + (c_1 + c_3 + \log(2000))^2 + c_3^2 \\
&+(c_1 + c_2 + c_4 + \log(5000))^2 + (c_4 + \log(30))^2 \\
&+(c_1 + 2c_2 + \log(8000) - \log(2000))^2 + (2c_1 + 2c_2 + \log(8000))^2 \\
&+(c_1 + \log(2000))^2 + (c_1 + c_2 + \log(5000) - \log(30))^2.
\end{aligned} \tag{3.4}$$

While in [30] r is considered as a second degree polynomial in four unknowns c_1, c_2, c_3, c_4 and is minimized by the solution of

$$\frac{\partial r}{\partial c_i} = 0 \quad \text{for } i = 1, 2, 3, 4,$$

we rewrite r in (3.4) as

$$\begin{aligned} r &= \left\| \underbrace{\begin{pmatrix} 2 & 2 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix}}_{x} - \underbrace{\begin{pmatrix} -\log(8000) \\ -\log(2000) \\ 0 \\ -\log(5000) \\ -\log(30) \\ \log(2000) - \log(8000) \\ -\log(8000) \\ -\log(2000) \\ \log(30) - \log(5000) \end{pmatrix}}_{b} \right\|_2^2 \\ &= \| Ax - b \|_2^2 , \end{aligned}$$

and consider its minimization as a linear least squares problem. With the solution of this least squares problem $c_1 = -3.3437$, $c_2 = 1.3495$, $c_3 = 0.0427$, and $c_4 =$

-1.5909, the original equations then become

$$\begin{aligned} 0.9064\, z_1^2 z_2^2 - z_1 + 1.1033 &= 0 \\ 1.2996\, z_1 z_2 - 0.7695 &= 0. \end{aligned}$$

Clearly, the new system has coefficients with magnitudes smaller than those of the original one. They are closer to unity and to each other. When solutions $z = (z_1, z_2)$ of the new system are located, the solutions $x = (x_1, x_2)$ can be attained by applying the transformation $x_1 = 10^{c_1} z_1$ and $x_2 = 10^{c_2} z_2$.

3.4.3 The end game

Some of the homotopy paths $x(s)$ of $\bar{H}(x, s) = 0$ for $s \in (-\infty, 0]$ may diverge, or go to ∞, as s approaches 0. (In fact, solving systems like reimer-n [37] by the homotopy continuation method, the majority of homotopy paths diverge.) Tracing divergent paths usually consumes a multiple computing time. Typically near the end of the solution path, very small step sizes must be taken to determine the convergence of the path. For a more efficient method, write

$$x_j(s) = a_j(1 - e^s)^{\frac{\omega_j}{m}}\left(1 + \sum_{i=1}^{\infty} a_{ij}(1 - e^s)^{\frac{i}{m}}\right), \quad j = 1, \ldots, n \tag{3.5}$$

where m, called the *cyclic number*, is a positive integer and $\omega = (\omega_1, \ldots, \omega_n) \in \mathbb{Z}^n$. It was shown in Morgan et al. [33] that such expansions exist for each path in the neighborhood of $s = 0$. Taking the logarithm of the absolute value of both sides of equation (3.5) yields

$$\log |x_j(s)| = \log |a_j| + \frac{\omega_j}{m} \log(1 - e^s) + \sum_{i=1}^{\infty} b_{ij}(1 - e^s)^j. \tag{3.6}$$

Here $\sum_{i=1}^{\infty} b_{ij}(1 - e^s)^j$ is the Taylor expansion of $\log(1 + \sum_{i=1}^{\infty} a_{ij}(1 - e^s)^{\frac{i}{m}})$. During the process of homotopy continuation, a sequence of points $x(s_k)$, $-\infty < s_0 < s_1 < \cdots \leq 0$ were generated. Choose two consecutive s_k and s_{k+1} close to 0, say $1 - e^{s_k} < 10^{-6}$. By equation (3.6),

$$\frac{\log |x_j(s_k)| - \log |x_j(s_{k+1})|}{\log |1 - e^{s_k}| - \log |1 - e^{s_{k+1}}|} = \frac{\omega_j}{m} + O(1 - e^{s_k}).$$

Therefore $\frac{\omega_j}{m}$ may be estimated by the left hand side of the above equation when s_k is close to 0.

In HOM4PS, so is in PHCpack [38], the convergence (or divergence) of a path $x(s) = (x_1(s), \ldots, x_n(s))$ when $s \to 0$ is determined by

1. If $w_j < 0$ for certain $j \in \{1, \ldots, n\}$, then $\frac{\omega_j}{m} < 0$, and the path component $x_j(s)$ diverges. Hence the path $x(s)$ diverges when $s \to 0$.

2. If $w_j \geq 0 \ \forall\, j = 1, \ldots, n$, then $\frac{\omega_j}{m} \geq 0 \ \forall\, j$ and all path components $x_j(s)$ converge. Hence the path $x(s)$ converges when $s \to 0$.

Newly inserted in our HOM4PS-2.0 is the observation that when $0 < \frac{\omega_j}{m} < 1$ for certain j then $\lim_{s\to 0}\left|\frac{dx_j(s)}{ds}\right| = \infty$ which implies the divergence of the component $x_j(s)$, hence the divergence of the path $x(s)$ when $s \to 0$. We thus split the second item above as follows:

2.1 If $w_j = 0 \ \forall\, j = 1, \ldots, n,$, then $\frac{\omega_j}{m} = 0 \ \forall\, j$ and all path components $x_j(s)$ converge. Hence the path $x(s)$ converges when $s \to 0$.

2.2 If $0 < \frac{\omega_j}{m} < 1$ for certain $j \in \{1, \ldots, n\}$, then the path component x_j diverges and hence the path $x(s)$ diverges when $s \to 0$.

2.3 If $\frac{\omega_j}{m} \geq 1 \ \forall\, j = 1, \ldots, n,$ then all path components $x_j(s)$ converge and hence the path $x(s)$ converges when $s \to 0$.

Our numerical results show that this splitting provides a much accurate judgement in many situations.

3.5 Numerical results

To demonstrate the performance of HOM4PS-2.0 and to compare it with the existing packages HOM4PS, PHCpack [38] and PHoM [12] which implemented the polyhedral homotopy method, we will focus on those size-expandable bench mark systems listed in Table 3.3. All the computations in this section were carried out on a Dell PC with a Pentium 4 CPU of 2.2GHz, 1GB of memory. Results presented are mainly restricted to those systems that can be solved within 12 hours of cpu time. The package HOM4PS-2.0 is written in FORTRAN 90. The binary codes as well as its Matlab interface are available at: **`http://www.math.msu.edu/~li/Software.htm`**

eco-n [30] Total degree $= 2 \cdot 3^{n-2}$
$(x_1 + x_1x_2 + \cdots + x_{n-2}x_{n-1})x_n - 1 = 0$
$(x_2 + x_1x_3 + \cdots + x_{n-3}x_{n-1})x_n - 2 = 0$
$\vdots$
$x_{n-1}x_n - (n-1) = 0$
$x_1 + x_2 + \cdots + x_{n-1} + 1 = 0$
noon-n [35] Total degree $= 3^n$
$x_1(x_2^2 + x_3^2 + \cdots + x_n^2 - 1.1) + 1 = 0$
$x_2(x_1^2 + x_3^2 + \cdots + x_n^2 - 1.1) + 1 = 0$
$\vdots$
$x_n(x_1^2 + x_2^2 + \cdots + x_{n-1}^2 - 1.1) + 1 = 0$
cyclic-n [2] Total degree $= n!$
$x_1 + x_2 + \cdots + x_n = 0$
$x_1x_2 + x_2x_3 + \cdots + x_{n-1}x_n + x_nx_1 = 0$
$x_1x_2x_3 + x_2x_3x_4 + \cdots + x_{n-1}x_nx_1 + x_nx_1x_2 = 0$
$\vdots$
$x_1x_2\cdots x_n - 1 = 0$
katsura-n [3] Total degree $= 2^n$
$2x_{n+1} + 2x_n + \cdots + 2x_2 + x_1 - 1 = 0$
$2x_{n+1}^2 + 2x_n^2 + \cdots + 2x_2^2 + x_1^2 - x_1 = 0$
$2x_nx_{n+1} + 2x_{n-1}x_n + \cdots + 2x_2x_3 + 2x_1x_2 - x_2 = 0$
$2x_{n-1}x_{n+1} + 2x_{n-2}x_n + \cdots + 2x_1x_3 + x_2^2 - x_3 = 0$
$\vdots$
$2x_2x_{n+1} + 2x_1x_n + 2x_2x_{n-1} + \cdots + 2x_{n/2}x_{(n+2)/2} - x_n = 0$ (if n is even)
$2x_2x_{n+1} + 2x_1x_n + 2x_2x_{n-1} + \cdots + x_{(n+1)/2}^2 - x_n = 0$ (if n is odd)
reimer-n [37] Total degree $= (n+1)!$
$2x_1^2 - 2x_2^2 + \cdots + (-1)^{n+1}2x_n^2 - 1 = 0$
$2x_1^3 - 2x_2^3 + \cdots + (-1)^{n+1}2x_n^3 - 1 = 0$
$\vdots$
$2x_1^{n+1} - 2x_2^{n+1} + \cdots + (-1)^{n+1}2x_n^{n+1} - 1 = 0$

Table 3.3: The polynomial systems

System	CPU time	Mixed volume	# of curve jumpings	# of isolated solutions
eco-16	6m35s	16,384	-	16,384
eco-17	22m23s	32,768	-	32,768
eco-18	1h51m30s	65,536	-	65,536
noon-10	1m27s	59,029	-	59,029
noon-11	5m32s	177,125	2	177,125
noon-12	27m29s	531,417	2	531,417
noon-13	3h7m10s	1,594,297	10	1,594,297
katsura-17	40m48s	131,072	-	131,072
katsura-18	1h35m47s	262,144	-	262,144
katsura-19	3h50m48s	524,288	4	524,288
katsura-20	8h58m00s	1,048,576	4	1,048,576
cyclic-9	44s	11,016	-	6,642
cyclic-10	2m47s	35,940	-	34,940
cyclic-11	19m40s	184,756	-	184,756
cyclic-12	1h36m40s	500,352	-	367,488
reimer-7	1m58s	40,320	-	2,880
reimer-8	30m43s	362,880	-	14,400
reimer-9	7h52m40s	3,628,800	14	86,400

Table 3.4: The performance of HOM4PS-2.0

3.5.1 The performance of HOM4PS-2.0 in dealing with curve jumping and diverging paths

Listed in Table 3.4 is the performance of HOM4PS-2.0 on solving all those bench mark systems in Table 3.3. From the 4th column in the table, one can see that the occurrences of curve jumping have been mostly diminished for HOM4PS-2.0. We must note here that curve jumping never appears for systems in each system category with sizes smaller than those listed in the table. On the other hand, we only need to retrace 10 paths (among 1,594,297 paths) for noon-13, 4 paths (among 1,048,576 paths) for katsura-20 and 14 paths (among 3,628,800) for reimer-9 due to curve jumping. Moreover, when retracing was necessary, no homotopy paths need to be retraced more than once, while, as reported in [12], multiple retracings were required very often for the software package PHoM [12] to deal with curve jumping.

As shown in the table, when solving reimer-n systems, most homotopy paths diverged. For example, the mixed volume, or the total number of paths we must follow, of the reimer-8 system is 362,880, but the number of its isolated solutions is just 14,400. While it's normally costly in tracing diverging paths, HOM4PS-2.0 is capable of determining those divergent homotopy paths very efficiently with the end game criteria discussed in Section 3.4.3 as the cpu times in the table show.

As mentioned before, we may solve katsura-n and reimer-n systems by the classical linear homotopies because the mixed volume of each system agrees with its total degree. As a comparison, we list in Table 3.5 the results of solving those systems by both the classical linear homotopy option and the polyhedral-linear homotopy option in HOM4PS-2.0. While the proof is not available at this moment, by the observation on a collective numerical data from intensive experiments on noon-n systems, the total degree of each noon-n system and its mixed volume satisfy:

$$\text{total degree} = 3^n = \text{mixed volume} + 2\,n.$$

So, when n becomes large, the difference between them becomes very slim relatively.

System	Total Degree	CPU time		# of isolated solutions
		Linear	**Poly-Linear**	
noon-10	59,029 + 20	1m27s	5m12s	59,029
noon-11	177,125 + 22	5m32s	23m27s	177,125
noon-12	531,417 + 24	27m29s	1h28m00s	531,417
noon-13	1,594,297 + 26	3h7m10s	7h02m10s	1,594,297
katsura-15	32,768	7m03s	1h50m26s	32,768
katsura-16	65,536	16m25s	-	65,536
katsura-17	131,072	40m48s	-	131,072
katsura-18	262,144	1h35m47s	-	262,144
katsura-19	524,288	3h50m48s	-	524,288
katsura-20	1,048,576	8h58m00s	-	1,048,576
reimer-7	40,320	1m58s	2m49s	2,880
reimer-8	362,880	30m43s	36m43s	14,400
reimer-9	3,628,800	7h52m40s	8h47m42s	86,400

Table 3.5: Comparison of the classical linear homotopy and the polyhedral-linear homotopy in HOM4PS-2.0

Therefore we also include them in the table. Apparently, as it shows, if the closeness of the mixed volume and the total degree of the system can be revealed beforehand, sometimes HOM4PS-2.0 can handle much bigger systems within tolerable time.

For reimer-n systems, the cpu time for finding mixed cells is very minimal (less than 1 second most of the times). While tracing the same number of curves, the differences in the cpu times of the classical linear homotopy and the polyhedral-linear homotopy in the table indicate that nonlinear homotopies can be costly for large systems.

3.5.2 HOM4PS-2.0 vs. HOM4PS

Table 3.6 lists the numerical results that compare HOM4PS-2.0 with HOM4PS. Since the classical linear homotopy method was not implemented in HOM4PS, the table only displays the results that used the polyhedral homotopy method uniformly on all the systems.

As it shows, HOM4PS-2.0 is considerably faster than HOM4PS, and the speed-up ratio increases by quite a bit as the mixed volume of the polynomial system increases. Recall that for a specific system, HOM4PS-2.0 only needs to trace the mixed volume number of homotopy paths, while twice this number of paths needs to be traced in HOM4PS. Moreover, HOM4PS-2.0 is much more powerful in dealing with larger systems. For instance, originally HOM4PS can not solve noon-13 system within tolerable time, whereas HOM4PS-2.0 followed over 1.5 million curves in 7 hours.

System	Mixed Volume (# of paths)	CPU time		Speed-up ratio
		HOM4PS	**HOM4PS-2.0**	
eco-15	8,192	33m25s	2m25s	13.8
eco-16	16,384	2h55m12s	6m35s	26.6
eco-17	32,768	-	22m23s	-
eco-18	65,536	-	1h51m30s	-
noon-9	19,665	21m41s	1m15s	17.3
noon-10	59,029	3h20m45s	5m12s	38.6
noon-11	177,125	-	23m27s	-
noon-12	531,417	-	1h28m00s	-
noon-13	1,594,297	-	7h02m10s	-
katsura-12	4,096	43m54s	1m42s	25.8
katsura-13	8,192	3h40m54s	4m56s	44.8
katsura-14	16,384	-	25m15s	-
katsura-15	32,768	-	1h50m26s	-
cyclic-9	11,016	8m37s	44s	11.8
cyclic-10	35,940	58m02s	2m47s	20.9
cyclic-11	184,756	-	19m40s	-
cyclic-12	500,352	-	1h36m40s	-
reimer-7	40,320	7m47s	2m49s	2.8
reimer-8	362,880	1h44m18s	36m43s	2.8
reimer-9	3,628,800	-	8h47m42s	-

Table 3.6: Comparison of HOM4PS and HOM4PS-2.0

3.5.3 HOM4PS-2.0 vs. PHCpack and PHoM

Listed in Table 3.7 is the comparison of the performance of HOM4PS-2.0 and PHCpack [38]. The implementation of solving polynomial systems by the classical linear homotopies is also available in PHCpack. Therefore the comparisons listed in Table 3.7 on noon-n, katsura-n and reimer-n systems whose mixed volume and total degree of each system are the same (or almost the same) are the results by using the classical linear homotopy option in each package. Our powerful code MixedVol-2.0 [20] for computing mixed cells has no place in this computation because no mixed cell computations are required. Nonetheless, as it stands, HOM4PS-2.0 still leads PHCpack in speed by a big margin in those situations.

For eco-n and cyclic-n systems, there is a considerable difference, sometimes huge, between the mixed volume and the total degree of the system. So, we must employ the polyhedral homotopy here. When the PHCpack was tested, we used its fastest option, as indicated in the package, which utilizes MixedVol [10] for mixed cell computations.

System	Total degree	CPU time		Speed-up	#isolated
		PHCpack	**HOM4PS-2.0**	ratio	solutions
eco-14	1,062,882	1h26m04s	52.9s	97.6	4,096
eco-15	3,188,646	3h55m23s	2m25s	97.4	8,192
eco-17	28,697,814	-	22m23s	-	32,768
eco-18	86,093,442	-	1h51m30s	-	65,536
noon-9	19,683	33m28s	22.2s	90.5	19,665
noon-10	59,049	2h33m27s	1m27s	105.8	59,029
noon-11	177,147	-	5m32s	-	177,125
noon-13	1,594,323	-	3h7m10s	-	1,594,297
katsura-14	16,384	2h49m00s	2m52s	59.0	16,384
katsura-15	32,768	8h22m45s	7m03s	71.3	32,768
katsura-16	65,536	-	16m25s	-	65,536
katsura-20	1,048,576	-	8h58m00s	-	1,048,576
cyclic-9	362,880	3h50m48s	44s	314.7	6,642
cyclic-10	3,628,800	11h00m23s	2m47s	237.2	34,940
cyclic-11	39,916,800	-	19m40s	-	184,756
cyclic-12	479,001,600	-	1h36m40s	-	367,488
reimer-6	5,040	15m08s	9.6s	94.5	576
reimer-7	40,320	3h45m43s	1m58s	114.7	2,880
reimer-8	362,880	-	30m43s	-	14,400
reimer-9	3,628,800	-	7h52m40s	-	86,400

Table 3.7: Comparison of HOM4PS-2.0 and PHCpack

System	Total degree	CPU time		Speed-up	#isolated
		PHoM	**HOM4PS-2.0**	ratio	solutions
eco-13	354,294	2h39m31s	19s	503.7	2,048
eco-14	1,062,882	9h57m15s	52.9s	677.4	4,096
eco-15	3,188,646	-	2m25s	-	8,192
eco-18	86,093,442	-	1h51m30s	-	65,536
noon-8	6,661	54m18s	19s	171.5	6,645
noon-9	19,683	5h01m06s	1m15s	240.9	19,665
noon-10	59,049	-	5m12s	-	59,029
noon-13	1,594,323	-	7h02m10s	-	1,594,297
katsura-11	2,048	1h21m13s	28s	174.0	2,048
katsura-12	4,096	4h00m09s	1m42s	141.3	4,096
katsura-13	8,192	-	4m56s	-	8,192
katsura-15	32,768	-	1h50m26s	-	32,768
cyclic-8	40,320	32m32s	6.8s	287.0	1,152
cyclic-9	362,880	-	44s	-	6,642
cyclic-12	479,001,600	-	1h36m40s	-	367,488
reimer-6	5,040	1h14m50s	12.1s	371.0	576
reimer-7	40,320	-	2m49s	-	2,880
reimer-9	3,628,800	-	8h47m42s	-	86,400

Table 3.8: Comparison of HOM4PS-2.0 and PHoM

Table 3.8 compares the performance of HOM4PS-2.0 and PHoM [12]. The implementation of the classical linear homotopy for solving polynomial systems is not available in PHoM. Therefore all the results listed in Table 3.8 used the polyhedral homotopy on all the systems.

System	Maximum size					
	PHoM		**PHCpack**		**HOM4PS-2.0**	
eco -	14	(1,062,882)	15	(3,188,646)	18	(86,093,442)
noon -	9	(19,683)	10	(59,049)	13	(1,594,323)
katsura -	12	(2,048)	15	(32,768)	20	(1,048,576)
cyclic -	8	(40,320)	10	(3,628,800)	12	(479,001,600)
reimer -	6	(5,040)	7	(40,320)	9	(3,628,800)

Table 3.9: Maximum sizes of polynomial systems that can be solved by PHCpack, PHoM, and HOM4PS-2.0 within 12 hours of cpu time

Table 3.9 provides the maximum sizes of the systems that can be solved by PHCpack, PHoM, and HOM4PS-2.0 within 12 hours of cpu time. The total degree of the system is given in the parenthesis. As it shows, HOM4PS-2.0 can solve systems of much larger size than the other two packages.

Chapter 4

HOM4PS-2.0para-Parallelizing HOM4PS-2.0

Each of the three main steps in HOM4PS-2.0, finding mixed cells, following homotopy paths and checking for possible curve jumping, may all be parallelized. HOM4PS-2.0para is the parallel version of HOM4PS-2.0 which parallelizes the three main steps in HOM4PS-2.0. A common feature of those three stages, as mentioned before, is that each important task on a single CPU can be successively subdivided into multiple subtasks, which can be processed independently from the others without communicating with them. We therefore adopt a simple master-workers parallel computation environment where one master CPU communicates with all worker CPUs, and workers do not communicate among themselves. We use MPI [34], which provides a message passing library for this type of communication between master and workers, in our implementation of the parallelization. In this chapter, we will address the details of parallelizing HOM4PS-2.0 in each step.

4.1 Parallelizing the computation of mixed cells

Recall that a main strategy for finding mixed cells is the extension of level-k subfaces $\hat{E}_k = (\hat{e}_{i_1}, \ldots, \hat{e}_{i_k})$ of the lifted support $\hat{S} = (\hat{S}_1, \ldots, \hat{S}_n)$ starting from $k = 1$ and when $k = n-1$ an extensible $\hat{E}_k$ yields mixed cells of $S = (S_1, \ldots, S_n)$ induced by elements in $\hat{S}_{i_n}$. In our parallelization in computing mixed cells we first choose i_1 to be the smallest integer in $\{1, \ldots, n\}$ such that $|L(\hat{S}_{i_1})| \geq |L(\hat{S}_j)|$ for all $j \in \{1, \ldots, n\} \setminus \{i_1\}$. Each lower edge in $L(\hat{S}_{i_1})$ can be extended to subfaces of the next and consecutive levels independently by employing the idea of dynamic enumeration of all mixed cells [29]. Thus each worker processor will be assigned to extend a lower edge of $L(\hat{S}_{i_1})$, and the work is dynamically distributed in the following manner.

Algorithm for the master

```
Choose $i_1$ to be the smallest integer in $\{1, \ldots, n\}$ such that $|L(\hat{S}_{i_1})| \geq |L(\hat{S}_j)|$ for all $j \in \{1, \ldots, n\} \setminus \{i_1\}$
$N$ = the # of lower edges in $L(\hat{S}_{i_1})$
$L(\hat{S}_{i_1}) = \{\hat{e}_{11}, \ldots, \hat{e}_{1N}\}$
$p$ = the # of processors (i.e., 1 master and $p-1$ workers)
do $s = 1, p-1$
```

```
    send job $s$ to worker $s$ (i.e., processor $s$ works on extending edge $\hat{\mathrm{e}}_{1s}$)
end do
```

(**Comment**: The time required for each job is different. When worker j finishes and is ready for a new job, it sends the result to the master, and receives the next open job in the queue.)

```
do $s = p$, $N + p - 1$
    receive results from worker $j$ (i.e., worker $j$ finishes its current job)
    send job $s$ to worker $j$ (i.e, worker $j$ works on extending another edge $\hat{\mathrm{e}}_{1s}$)
end do
```

(**Comment**: For $s > N$, job s is an "empty job", an ending signal to worker processors.)

Algorithm for workers

Choose i_1 to be the smallest integer in $\{1, \dots, n\}$ such that $|L(\hat{S}_{i_1})| \geq |L(\hat{S}_j)|$ for all $j \in \{1, \dots, n\} \setminus \{i_1\}$

N = the # of lower edges in $L(\hat{S}_{i_1})$

$L(\hat{S}_{i_1}) = \{\hat{\mathrm{e}}_{11}, \dots, \hat{\mathrm{e}}_{1N}\}$

```
Step 0: Receive job $s$ from the master
            If ($s > N$) then stop. (i.e., there are only $N$ edges)
            Otherwise, set $\hat{F}_1 := \{\hat{\mathrm{e}}_{1s}\}, k := 1$.
Step 1: If $k = 0$, send all mixed cells to the master. Stop.
            If $\hat{F}_k = \emptyset$, set $k := k - 1$, and go to Step 1. Otherwise go to Step 2.
```

Step 2: Select $\hat{e}_{i_k} \in \hat{F}_k$, and set $\hat{F}_k := \hat{F}_k \backslash \{\hat{e}_{i_k}\}$.

Let $\hat{E}_k = (\hat{e}_{i_1}, \ldots, \hat{e}_{i_k})$, where $\hat{e}_{i_1} = \hat{e}_{1s}$, and

$\hat{e}_{i_j} = \{\hat{a}_{i_j}, \hat{a}'_{i_j}\} \in L(\hat{S}_{i_j})$ for $j = 2, \ldots, k$, and go to Step 3.

Step 3: To extend level-k subface $\hat{E}_k$, search among

$M := \{\hat{S}_l : l \in \{1, \ldots, n\} \backslash \{i_1, \ldots, i_k\}\}$ for $\hat{S}_{i_{k+1}}$ that has minimal number of suitable points where only lower edges among them can possibly extend $\hat{E}_k$ to a level-$(k+1)$ subface.

Find $\varepsilon(\hat{E}_k)$ (= The collection of all lower edges in $L(\hat{S}_{i_{k+1}})$ that extends $\hat{E}_k$.)

If $\varepsilon(\hat{E}_k) = \emptyset$, go to Step 1. Otherwise set $\hat{F}_{k+1} = \varepsilon(\hat{E}_k)$, $k := k + 1$, then go to Step 4.

Step 4: If $k = n$, all $C = (\hat{e}_{i_1}, \ldots, \hat{e}_{i_{n-1}}, \hat{e}_{i_n})$ with $\hat{e}_{i_n} \in \hat{F}_n$ are mixed cells.

Collect all these mixed cells, set $k := k - 1$, and go to Step 1. Otherwise go to Step 2.

4.2 Parallelizing the curve tracing of polyhedral homotopy

The fact that each homotopy path can be followed independently seemingly provides a perfect environment for the parallelization. Nonetheless when the polyhedral homotopy method is used to solve polynomial systems, different mixed cells induce different polyhedral-linear homotopies with different binomial systems to solve. Initially, it would seem natural to distribute the workload by sending one mixed cell to each worker, and when each worker receives a mixed cell, it induces the corresponding polyhedral-linear homotopy, solves the associate binomial system for starting points, and follows the resulting homotopy paths. However, the volume of each mixed cell may differ, hence the number of paths followed by each worker would differ. Moreover, there may be more workers than cells. For an extreme case: one can show that reimer-n systems [37] permit only one mixed cell regardless of what sort of liftings are used. In this situation, only one worker traces homotopy paths while the rest of the workers all remain idle. Therefore, we propose to distribute the workload consisting of a suitable amount of homotopy paths rather than sending one mixed cell to each worker.

To decide how many paths should be assigned to each worker each time, let the chunk size, denoted CS, be the number of paths given to each worker to follow in each job. If CS is too small compared to the mixed volume, then too much time would be spent on sending and receiving messages between workers and the master. Since all workers finish their tasks quickly and report to the master at about the same time, traffic jams will result and furthermore it will increase the communication time between the master and workers. On the other hand, choosing relatively large CS may reduce the communication time, but it may also result in imbalance of the workload in certain situations. Tracing different paths may consume different amount of time, for instance, following divergent paths is about 3 to 5 times slower than following convergent ones. Therefore it may not take the same cpu time for all workers each to finish tracing the same number of homotopy paths. Those workers who finish all their jobs must wait for those who have not done their jobs, and the amount of waiting time depends largely on the size of CS. In the following, a method on choosing an appropriate CS to make the work time more balanced among processors is proposed.

Let MV represent the mixed volume of the system, which is known after all mixed cells are found. Let p be the number of processors (1 master and $p-1$ workers), t be the average cpu time to follow one path, and l be the average number of jobs each worker should finish. Then

$$\mathrm{CS} * l * (p-1) \approx MV \Rightarrow \mathrm{CS} = \frac{MV}{l * (p-1)},$$

and once we choose l, chunk size CS is determined. Each worker will then follow $\mathrm{CS} = \dfrac{MV}{l * (p-1)}$ homotopy paths per job. The average computation time for each worker to follow CS paths is about $\mathrm{CS} * t$. The computation time for each worker to finish its l jobs is about $t_2 = \mathrm{CS} * t * l$. The workers that finish all their jobs first must wait for those that have not done their jobs, and the waiting time is at most $t_1 = \mathrm{CS} * t$. To balance the workload distribution, the ratio between the worst case waiting time and total time required for each worker to finish all its l jobs needs to be small, say 0.01. In other words, we want $\dfrac{t_1}{t_2} = \dfrac{1}{l} \approx 0.01$. So l is about 100, yielding $\mathrm{CS} = \dfrac{MV}{100(p-1)}$. This means each worker will be assigned about 100 jobs, and each job consists of following $\frac{MV}{100(p-1)}$ homotopy paths. The aim of parallelizing our software package HOM4PS-2.0 is to increase the computing resources for solving larger systems, which inevitably requires tracing a big amount of homotopy paths. Therefore, in the following, we always take $l = 100$ for simplicity.

To distribute the jobs dynamically, we first number all mixed cells, and for each mixed cell $(\{a_{11}, a_{12}\}, \ldots, \{a_{n1}, a_{n2}\})$, we number all the isolated solutions of its associated binomial system

$$\begin{aligned} c_{11}x^{a_{11}} + c_{12}x^{a_{12}} &= 0, \\ &\vdots \\ c_{n1}x^{a_{n1}} + c_{n2}x^{a_{n2}} &= 0, \end{aligned} \tag{4.1}$$

in the following manner. Equivalent to (4.1), write

$$\begin{aligned} x^{a_{11}-a_{12}} &= b_1, \\ &\vdots \\ x^{a_{n1}-a_{n2}} &= b_n, \end{aligned} \tag{4.2}$$

with $b_j = -\frac{c_{j2}}{c_{j1}}$ for $j = 1, 2, \ldots, n$. When solving (4.2) [22, 24], the integer matrix

$$A = [a_{11} - a_{12}, \ldots, a_{n1} - a_{n2}]$$

is upper triangularized first. That is, finding an integer matrix B with $|det\, B| = 1$ such that

$$BA = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ 0 & v_{22} & \cdots & v_{2n} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & v_{nn} \end{bmatrix}.$$

Recall that if $y = (y_1, \ldots, y_n) \in \mathbb{C}^n$ and $m_1, \ldots, m_n$ are columns of an $n \times n$ integer matrix M, then $y^M := (y^{m_1}, \ldots, y^{m_n})$. With these notations, if we find $z = (z_1, \ldots, z_n) \in \mathbb{C}^n$ satisfying

$$z^{BA} = (b_1, \ldots, b_n), \tag{4.3}$$

then $x = z^B$ is a solution to (4.2). Written out explicitly, (4.3) becomes

$$\begin{aligned} z_1^{v_{11}} &= b_1, \\ z_1^{v_{12}} z_2^{v_{22}} &= b_2, \\ &\vdots \\ z_1^{v_{1n}} \cdots z_n^{v_{nn}} &= b_n. \end{aligned} \tag{4.4}$$

Without loss of generality, we may assume $v_{jj} > 0$ for all $j = 1, \ldots, n$. Now, v_{11}

solutions of z_1 in (4.4) can be expressed as

$$z_1^{(k_1)} = |b_1|^{\frac{1}{v_{11}}} Exp\left(\frac{arg\,(b_1) + 2\pi k_1 i}{v_{11}}\right) \qquad \text{for} \quad k_1 = 0, \ldots, v_{11} - 1$$

where $arg\,(b_1)$ is the principal argument of b_1. And, by forward substitutions in (4.4), for $j = 2, \ldots, n$

$$z_j^{(k_j)} = |\bar{b}_j|^{\frac{1}{v_{jj}}} Exp\left(\frac{arg\,(\bar{b}_j) + 2\pi k_j i}{v_{jj}}\right) \qquad \text{for} \quad k_j = 0, \ldots, v_{jj} - 1$$

where $\bar{b}_j = \dfrac{b_j}{z_1^{v_{1j}} \cdots z_{j-1}^{v_{j-1j}}}$ and $arg\,(\bar{b}_j)$ is the principal argument of $\bar{b}_j$. So the system in (4.4) has $v_{11} \times \cdots \times v_{nn} (= |\det A| = volume$ of the mixed cell $(\{a_{11}, a_{12}\}, \ldots, \{a_{n1}, a_{n2}\}))$ isolated solutions in total, and each solution can be written in the form

$$(z_1^{(k_1)}, \ldots, z_n^{(k_n)}) \quad \text{where} \quad 0 \le k_j < v_{jj} \quad \text{for} \quad j = 1, \ldots, n.$$

We order those isolated solutions as follows: For $1 \le l \le v_{11} \times \cdots \times v_{nn}$, consecutively applying "division algorithm" on $l - 1$ yields a unique decomposition of $l - 1$ in the form

$$l - 1 = u_1\,(v_{22} \times \cdots \times v_{nn}) + u_2\,(v_{33} \times \cdots \times v_{nn}) + \cdots + u_{n-1}(v_{nn}) + u_n.$$

Apparently, $0 \le u_j < v_{jj}$ for $j = 1, \ldots, n$, and we thereby label $(z_1^{(u_1)}, \ldots, z_n^{(u_n)})$ as the $l^{\underline{th}}$ solution of the system in (4.4). For the corresponding solution $x = z^B$

to the system in (4.1), we label it with the same order index of the solution z to the system in (4.4).

The polyhedral-linear homotopy paths can now be ordered by taking the same indices of their starting points of the associated binomial systems together with the number indices of the corresponding mixed cells. For this purpose, a table fT is established in which each entry in the first row represents the index of a mixed cell and the entry in the row below which denotes the index of the one of the solutions of the associated binomial system of the mixed cell. For example, if there are 30 mixed cells $C_1, C_2, C_3, \ldots, C_{30}$ and $\text{Volume of } C_1 = 8, \text{Volume of } C_2 = 12, \ldots, \text{Volume of } C_{30} = 8$, then the table fT is: Suppose the mixed volume

	cell 1			cell 2			...	cell 30		
Cell #	1	...	1	2	...	2	...	30	...	30
Path #	1	...	8	1	...	12	...	1	...	8

Table 4.1: fT

MV of the system is 1500 and the number of processors $p = 4$. Then CS $= \dfrac{1500}{(4-1)*100} = 5$ and the number of total jobs $= 300$ where each job consists of following 5 paths. Based on the first row $fT(1,j)$ in table fT, we associate to each job $i = 1, \ldots, 300$ a table $fT(i)$ with 3 rows and $fT(1, i*5) - fT(1, (i-1)*5+1)+1$ columns. Entries in the 1st row are the indices of the mixed cells. And entries in the 2nd row and 3rd row list the beginning indices and the ending indices of the chunk respectively. For instance, associated to job 1 is the table $fT(1)$ of size 3×1 :

$\begin{array}{|c|}\hline 1\\ \hline 1\\ \hline 5\\ \hline\end{array}$, which means the worker that receives job 1 will follow paths emanating from solutions number 1 through number 5 of the binomial system associated with cell 1. Similarly, associated to job 2 is the table $fIT(2)$ of size 3×2 : $\begin{array}{|c|c|}\hline 1 & 2\\ \hline 6 & 1\\ \hline 8 & 2\\ \hline\end{array}$. So the worker that receives job 2 will trace paths number 6 through number 8 of the binomial system associated with cell 1 as well as paths 1 and 2 of the binomial system associated with cell 2, making it 5 in total.

We summarize the above in the next two algorithms.

Algorithm for the master

```
Establish Table $fT$ of two rows in which each entry in the first row represents the
index of a mixed cell and the entry below which in the second row denotes the index
of the solution of the associated binomial system of the mixed cell
$NumOfJobs = \frac{MV}{CS}$    ($MV$ is the mixed volume and $CS$ is the chunk size.)
$p$ = # of processors    (1 master and $p-1$ workers)
do $i = 1$ to $p-1$
    send job $i$ (along with table $fT(i)$) to worker $i$
end do
do $i = p$ to $NumOfJobs + p - 1$
```

```
    receive results from worker $j$
    send job $i$ (along with table $fT(i)$) to worker $j$
end do
```

Algorithm for worker

```
receive job $i$ from the master
if ($i > NumOfJobs$), then
    stop
else
    receive table $fT(i)$ with 3 rows and $fT(1, i * CS) - fT(1, (i-1) * CS + 1) + 1$
    columns
    entries in $1^{st}$ row are the indices of the mixed cells
    entries in $2^{nd}$ row are the beginning indices of the chunk
    entries in $3^{rd}$ row are the ending indices of the chunk
    worker solves the corresponding binomial systems and follows $CS$ paths based on
    $fT(i)$
    send results to the master
end if
```

When a worker receives job i along with table $fT(i)$ from the master, it solves the requisite binomial systems for picking out the appropriate paths to follow. However, as mentioned before, some systems only allow very few mixed cells. Solving the same binomial system repeatedly will certainly effect the scalability of the dynamic distribution. So if there are only a few mixed cells, we first solve all the binomial systems corresponding to those mixed cells respectively, and save all solutions of the binomial systems. When homotopy paths are distributed to workers, we only send CS binomial system solutions as starting points without having workers solve the same binomial systems over and over again.

4.3 Parallelizing the curve tracing of the classical linear homotopy

For systems whose mixed volumes are the same or almost the same as their total degree (or the Bézout number), the polyhedral-linear homotopy method provides no advantages when it is used for solving them. As mentioned in Section 3.2.2, those systems $P(x) = (p_1(x), \ldots, p_n(x))$ with $x = (x_1, \ldots, x_n)$ should be solved directly by following the total degree number of solution paths of the classical linear homotopy $H(x,t) = (1-t)\gamma\, Q(x) + t\, P(x) = 0$ for generic $\gamma \in \mathbb{C}$ where a typical choice of $Q(x) = (q_1(x), \ldots, q_n(x))$ is

$$\begin{aligned} q_1 &= a_1 x_1^{d_1} - b_1, &\qquad d_1 &= \text{degree of } p_1(x), \\ &\vdots & & \\ q_n &= a_n x_n^{d_n} - b_n, &\qquad d_n &= \text{degree of } p_n(x), \end{aligned} \tag{4.1}$$

with randomly chosen $(a_1, \ldots, a_n, b_1, \ldots, b_n) \in \mathbb{C}^{2n}$ [24]. In this way, since no polyhedral homotopies are involved, no costly mixed cell computations are required. We now describe the parallel algorithm in our HOM4PS-2.0para for this homotopy.

While the starting system $Q(x)$ in (4.1) is a special binomial system as in (4.1), the situation is much simpler here. Let $TotalDeg = d_1 \times d_2 \times \cdots \times d_n$ be the total degree of the system $P(x)$, the total number of paths needed to be followed for the classical linear homotopy. Let p be the number of processors, among which there are 1 master and $p-1$ workers. Let CS be the number of homotopy paths that should be followed in each job. To dynamically assign the jobs, we choose $\text{CS} = \dfrac{TotalDeg}{100(p-1)}$ as discussed in the last subsection. For workers to decide which CS paths to follow, we first number all the isolated zeros of the starting system $Q(x)$ as follows: Each isolated zero of $Q(x)$ can be written in the form

$$(x_1^{(k_1)}, \ldots, x_n^{(k_n)}) \quad \text{where} \quad 0 \le k_j < d_j \quad \text{for} \quad j = 1, \ldots, n$$

and

$$x_j^{(k_j)} = |\bar{b}_j|^{\frac{1}{d_j}} Exp\left(\frac{arg\,(\bar{b}_j) + 2\pi k_j i}{d_j}\right) \qquad \text{for} \quad k_j = 0, \ldots, d_j - 1$$

with $\bar{b}_j = \dfrac{b_j}{a_j}$ and $arg\,(\bar{b}_j)$ being the principal argument of $\bar{b}_j$. Again, for $1 \le l \le TotalDeg$, consecutively applying "division algorithm" on $l-1$ will result in a unique decomposition of $l-1$ in the form

$$l-1 = u_1\,(d_2 \times \cdots \times d_n) + u_2\,(d_3 \times \cdots \times d_n) + \cdots + u_{n-1}(d_n) + u_n \qquad (4.2)$$

with $0 \le u_j < d_j$ for $j = 1, \dots, n$. So, as before, $(x_1^{(u_1)}, \dots, x_n^{(u_n)})$ will be the $l^{\underline{th}}$ solution of the starting system $Q(x) = 0$, and homotopy paths will be ordered by taking the same indices of their starting points, respectively, as solutions of $Q(x) = 0$. With this order arrangement, the worker that receives job i for $1 \le i \le \frac{TotalDeg}{\mathrm{CS}}$ will follow homotopy paths from number $(i-1)\mathrm{CS}+1$ to number $i\,\mathrm{CS}$, which is CS of them in total.

On the other hand, this order arrangement can also help to avoid saving *all* the solutions of the system $Q(x) = 0$ in the main memory by constructing the table Bixlib of size $n \times d$ where $d = \max\{d_1, \dots, d_n\}$, in which the $j^{\underline{th}}$ row stores the d_j solutions $x_j^{(0)}, \dots, x_j^{(d_j-1)}$ of the equation $a_j x_j^{d_j} - b_j = 0$ for $j = 1, \dots, n$. As an example, for $n = 3$, $d_1 = 8$, $d_2 = 5$ and $d_3 = 10$, the 3×10 Bixlib table is:

$x_1^{(0)}$	$x_1^{(1)}$	$x_1^{(2)}$	$x_1^{(3)}$	$x_1^{(4)}$	$x_1^{(5)}$	$x_1^{(6)}$	$x_1^{(7)}$		
$x_2^{(0)}$	$x_2^{(1)}$	$x_2^{(2)}$	$x_2^{(3)}$	$x_2^{(4)}$					
$x_3^{(0)}$	$x_2^{(1)}$	$x_3^{(2)}$	$x_3^{(4)}$	$x_3^{(4)}$	$x_3^{(5)}$	$x_3^{(6)}$	$x_3^{(7)}$	$x_3^{(8)}$	$x_3^{(9)}$

Table 4.2: Bixlib

When table Bixlib is established, one can easily identify the $l^{\underline{th}}$ solution $(x_1^{(u_1)}, \dots, x_n^{(u_n)})$ of $Q(x) = 0$ as long as $u_1, \dots, u_n$ in the decomposition of $l-1$ in (4.2) are determined, and there is no need to save *TotalDeg* number of solutions of $Q(x) = 0$ in the memory.

We now list the algorithms for the master and the workers below.

Algorithm for the master

```
Create table Bixlib

$p =$ # of processors   (1 master and $p-1$ workers)

$NumOfJobs = \frac{TotalDeg}{CS}$   ($TotalDeg$ is the total degree and $CS$ is the chunk size)

do $i = 1$ to $p-1$
    send job $i$ to worker $i$
end do

do $i = p$ to $NumOfJobs + p - 1$
    receive results from some worker $j$
    send job $i$ to worker $j$
end do
```

Algorithm for worker

```
receive job $i$ from the master

if ($i > NumOfJobs$), then
   stop
else
   receive 2 integers $(i-1)\mathrm{CS}+1$ and $i\,\mathrm{CS}$, indicating the starting and ending
   indices of paths
   for each integer between the starting and ending indices, find the corresponding
```

```
    starting point from table Bixlib to follow the path
    send results to the master
end if
```

4.4 Parallelizing the checking of possible curve jumping

Once all the solutions are attained after following all homotopy paths, we proceed to check for possible curve jumping. First of all, the master will divide the solutions into 200 groups, as we explained in Section 3.3, where each group of solutions is characterized by having the same sign of the imaginary part of the solutions' first component as well as the same k-th digit b_k and $(k+1)$-th digit b_{k+1} after the decimal point of its decimal representation. Note that solutions may not be evenly distributed among those groups, therefore for systems having large number of isolated solutions, some of those 200 groups may still have a big amount of solutions. In HOM4PS-2.0para, if any group has more than say 1,000 solutions, then this specific group will be divided into 200 subgroups again based on the imaginary part of the second component of each solution in the group. A table gpT is established in which each entry in the first row represents the index of a group and the entry in the row below denotes the index of subgroup associated to the group in the first row. We use 0 in the second row to indicate that if the group in the first row does not have more than 1,000 elements, hence no further division of the group is necessary. For example, let $g_1, \ldots, g_{200}$ denote the original 200 groups. Suppose $|g_2| > 1,000, |g_3| > 1,000$, and $|g_4| > 1,000$ and the rest of the groups have no more than 1000 elements. Then g_2, g_3, and g_4 will all be divided into 200 subgroups based on the imaginary part of the second component of each solution in those groups. Let $g_{(2,1)}, \ldots, g_{(2,200)}, g_{(3,1)}, \ldots, g_{(3,200)}, g_{(4,1)} \cdots, g_{(4,200)}$ be the subgroups of g_2, g_3, and g_4 respectively. The table gpT becomes: In total, there are $(200 - 3) +$

	g_1	g_2			g_3			g_4			$\ldots$	g_{200}
Group #	1	2	...	2	3	...	3	4	...	4	...	200
Subgp #	0	1	...	200	1	...	200	1	...	200	...	0

Table 4.3: gpT

$200 * 3 = 797$ groups (or subgroups) which need to be checked. We dynamically distribute the jobs in a similar manner as what was done before as discussed in Section 4.2 and 4.3. Suppose that there are 4 processors (1 master and 3 workers). Let CS(=chunk size) be the number of groups needed to be checked in each job. Here, in our case, $CS = \dfrac{797}{(4-1)100} \approx 3$. This means each worker would have to check 3 groups (or subgroups) in each job, and the total number of jobs is $\dfrac{797}{3} = 266$. Based on the first row $gpT(1,j)$ in table gpT, to each job $i = 1, \ldots, 266$, we associate a table $gpT(i)$ with 3 rows and $gpT(1, i * 3) - gpT(1, (i-1) * 3 + 1) + 1$ columns. Entries in the 1^{st} row are the indices of the original groups. And entries in the 2^{nd} row and 3^{rd} row list the beginning indices and the ending indices of the subgroups associated to the group index in the first row respectively. For instance, associated to job 1 is a table $gpT(1)$ of size 3×2 :

1	2
0	1
0	2

, which means the worker that receives job 1 will check group g_1 and also subgroups $g_{(2,1)}$ and $g_{(2,2)}$ of the group g_2. Here 0 in the second and third row of table $gpT(1)$ means g_1 has no more than 1,000 elements. There is no need to divide g_1 into 200 subgroups. Associated to job 2 is a table $gpT(2)$ of size 3×1 :

2
3
5

. So the worker that receives job 2 will check subgroups $g_{(2,3)}, g_{(2,4)}$ and $g_{(2,5)}$ of group g_2 making it 3 in total. Similarly, associated to job 3 is a table $gpT(3)$ of size 3×1 :

2
6
8

. So the worker that receives job 3 will check subgroups $g_{(2,6)}, g_{(2,7)}$ and $g_{(2,8)}$ of the group g_2.

The dynamic distribution algorithms for the master and workers for checking the curve jumping described in Section 3.3 are listed below.

Algorithm for the master

$p =$ number of processors (1 master and $p-1$ workers)

Divide the solution set into 200 groups $g_1, \ldots, g_{200}$ based on the imaginary part of the first component of each solution.

Establish Table gpT of two rows in which each entry in the first row represents the index of a group among $g_1, \ldots, g_{200}$ and the entry below which in the second row denotes the index of subgroup associated to the group in the first row.

Note: entries 0 in the second row of *gpT* means the corresponding group in the first row has less than or equal to 1,000 elements.

Let $N = \#\{i \in \{1, 2 \ldots, 200\} \mid |g_i| > 1,000\}$

$TotalGps = 200 * N + (200 - N)$: total number of groups (or subgroups)

$CS = \dfrac{TotalGps}{(p-1)100}$: number of groups (or subgroups) in each job

$NumOfJobs = \dfrac{TotalGps}{CS}$

```
do $i = 1$ to $p - 1$
    send job $i$ (along with $gpT(i)$) to worker $i$
end do
do $i = p$ to $NumOfJobs + p - 1$
    receive computation results from some worker $j$
    send job $i$ (along with $gpT(i)$) to worker $j$
end do
```

(**Comment**: For $i = 1, \ldots, NumofJobs$, job i consists of checking solutions in groups (or subgroups) from table $gpT(i)$. Job $NumofJobs + 1$, job $NumofJobs + 2$, $\ldots$, job $NumOfJobs + p - 1$ are empty jobs.)

Algorithm for the workers

```
receive job $i$ from the master
```

```
if ( $i > NumOfJobs$ ) stop
if ( $i \leq NumOfJobs$ ) then
  receive table $gpT(i)$ with 3 rows and $gpT(1, i*CS) - gpT(1, (i-1)*CS+1)+1$
  columns
  entries in 1st row are the indices of among the groups $g_1, \dots, g_{200}$
  entries in 2nd row are the beginning indices of the subgroups in the chunk
  entries in 3rd row are the ending indices of the subgroups in the chunk
  for any group (or subgroup) $g = \{x_1, \dots, x_{|g|}\} \in gpT$
    do $l = 1$ to $|g| - 1$
      do $m = l + 1$ to $|g|$
        if $\frac{\| x_l - x_m \|}{\| x_l \|}$ < a chosen parameter $\epsilon_0$, then
          if the minimum singular value of the Jacobian matrix $P_x(x_l)$ is bigger
            than a chosen threshold $\epsilon_1$, then retrace the two paths with smaller
            step sizes
            send the updated solutions to the master
          else
            if $x_l$ is a nonsingular point on a positive dimensional component
              curve jumping occurs
              retrace the two paths with smaller step sizes
            end if
          end if
```

```
            end if
          end do
        end do
      end for
    end if
```

After two (or more) numerically identical solutions are detected and the occurrence of curve jumping is determined, we need to identify the associated paths and retrace them with smaller step sizes. For this purpose, the order index of the path must be saved when it reaches a zero of $P(x) = (p_1(x), \ldots, p_n(x))$ at the end of the path. When the polyhedral-linear homotopy is used to solve the system, each path corresponds to a unique mixed cell along with a unique path number in that cell as elaborated in Section 3.2. We save these two numbers for the path so that identification of the mixed cell provides the associated binomial system, yielding the starting point of a particularly numbered path and so that retracing of the paths may then proceed. For the classical linear homotopy, the situation is simpler. The solution paths of the homotopy are numbered according to the order indices of their starting points as solutions of the starting system. Hence storing the path number is sufficient for retracing the path.

Remark: In the very beginning, when the master receives the solutions of $P(x) = 0$ in $\mathbb{C}^n$, they are saved in a 2-dimensional array of complex numbers in HOM4PS-2.0para. The size of the 2-dimensional array is $n \times MV$ for the polyhedral homotopy and $n \times TotalDeg$ for the classical linear homotopy (n complex components of the solution and MV or $TotalDeg$ number of solutions). As before, MV is the mixed volume of $P(x)$ and $TotalDeg$ is the total degree of $P(x)$. Different from the parallel version of PHCpack [39], in which the master will write the solutions of $P(x) = 0$ to a file when the solutions are received from the workers, we save all the solutions in an array for the purpose of grouping the solutions for the detection of curve jumping. Having the solutions stored in an array of complex numbers is convenient for retrieving the necessary information to retrace the homotopy paths, and for updating the new solutions.

4.5 Numerical results

To demonstrate the performance of our parallel software package HOM4PS-2.0para we will focus on those size-expandable bench mark systems listed in Table 3.3. All the computations were carried out on a cluster 8 AMD dual 2.2 GHz cpus. Again, we only list those systems that can be solved within 12 hours cpu time.

4.5.1 The performance on large systems

Table 4.4 shows the results of solving eco-n and cyclic-n systems using 1 master and 7 workers. For eco-n and cyclic-n systems, the total degree of each individual system is much greater than its mixed volume, especially when n becomes larger. We therefore use the polyhedral-linear homotopy to solve those systems, and the total number of homotopy paths that need to be followed is the mixed volume of the system. As it shows, curve jumping rarely occurs. On the other hand, when retracing was necessary, no homotopy paths need to be retraced more than once.

System	CPU time	Total degree	Mixed volume (=# of paths)	# of curve jumping	# of isolated solutions
eco-17	2m11s	28,697,814	32,768	-	32,768
eco-18	6m30s	86,093,442	65,536	-	65,536
eco-19	26m26s	258,280,326	131,072	-	131,072
eco-20	1h29m29s	774,840,978	262,144	-	262,144
eco-21	10h08m55s	2,324,522,934	524,288	1	524,288
cyclic-11	3m34s	39,916,800	184,756	-	184,756
cyclic-12	14m07s	479,001,600	500,352	-	367,488
cyclic-13	1h39m10s	6,227,020,800	2,704,156	-	2,704,156
cyclic-14	7h32m42s	87,178,291,200	8,795,976	4	8,464,307

Table 4.4: Solving systems by the polyhedral-linear homotopy with 1 master and 7 workers

Table 4.5 lists the results of solving systems noon-n, katsura-n, and reimer-n with again 1 master and 7 workers. For katsura-n and reimer-n systems, total degree of each system is equal (or almost equal) to the mixed volume of the system. Therefore we solved these systems by using the classical linear homotopy to avoid the costly mixed cell computations. Indeed, it is very costly to find all the mixed cells of katsura-n system when $n > 15$. For noon-n systems, the difference between the total degree of each system and its mixed volume is $2n$ [21], which becomes much slimmer relatively as n becomes larger. Hence we also solved noon-n systems by the classical linear homotopy. Note that while curve jumping occurs in solving very big systems, the number of occurrences is very small relatively. Again, we only need to retrace the homotopy paths once when curve jumping was detected.

System	CPU time	Total degree (=# of paths)	# of curve jumping	# of isolated solutions
noon-12	2m23s	531,417+24	-	531,417
noon-13	7m48s	1,594,297+26	-	1,594,297
noon-14	38m12s	4,782,941+28	-	4,782,941
noon-15	4h14m33s	14,348,877+30	-	14,348,877
katsura-18	9m46s	262,144	-	262,144
katsura-19	23m36s	524,288	2	524,288
katsura-20	55m10s	1,048,576	4	1,048,576
katsura-21	2h08m42s	2,097,152	8	2,097,152
katsura-22	4h52m01s	4,194,304	20	4,194,304
katsura-23	11h17m40s	8,388,608	52	8,388,608
reimer-8	2m36s	362,880	-	14,400
reimer-9	28m04s	3,628,800	8	86,400
reimer-10	8h40m46s	39,916,800	20	518,400

Table 4.5: Solving systems by the classical linear homotopy with 1 master and 7 workers

System	# of workers	Total time to solve system		Time to find mixed cells		Time to trace curve		Time to check solutions	
	k	cpu(s)	ratio	cpu(s)	ratio	cpu(s)	ratio	cpu(s)	ratio
eco-16	1	445.32	1.00	120.02	1.00	325.00	1.00	0.30	1.00
	2	223.49	1.99	60.66	1.98	162.58	2.00	0.25	1.20
	3	150.69	2.96	40.94	2.93	109.53	2.97	0.22	1.36
	5	91.31	4.88	25.22	4.76	65.89	4.93	0.20	1.59
	7	68.70	6.48	19.99	6.00	48.58	6.69	0.13	2.31
cyclic-11	1	1475.39	1.00	38.15	1.00	1436.49	1.00	0.75	1.00
	2	734.96	2.00	19.10	2.00	715.41	2.00	0.45	1.67
	3	494.19	2.99	12.94	2.95	480.86	2.99	0.39	1.92
	5	295.90	4.99	8.05	4.74	287.47	5.00	0.38	1.97
	7	212.87	6.93	6.47	6.00	206.06	6.97	0.34	2.21

Table 4.6: The scalability of solving systems by the polyhedral-linear homotopy

4.5.2 Scalability

To test the scalability of our HOM4PS-2.0para, we solved eco-16, noon-12, cyclic-11, katsura-17 and reimer-8 systems with varying numbers, $k = 1, 2, 3, 5, 7$, of workers. Table 4.6 shows the scalability of solving eco-16 and cyclic-11 by the polyhedral-linear homotopy and Table 4.7 exhibits the scalability of solving reimer-8, noon-12 and katsura-17 systems by the classical linear homotopy. The ratios in both tables stand for the cpu time for one worker to finish the task divides by the cpu time of k workers for the same task. The ratios for curve tracing in both tables illustrate an excellent scalability (nearly perfect) for each system, showing evidently that each path can be traced independently. The cpu time to check the solutions on both tables includes the cpu times for the detection of curve jumping as well as retracing the associated paths when curve jumping occurs. The ratios in this category do not

show an excellent scalability because the cpu time for checking curve jumping is

System	# of workers	Total time to solve system		Time to trace curve		Time to check solutions	
	k	cpu(s)	ratio	cpu(s)	ratio	cpu(s)	ratio
noon-12	1	1003.32	1.00	980.72	1.00	22.50	1.00
	2	501.75	2.00	490.33	2.00	11.42	1.97
	3	335.18	2.99	326.68	3.00	8.50	2.65
	5	201.27	4.98	195.30	5.00	5.97	3.77
	7	143.22	7.00	138.88	7.00	4.34	5.18
reimer-8	1	1088.95	1.00	1087.74	1.00	1.21	1.00
	2	545.08	2.00	543.96	2.00	1.12	1.08
	3	363.89	2.99	362.81	3.00	1.08	1.12
	5	218.69	4.98	217.97	4.99	0.72	1.68
	7	156.54	6.96	155.86	6.98	0.68	1.78
katsura-17	1	1964.22	1.00	1963.12	1.00	1.10	1.00
	2	982.38	2.00	981.35	2.00	1.03	1.07
	3	654.17	3.00	653.22	3.00	0.95	1.16
	5	394.02	4.99	393.18	4.99	0.84	1.31
	7	280.56	7.00	279.85	7.00	0.71	1.55

Table 4.7: The scalability of solving systems by the classical linear homotopy

very fast, as one can see, and does not change much for various number of workers.

As expected, when the classical linear homotopy is used the scalabilities in Table 4.7 are very close to being perfect. Notably illuminating is the almost perfect scalability in solving the whole system of cyclic-11 in Table 4.6. This shows that when the polyhedral-linear homotopy is used our strategy for the dynamic workload distribution is quite balanced.

4.5.3 The multiple precision

As in general, we use Newton's method for the "corrector" in our prediction-correction scheme in following homotopy paths. Namely, when the predictor (we use the cubic Hermite interpolation as our predictor after the first step which uses the Euler predictor) provides a prediction point (x_0, t_0), Newton's iteration is applied to $H(x, t_0)$ initiated at (x_0, t_0), until the magnitude of $\|H(x, t_0)\|$ is less than a predetermined accuracy $\epsilon > 0$. Realistically, this accuracy parameter should vary according to the size of the Jacobian $\|H_x\|$ as well as the size of $\|x\|$, and in HOM4PS-2.0, we set ϵ to be the minimum of $\|H_x\| * \|x\| * 10^{-8}$ and 0.1.

Now, during the computation, if the machine precision is set to be double precision, it allows 15 digits accuracy. So when the condition number of the Jacobian matrix $H_x(x, t_0)$ is greater than 10^{15}, then Newton's method can hardly converge within the desired accuracy. The step size will then be cut in half, followed by repeating the prediction-correction scheme. If the step size is less than the minimum step size allowable, say 10^{-11}, before reaching the stage of the end game, then the procedure for tracing this path will cease to continue. We then retrace this homotopy path by increasing the precision from double precision to quad precision which is about 10 times slower in tracing paths. Fortunately, those situations seldom occur, and therefore causing only very minor effect on the scalability and efficiency. For instance, only about 7 (among $65,536$) paths for eco-18, 20 (among $131,072$) paths for eco-19, and 18 (among 262,144) paths for eco-20 need to use quad precision to retrace paths.

Chapter 5

Conclusion and future work

Polyhedral-linear and classical linear homotopies can be used to solve large polynomial systems efficiently. The sequential version HOM4PS-2.0 leads the existing software by huge margins. With multi-processors, HOM4PS-2.0para-parallel version of HOM4PS-2.0 can solve systems that have not been solved in the past within tolerable time.

As we can see from Table 4.6, the ratios of computing mixed cells do not exhibit near perfect scalabilities as the ratios of tracing curves show. At this time, while the master sends the lower edges of $L(\hat{S}_{i_1})$ to the workers dynamically, some lower edges may take much more time for the extension than the others. Therefore at the very end, workers who finish their lower edge extensions will need to wait for those workers who have not finished theirs. This will result in an imbalance of workload distribution. Apparently a more sophisticated method needs to be developed for the workload distribution for mixed cell computations.

Bibliography

[1] D. N. Bernshtein (1975), "The number of roots of a system of equations", *Functional Analysis and Appl.*, **9**(3), 183-185.

[2] G. Björk and R. Fröberg (1991), "A faster way to count the solutions of inhomogeneous systems of algebraic equations", *J. Symbolic Computaion*, **12**(3), 329-336.

[3] W. Boege, R. Gebauer, and H. Kredel (1986), "Some examples for solving systems of algebraic equations by calculating Groebner bases", *J. Symbolic Computation*, **2**, 83-98.

[4] J. Canny and J. M. Rojas (1991), "An optional condition for determining the exact number of roots of a polynomial system", *IN Proceedins of the 1991 International Symposium on Symbolic and Algebraic Computaion*,ACM,83-98.

[5] S. N. Chow, J. Mallet-Paret, and J. A. Yorke (1978), "Finding zeros of maps: homotopy methods that are constructive with probability one", *Math. Comput.*, **32**, 887-899.

[6] H. Cohn (1982) "An explicit modular equation in two variables and Hilbert's twelfth problem", *Math. of Comp.*, **38**, 227-236.

[7] F. J. Drexler (1977) "Eine Methode zur Berechnung sämtlicher Lösungen von Polynongleichungssystemen", *Numer. Math.*, **29**, 45-58.

[8] T. Gao and T. Y. Li (2000), "Mixed volume computation via linear programming", *Taiwan J. of Math.*, **4**, 599-619.

[9] T. Gao and T. Y. Li (2003), "Mixed volume computation for semi-mixed systems", *Discrete Comput. Geom.*, **29**(2), 257-277.

[10] T. Gao, T. Y. Li and M. Wu (2005), "Algorithm 846: MixedVol: A software package for mixed volume computation", *ACM Transactions on Math. Software*, **31**(4), 555-560.

[11] C. B. Garcia and W. I. Zangwill (1979), "Findind all solutions to polynomial systems and other systems of equations", *Math. Programming*, **16**, 159-176.

[12] T. Gunji, S. Kim, M. Kojima, A. Takeda, K. Fujisawa and T. Mizutani (2004), "PHoM - A polyhedral homotopy continuation method", *Computing*, **73**, 53-57.

[13] T. Gunji, S. Kim, K. Fujisawa and M. Kojima (2006), "PHoMpara - Parallel implementation of the polyhedral homotopy continuation method for polynomial systems", *Computing*, **77**, 387-411.

[14] B. Huber (1996), "Solving sparse polynomial systems", Ph.D. dissertation, Cornell University.

[15] B. Huber and B. Sturmfels (1995), "A Polyhedral method for solving sparse polynomial systems", *Math. Comp.*, **64**, 1541-1555.

[16] A. G. Khovanski (1978), "Neton polyhedral and the genus of complete intersections", *Functional Analysis and Appl.*, **12**(1), 38-46.

[17] S. Kim and M. Kojima (2004), "Numerical stability of path tracing in polyhedral homotopy continuation methods", *Computing*, **73**, 329-348.

[18] A. G. Kushnirenko (1976), "Newton Plytopes and the Bezout Theorem", *Functional Analysis and Appl.*, **10**(3), 233-235.

[19] Y. C. Kuo and T. Y. Li (2008), "Determine whether a numerical solution of a polynomial system is isolated", *J. Math. Anal. Appl.*, **338**(2), 840-851.

[20] T. Lee and T. Y. Li,"Mixed volume computation, A Revisit",(Submitted).

[21] T. Lee, T. Y. Li and C. Tsai," HOM4PS-2.0: A software package for solving polynomial systems by the polyhedral homotopy continuation method", (Submitted).

[22] T. Y. Li (1997), "Numerical solution of multivariate polynomial systems by homotopy continuation methods", *ACTA Numerica*, 399-436.

[23] T. Y. Li (1999), "Solving polynomial systems by polyhedral homotopies", *Taiwan J. of Math.*, **3**, 251-279.

[24] T. Y. Li (2003), "Solving polynomial systems by the homotopy continuation method", *Handbook of numerical analysis*, Vol. XI, Edited by P. G. Ciarlet, North-Holland, Amsterdam.

[25] T. Y. Li and X. Li (2001), "Finding mixed cells in the mixed volume computation", *Foundation of Computational Mathematics*, **1**, 161-181.

[26] T. Y. Li and X. Wang (1996), "The BKK root count in $\mathbb{C}^n$", *Math. Comp.*, **65**, no. 216, 161-181.

[27] T. Y. Li, T. Sauer and J. A. Yorke (1989), "The cheaters homotopy: an efficient procedure for solving systems of polynomial equations", *SIAM J. Numer. Anal.*, **26**, 1241-1251.

[28] T. Y. Li and Z. Zeng (2005), "A rank-revealing method with updating, downdating, and applications", *SIAM J. Matrix Anal. Appl.*, **26**, 918-946.

[29] T. Mizutani, A. Takeda and M. Kojima (2007), "Dynamic enumeration of all mixed cells", *Discrete Comput. Geom.*, **37**, 351-367.

[30] A. MORGAN, (1987), *Solving polynomial systems using continuation for engineering and scientific problems*, Prentice-Hall, New Jersey.

[31] A. P. Morgan and A. J. Sommese (1987), "A homotpy for solving general polynomial systems that respect m-homogeneous structures", *Appl. Math. Comput.*,**24**,101-113.

[32] A. P. Morgan and A. J. Sommese (1989), "Coefficient-parameter polynomial continuation", *Appl. Math. Comput.*, **29**, 123-160. Errata: *Appl. Math. Comput.*, **51**, 207 (1992).

[33] A. P. Morgan, A. J. Sommerse, and C. W. Wampler (1992), "A power series method for computing singular solutions to nonlinear analytic systems", *Numer. Math.*, **63**(3), 1779-1792.

[34] MPI: http://www.mpi-forum.org.

[35] V. W. Noonburg (1989), "A neural network modeled by an adaptive Lotka-Volterra system", *SIAM J. Appl. Math.*, **49**, 1779-1792.

[36] J. M. Rojas (1994), "A convex geometric approach to counting the roots of a polynomial system", *Theoret. Comput. Sci.*, **49**, 105-140.

[37] C. Traverso, The PoSSo test suite examples, Available at: www.inria.fr/saga/POL

[38] J. Verschelde (1999), "Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation", *ACM Trans. Math. Softw.*, **25**, 251-276.

[39] Y. Zhuang (2007), "Parallel Implementation of Polyhedral Homotopy Methods", Ph.D. thesis, Univesity of Illinois at Chicago.