HOM4PS-2.0para: Parallelization of HOM4PS-2.0 for Solving Polynomial Systems

Tien-Yien Li* and Chih-Hsiung Tsai†

Abstract

HOM4PS-2.0 is a software package in FORTRAN 90 which implements the polyhedral homotopy continuation method for solving polynomial systems. It leads in speed over the existing software packages in the same category by huge margins. This article details the description of the parallel version of HOM4PS-2.0, named HOM4PS-2.0para. 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.

AMS Subject Classification: 65H10, 65H20

Keywords: Polynomial systems, parallel computing, homotopy continuation methods, polyhedral homotopy, numerical experiments, software package.

1 Introduction

It has been established over the years that finding the full set of isolated solutions to a polynomial system numerically by the homotopy continuation method is reliable and efficient. See [**?**, **?**] for a survey. However, as the size of the polynomial system becomes larger, we need more computing resources to solve the system more efficiently. And a natural way to attain more computing resources is to parallelize the homotopy continuation method which seems to be naturally parallel in the sense that each isolated zero is computed independently of the others.

The state of the art of the homotopy continuation method for solving polynomial systems is the *polyhedral* homotopy continuation method initiated in [**?**]. It provides a big

*Department of Mathematics, Michigan State University, East Lansing, MI 48824, email: li@math.msu.edu. Research supported in part by NSF under Grant DMS-0811172.

†Department of Mathematics, Michigan State University, East Lansing, MI 48824, email: tsaichih@msu.edu.

computational advantage over the classical linear homotopy method in most occassions because the number of its homotopy curves is far less than that of the classical linear homotopies, especially for large and sparse systems. This method has successfully been implemented in the software packages PHCpack [**?**], PHoM [**?**] and HOM4PS-2.0 [**?**]. Among which, as shown in [**?**], HOM4PS-2.0 leads in speed by huge margins over the others. The speed-up ratios can reach over 100s for many large bench mark polynomial systems. The purpose of this article is to present a parallel implementation of HOM4PS-2.0, we call it HOM4PS-2.0para. Incidently, the description of the parallel implementation of PHoM, PHoMpara, has appeared in [**?**] and the parallelization of PHCpack was discussed in [**?**].

The polyhedral homotopy method for solving polynomial systems contains three main stages: finding mixed cells which provide starting points for the homotopy paths, following homotopy paths and checking for possible curve jumping. Those will be briefly reviewed in § 2. In § 3, we elaborate the parallelization of those three main stages implemented in HOM4PS-2.0. A common feature of those stages 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 [**?**]. This feature fits better to a simple master-workers parallel computation environment where one master CPU communicates with all worker CPUs and no communication between the workers. We use MPI [**?**] which provides message passing library for this type of communications between master and the workers in our parallelization.

For some polynomial systems, such as katsura-n [**?**] and reimer-n [**?**], the mixed volume of each system is the same as its total degree (or the Bézout number). Obviously, those systems should be solved directly by following the total degree number of solution paths of the classical linear homotopies rather than employing the polyhedral homotopies for solving them. As a major advantage, without using polyhedral homotopies requires no costly mixed cell computations, very costly for large systems indeed. Apparently, when such information that reveals the closeness between the mixed volume of the system and its total degree is known beforehand, the classical linear homotopy is importantly useful. Thus, the algorithm for solving polynomial systems by the classical linear homotopies has also been implemented as an option in HOM4PS-2.0, and details of its parallel version is included in § 3.

In § 4, numerical results on bench mark systems, economic-n [**?**], katsura-n [**?**], noon-n [**?**], reimer-n [**?**], and cyclic-n [**?**] are presented. Near perfect speed-ups in the results show 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.

2 The polyhedral continuation method

2.1 Computing the mixed cells

For a system of polynomials $P(x) = (p_1(x), \ldots, p_n(x))$ with $x = (x_1, \ldots, x_n)$, write

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

where $a = (a_1, \ldots, a_n) \in \mathbb{N}^n$, $c_{ja} \in \mathbb{C}^* = \mathbb{C}\backslash\{0\}$ and $x^a = x_1^{a_1} \cdots x_n^{a_n}$. Here S_j, a finite subset of $\mathbb{N}^n$, is called the *support* of $p_j(x)$, and $S = (S_1, \ldots, S_n)$ is called the *support* of $P(x)$. 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 $j = 1, \ldots, n$. For $\hat{\alpha} = (\alpha, 1) \in \mathbb{R}^{n+1}$, where $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{R}^n$, let $< \hat{a}, \hat{\alpha} >$ denote the usual inner product in Euclidean space. A collection of pairs $(\{a_{11}, a_{12}\}, \{a_{21}, a_{22}\}, \ldots, \{a_{n1}, a_{n2}\})$ where $\{a_{j1}, a_{j2}\} \subset S_j$ for each $j = 1, \ldots, n$, is called a *mixed cell* of $S = (S_1, \ldots, S_n)$ if there exists $\hat{\alpha} = (\alpha, 1) \in \mathbb{R}^{n+1}$ with $\alpha \in \mathbb{R}^n$ such that

$$< \hat{a}_{j1}, \hat{\alpha} > = < \hat{a}_{j2}, \hat{\alpha} > < < \hat{a}, \hat{\alpha} > \qquad \forall\ \hat{a} \in \hat{S}_j \setminus \{\hat{a}_{j1}, \hat{a}_{j2}\}, \quad j = 1, \ldots, n,$$

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

$$< \hat{a}, \hat{\alpha} > = < \hat{a}', \hat{\alpha} > \le < \hat{b}, \hat{\alpha} > \quad \forall\ b \in S_i \setminus \{a, a'\}.$$

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

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

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

$$< \hat{a}_{i_j}, \hat{\alpha} > = < \hat{a}'_{i_j}, \hat{\alpha} > \le < \hat{a}, \hat{\alpha} > \quad \forall\ a \in S_{i_j} \backslash \{a_{i_j}, a'_{i_j}\}.$$

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

An established strategy [**?**, **?**, **?**, **?**, **?**] for finding mixed cells is the extension of level-k subfaces $\widehat{E}_k$ of $\hat{S} = (\hat{S}_1, \ldots, \hat{S}_n)$ starting from $k = 1$ and when $k = n - 1$ an extensible subface $\widehat{E}_k$ yields mixed cells of $S = (S_1, \ldots, S_n)$ induced by elements in

$\hat{S}_{i_n}$. The software code MixedVol-2.0 [**?**] which we adopted in HOM4PS-2.0 for mixed cell computations implemented the novel idea of *dynamic enumeration* of all mixed cells in [**?**]. In which, to extend a particular level-k subface

$$\widehat{E}_k = (\hat{\mathbf{e}}_{i_1}, \ldots, \hat{\mathbf{e}}_{i_k}),\ 1 \le k < n, \quad \text{where } \hat{\mathbf{e}}_{i_j} = \{\hat{a}_{i_j}, \hat{a}'_{i_j}\} \in L(\hat{S}_{i_j}) \quad \text{for } j = 1, \ldots, k, \tag{2}$$

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 among them can possibly extend $\widehat{E}_k$ to a level-$(k+1)$ subface. The main strategy suggested in [**?**] 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 $\widehat{E}_k$, and select the one with minimal remaining points as $\hat{S}_{i_{k+1}}$.

2.2 The polyhedral-linear homotopy

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

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

let $Q(x) = (q_1(x), \ldots, q_n(x))$ be a polynomial system having the same monomials as $P(x)$ but with randomly chosen coefficients, *i.e.*, $q_j(x) = \sum_{a \in S_j} \bar{c}_{ja} x^a$ where $\bar{c}_{ja}$ are randomly chosen complex numbers. Originally, the first step of the polyhedral homotopy method is to solve this system by using a polyhedral homotopy $\widehat{Q}(x,t) = (\hat{q}_1(x,t), \ldots, \hat{q}_n(x,t)),\ t \in [0,1]$ with a random lifting given by $\omega = (\omega_1, \ldots, \omega_n),\ \omega_j : S_j \to \mathbb{R}$; *i.e.*, $\hat{q}_j(x,t) = \sum_{a \in S_j} \bar{c}_{ja} x^a t^{\omega_j(a)}$ for $j = 1, \ldots, n$. Then, by tracing homotopy solution paths of the linear homotopy $H(x,t) = (1-t)\,\gamma\, Q(x) + t\, P(x) = 0,\ t \in [0,1]$ with generically chosen $\gamma \in \mathbb{C}$, the so-called cheater's homotopy [**?**] (or coefficient-parameter continuation [**?**]), emanating from the solutions to $Q(x) = 0$ attained above, we will reach all isolated solutions of $P(x) = 0$ at $t = 1$.

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

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

where

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

so that the number of homotopy paths needed to be traced will be reduced by half. Note that $H(x,1) = P(x)$. Now, for a given mixed cell $(\{a_{11}, a_{12}\}, \{a_{21}, a_{22}\}, \ldots, \{a_{n1}, a_{n2}\})$ 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 $x = y t^{\alpha}$ 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 $j = 1, \ldots, n$

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

Letting

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

and "factoring out the lowest power of t" yields $\hat{H}(x,t) = (\hat{h}_1(x,t), \ldots, \hat{h}_n(x,t))$, $t \in [0,1]$, where

$$\hat{h}_j(x,t) = \sum_{a \in S_j} [(1-t)\bar{c}_{ja} + t c_{ja}] x^a t^{(<\hat{a}, \hat{\alpha}> - \beta_j)}, \quad \text{for } j = 1, \ldots, n. \tag{3}$$

Note that we still have $\hat{H}(x,1) = P(x)$, because $x(1) = y(1)$. To address the numerical difficulties this homotopy may encounter as indicated in [**?**], we apply the transformation $s = \ln t$ in (**??**), leading to the homotopy $\bar{H}(x,s) = (\bar{h}_1(x,s), \ldots, \bar{h}_n(x,s))$, $s \in (-\infty, 0]$, where

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

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

$$\begin{aligned} < \hat{a}_{j1}, \hat{\alpha} > &= < \hat{a}_{j2}, \hat{\alpha} >, \\ < \hat{a}_{j1}, \hat{\alpha} > &< < \hat{a}, \hat{\alpha} > \quad \forall\ \hat{a} \in \hat{S}_j \setminus \{\hat{a}_{j1}, \hat{a}_{j2}\}, \quad j = 1, \ldots, n. \end{aligned}$$

So, in each $\hat{h}_j(x,t)$ and hence in each $\bar{h}_j(x,s)$, there are exactly two powers equal to 0, namely, for each $j = 1, \ldots, n$,

$$\beta_j = \min_{a \in S_j} < \hat{a}, \hat{\alpha} > = < \hat{a}_{j1}, \hat{\alpha} > = < \hat{a}_{j2}, \hat{\alpha} > .$$

Therefore at $s = -\infty$, $\bar{H}(x, -\infty) = 0$ 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 $|\det(a_{11} - a_{12}, \ldots, a_{n1} - a_{n2})|$, the *volume* of mixed cell $C = (\{a_{11}, a_{12}\}, \ldots, \{a_{n1}, a_{n2}\})$, number of nonsingular isolated solutions which provide the starting points for following the solution paths of $\bar{H}(x,s) = 0$. See [**?**, **?**] for an efficient algorithm for solving binomial systems.

In the rest of this article, we shall call this combined homotopy the *polyhedral-linear* homotopy. This combined polyhedral-linear homotopy becomes particularly important for solving large problems whose mixed volumes exceed multi millions. Actually, combining linear and nonlinear homotopies into one step in the polyhedral homotopy method to reduce the number of solution paths needed to be followed by a half was originally suggested back in 1995 [**?**]. However, this strategy had not been successfully implemented in the original version HOM4PS of HOM4PS-2.0 because of the involved stability and efficiency problems. Addressing those difficulties by the transformation $s = \ln t$ and parametrizing the solution paths by $s \in (-\infty, 0]$ in HOM4PS-2.0 as above, a substantial improvement in algorithmic efficiency and stability has been achieved as evidenced by the results of intensive numerical experiments. (The same technique was independently developed in [**?**].)

2.3 On curve jumping

To check if curve jumping occurs during the process of path tracings, we must first verify *all* numerically attained solutions of the polynomial system to see if there are two solutions that are very close to each other. We may allow 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-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 $z = (z_1, \ldots, z_n) \in \mathbb{C}^n$ of the system, we 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 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 same 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 $\{0, 1, 2, \ldots, 9\}$. So in total, the solution set is divided into 200 groups. 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 sets.

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, and curve jumping clearly occurs. We will then retrace those 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

$$rank_{\mathbb{C}} \frac{\partial (p_1, \ldots, p_n)}{\partial (x_1, \ldots, x_n)} (z) = n - dim\ Z. \tag{4}$$

To differentiate those cases, the algorithm developed in [**?**] 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 (**??**). For the rank revealing of the Jacobian, we use the scheme developed in [**?**] rather than calculating the whole SVD (Singular Value Decomposition) of the matrix.

3 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. A common feature of those three stages, as mentioned before, is that each important task on a single CPU is 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 no communication between the workers, and use MPI [**?**], which provides message passing library for this type of communications between master and workers, in our implementation of the parallelization.

3.1 Parallelizing the computation of mixed cells

Recall that a main strategy for finding mixed cells is the extension of level-k subfaces $\widehat{E}_k = (\hat{\mathbf{e}}_{i_1}, \ldots, \hat{\mathbf{e}}_{i_k})$ of the lifted support $\hat{S} = (\hat{S}_1, \ldots, \hat{S}_n)$ starting from $k = 1$ and when $k = n-1$ an extensible $\widehat{E}_k$ 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 [**?**]. Thus each worker processor will be assigned to extend a lower edge of $L(\hat{S}_{i_1})$, and the works are 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 $L(\hat{S}_{i_1})$
$L(\hat{S}_{i_1}) = \{\hat{\mathbf{e}}_{11}, \ldots, \hat{\mathbf{e}}_{1N}\}$
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 $\hat{\mathbf{e}}_{1s}$)
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 $\hat{\mathbf{e}}_{1s}$)
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 $\{1, \ldots, n\}$ 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 $L(\hat{S}_{i_1})$
$L(\hat{S}_{i_1}) = \{\hat{\mathbf{e}}_{11}, \ldots, \hat{\mathbf{e}}_{1N}\}$

Step 0: Receive job s from the master
If $(s > N)$ then stop. (i.e., there are only N edges)
Otherwise, set $\widehat{F}_1 := \{\hat{\mathbf{e}}_{1s}\}, k := 1$.

Step 1: If $k = 0$, send all mixed cells to the master. Stop.
If $\widehat{F}_k = \emptyset$, set $k := k - 1$, and go to Step 1. Otherwise go to Step 2.

Step 2: Select $\hat{\mathbf{e}}_{i_k} \in \widehat{F}_k$, and set $\widehat{F}_k := \widehat{F}_k \backslash \{\hat{\mathbf{e}}_{i_k}\}$.
Let $\widehat{E}_k = (\hat{\mathbf{e}}_{i_1}, \ldots, \hat{\mathbf{e}}_{i_k})$, where $\hat{\mathbf{e}}_{i_1} = \hat{\mathbf{e}}_{1s}$, and
$\hat{\mathbf{e}}_{i_j} = \{\hat{a}_{i_j}, \hat{a}'_{i_j}\} \in L(\hat{S}_{i_j})$ 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 $\widehat{E}_k$ to a level-$(k+1)$ subface.
Find $\varepsilon(\widehat{E}_k)$ (= The collection of all lower edges in $L(\hat{S}_{i_{k+1}})$ that extends $\widehat{E}_k$.)
If $\varepsilon(\widehat{E}_k) = \emptyset$, go to Step 1. Otherwise set $\widehat{F}_{k+1} = \varepsilon(\widehat{E}_k)$, $k := k + 1$, then go to Step 4.

Step 4: If $k = n$, all $C = (\hat{\mathbf{e}}_{i_1}, \ldots, \hat{\mathbf{e}}_{i_{n-1}}, \hat{\mathbf{e}}_{i_n})$ with $\hat{\mathbf{e}}_{i_n} \in \widehat{F}_n$ are mixed cells. Collect all these mixed cells, set $k := k - 1$, and go to Step 1. Otherwise go to Step 2.

3.2 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 [**?**] 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 $\dfrac{MV}{100(p-1)}$ 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 associate binomial system

$$\begin{array}{rcl} c_{11}x^{a_{11}} + c_{12}x^{a_{12}} & = & 0, \\ \vdots & & \\ c_{n1}x^{a_{n1}} + c_{n2}x^{a_{n2}} & = & 0, \end{array} \tag{5}$$

in the following manner. Equivalent to (**??**), write

$$\begin{array}{rcl} x^{a_{11}-a_{12}} & = & b_1, \\ \vdots & & \\ x^{a_{n1}-a_{n2}} & = & b_n, \end{array} \tag{6}$$

with $b_j = -\dfrac{c_{j2}}{c_{j1}}$ for $j = 1, 2, \ldots, n$. When solving (**??**) [**?**, **?**], 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

$$z^{BA} = (b_1, \ldots, b_n), \tag{7}$$

then $x = z^B$ is a solution to (**??**). Writing out explicitly, (**??**) becomes

$$\begin{array}{rcl} z_1^{v_{11}} & = & b_1, \\ z_1^{v_{12}} z_2^{v_{22}} & = & b_2, \\ \vdots & & \\ z_1^{v_{1n}} \cdots z_n^{v_{nn}} & = & b_n. \end{array} \tag{8}$$

Without loss of generality, we may assume $v_{jj} > 0$ for all $j = 1, \ldots, n$. Now, v_{11} solutions of z_1 in (**??**) can be expressed as

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

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

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

where $\bar{b}_j = \frac{b_j}{z_1^{v_{1j}} \cdots z_{j-1}^{v_{j-1j}}}$ and $arg\,(\bar{b}_j)$ is the principle argument of $\bar{b}_j$. So the system in (**??**) 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 $\underline{l^{th}}$ solution of the system in (**??**). For the corresponding solution $x = z^B$ to the system in (**??**), we label it with the same order index of the solution z to the system in (**??**).

The polyhedral-linear homotopy paths can now be ordered by taking the same indexes of their starting points of the associated binomial systems together with the number indexes of the corresponding mixed cells. For this purpose, a table T 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 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 Volume of $C_1 = 8$, Volume of $C_2 = 12, \ldots,$ Volume of $C_{30} = 8$, then the table T is:

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

Table T

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

$\begin{array}{|c|}\hline 1 \\ \hline 1 \\ \hline 5 \\ \hline \end{array}$. Meaning, the worker that receives job 1 will follow paths emanated from number 1 through number 5 solutions of the binomial system associated with cell 1. Similarly, job 2 associates with table $T(2)$ 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.

In addition to those 300 jobs, we add three empty jobs 301, 302 and 303 representing the signal for no more paths to follow. When a worker finishes tracing its 5 assigned paths, it reports the computation result to the master, and the master will send a new job i along with table $T(i)$ in the queue to this worker. For example, if worker 2 finishes its job earlier than worker 1 and 3, then the next open job in the queue will be sent to worker 2. If worker 3 finishes its job after worker 2, then the next job will be sent to worker 3, etc. There are only 300 jobs in this example, so when any worker receives a job i with $i > 300$ will stop and wait for the other workers to finish their jobs.

We summarize the above in the next two algorithms.

Algorithm for the master

Establish Table T consisting of two rows in which each entry in the first row represents the index of a mixed cell and each entry in the second row denotes the index of the solution of the associated binomial system of the mixed cell

$NumOfJobs = \dfrac{MV}{CS}$ (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 $T(i)$) to worker i
end do
do $i = p$ to $NumOfJobs + p - 1$
    receive results from worker j
    send job i (along with table $T(i)$) to worker j
end do

Algorithm for worker

receive job i from the master
if $(i > NumOfJobs)$, then
  stop
else
  receive table $T(i)$ with 3 rows and $T(1, i*CS) - T(1, (i-1)*CS+1) + 1$ columns
  entries in 1^{st} row are the indexes of the mixed cells

entries in 2^{nd} row are the beginning indexes of the chunk
entries in 3^{rd} row are the ending indexes of the chunk
worker solves the corresponding binomial systems and follows CS paths based on $T(i)$
send results to the master
end if

When a worker receives job i along with table $T(i)$ 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.

3.3 Parallelizing the curve tracing of the linear homotopy

For certain systems, such as katsura-n [**?**] and reimer-n [**?**], the polyhedral-linear homotopy method provides no advantages when it is used for solving them, because the mixed volume of each such system is the same as its total degree (or the Bézout number). 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 $Q(x) = (q_1(x), \ldots, q_n(x))$ with

$$\begin{array}{rcll} q_1 & = & a_1 x_1^{d_1} - b_1, & d_1 = \text{degree of } p_1(x), \\ & \vdots & & \\ q_n & = & a_n x_n^{d_n} - b_n, & d_n = \text{degree of } p_n(x), \end{array} \tag{9}$$

and randomly chosen $(a_1, \ldots, a_n, b_1, \ldots, b_n) \in \mathbb{C}^{2n}$ [**?**]. In this way, with no polyhedral homotopies involved requires no costly mixed cell computations. The algorithm for solving polynomial systems by this classical linear homotopy was implemented as an option in HOM4PS-2.0. We now describe the parallel algorithm in our HOM4PS-2.0para for this homotopy.

While the starting system $Q(x)$ in (**??**) is a special binomial system as in (**??**), 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 $\mathrm{CS} = \dfrac{TotalDeg}{100(p-1)}$ 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

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

with $\bar{b}_j = \dfrac{b_j}{a_j}$ and $arg\,(\bar{b}_j)$ being the principle 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

$$l - 1 = u_1\,(d_2 \times \cdots \times d_n) + u_2\,(d_3 \times \cdots \times d_n) + \cdots + u_{n-1}(d_n) + u_n \tag{10}$$

with $0 \le u_j < d_j$ for $j = 1, \ldots, n$. So, as before, $(x_1^{(u_1)}, \ldots, x_n^{(u_n)})$ will be the $l^{\underline{th}}$ solution of the starting system $Q(x) = 0$, and homotopy paths will be ordered by taking the same indexes of their starting points, respectively, as solutions of $Q(x) = 0$. With this order arrangement, the worker that receives job i for $1 \le i \le \dfrac{TotalDeg}{\mathrm{CS}}$ will follow homotopy paths from number $(i-1)\mathrm{CS}+1$ to number $i\,\mathrm{CS}$, 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 $d = \max\{d_1, \ldots, d_n\}$, in which the $j^{\underline{th}}$ row stores the d_j solutions $x_j^{(0)}, \ldots, x_j^{(d_j-1)}$ of the equation $a_j\,x_j^{d_j} - b_j = 0$ for $j = 1, \ldots, n$. As an example, for $n = 3$, $d_1 = 8$, $d_2 = 5$ and $d_3 = 10$, the 3×10 Bixlib table is:

Table Bixlib

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

When table Bixlib is established, one can easily identify the $l^{\underline{th}}$ solution $(x_1^{(u_1)}, \ldots, x_n^{(u_n)})$ of $Q(x) = 0$ as long as $u_1, \ldots, u_n$ in the decomposition of $l-1$ in (**??**) 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)
$NumOfJobs = \dfrac{TotalDeg}{CS}$ ($TotalDeg$ 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 indexes of paths
   for each integer between the starting and ending indexes
   find the corresponding starting point from table Bixlib to follow the path
   send results to the master
end if

3.4 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 § 2.3, 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. Workers will check each group to find pairs of solutions that are numerically identical. Note that solutions may not be evenly distributed, 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 1000 solutions, then this specific group will be divided into 200 groups again based on the imaginary part of the second component of each solution in the group. Let $g_1, \ldots, g_{200}$ denote the original 200 groups. The dynamic distribution algorithms for the master and workers for checking the curve jumping described in § 2.3 are listed follow.

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

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

do $i = p$ to $200 + p - 1$
  receive computation results from some worker j
  send job i to worker j
end do

(**Comment**: For $i = 1, \ldots, 200$, job i consists of checking solutions in group g_i. Job 201, job 202, ..., job $200 + p - 1$ are empty jobs.)

Algorithm for the workers

receive job i from the master

if ($i > 200$) stop

if ($i \leq 200$) then
  receive $g_i = \{x_{i1}, \ \ldots \ , \ x_{i|g_i|}\}$
end if

if ($|g_i| > 1$ and $|g_i| \leq 1000$), then
  do $l = 1$ to $|g_i| - 1$
    do $m = l + 1$ to $|g_i|$
      if $\dfrac{\| x_{il} - x_{im} \|}{\| x_{il} \|} <$ a chosen parameter ϵ_0, then
        if the minimum singular value of the Jacobian matrix $P_x(x_{il})$ is bigger than a chosen threshold ϵ_1, then retrace the two paths with smaller step sizes
          send the updated solutions to the master
        else
          if x_{il} is a nonsingular point, then
            curve jumping occurs
            retrace the two paths with smaller step sizes
          end if
        end if
      end if
    end do
  end do

else if ($|g_i| > 1000$) then
  divide $|g_i|$ into 200 groups $g_{i\,1}, \ldots, g_{i\,200}$ based on the imaginary part of second component of each solution in g_i
  do $j = 1$ to 200
    let $g_{ij} = \{x_{ij1}, \ \ldots \ , \ x_{ij|g_{ij}|}\}$

```
      do l = 1 to |g_ij| − 1
         do m = l + 1 to |g_ij|
            if ‖ x_ijl − x_ijm ‖ / ‖ x_ijl ‖ < a chosen parameter ε_0, then
               if the minimum singular of the Jacobian matrix P_x(x_ijl) is bigger than
                  a chosen threshold, then
                  retrace the two paths with smaller step sizes
                  send the updated solutions to the master
               else
                  if x_ijl is a nonsingular point, then
                     curve jumping occurs
                     retrace the two paths with smaller step sizes
                  end if
               end if
            end if
         end do
      end do
   end do
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 §3.3. 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 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 indexes 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 $P(x) = 0$, they are saved in a 2-dimensional array 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. 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 [**?**], 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. On the other hand, when curve jumping is detected, to retrace the homotopy paths and updating the new solutions, it is much convenient if solutions are stored in an array.

eco-n [**?**] Total degree $= 2 \cdot 3^{n-2}$

$$
\begin{gathered}
(x_1 + x_1x_2 + \cdots + x_{n-2}x_{n-1})x_n - 1 = 0 \\
(x_2 + x_1x_3 + \cdots + x_{n-3}x_{n-1})x_n - 2 = 0 \\
\vdots \\
x_{n-1}x_n - (n-1) = 0 \\
x_1 + x_2 + \cdots + x_{n-1} + 1 = 0
\end{gathered}
$$

noon-n [**?**] Total degree $= 3^n$

$$
\begin{gathered}
x_1(x_2^2 + x_3^2 + \cdots + x_n^2 - 1.1) + 1 = 0 \\
x_2(x_1^2 + x_3^2 + \cdots + x_n^2 - 1.1) + 1 = 0 \\
\vdots \\
x_n(x_1^2 + x_2^2 + \cdots + x_{n-1}^2 - 1.1) + 1 = 0
\end{gathered}
$$

cyclic-n [**?**] Total degree $= n\,!$

$$
\begin{gathered}
x_1 + x_2 + \cdots + x_n = 0 \\
x_1x_2 + x_2x_3 + \cdots + x_{n-1}x_n + x_nx_1 = 0 \\
x_1x_2x_3 + x_2x_3x_4 + \cdots + x_{n-1}x_nx_1 + x_nx_1x_2 = 0 \\
\vdots \\
x_1x_2 \cdots x_n - 1 = 0
\end{gathered}
$$

katsura-n [**?**] Total degree $= 2^n$

$$
\begin{gathered}
2x_{n+1} + 2x_n + \cdots + 2x_2 + 2x_1 - 1 = 0 \\
2x_{n+1}^2 + 2x_n^2 + \cdots + 2x_2^2 + x_1^2 - x_1 = 0 \\
2x_nx_{n+1} + 2x_{n-1}x_n + \cdots + 2x_2x_3 + 2x_1x_2 - x_2 = 0 \\
2x_{n-1}x_{n+1} + 2x_{n-2}x_n + \cdots + 2x_1x_3 + x_2^2 - x_3 = 0 \\
\vdots \\
2x_2x_{n+1} + 2x_1x_n + 2x_2x_{n-1} + \cdots + 2x_{n/2}x_{(n+2)/2} - x_n = 0 \text{ (if } n \text{ is even)} \\
2x_2x_{n+1} + 2x_1x_n + 2x_2x_{n-1} + \cdots + x_{(n+1)/2}^2 - x_n = 0 \text{ (if } n \text{ is odd)}
\end{gathered}
$$

reimer-n [**?**] Total degree $= (n+1)\,!$

$$
\begin{gathered}
2x_1^2 - 2x_2^2 + \cdots + (-1)^{n+1}2x_n^2 - 1 = 0 \\
2x_1^3 - 2x_2^3 + \cdots + (-1)^{n+1}2x_n^3 - 1 = 0 \\
\vdots \\
2x_1^{n+1} - 2x_2^{n+1} + \cdots + (-1)^{n+1}2x_n^{n+1} - 1 = 0
\end{gathered}
$$

Table A The polynomial systems

4 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 A. All the computations were carried out on a cluster 8 AMD dual 2.2 GHz cpus. We only list those systems that can be solved within 12 hours cpu time.

4.1 The performance on large systems

Table 1 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, while, as reported in [**?**], multiple retracings were required very often for the software package PHoM [**?**] to deal with curve jumping.

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

Table 2 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 to the mixed volume of the system. Obviously 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 $2n$ [**?**], which becomes much slimmer relatively as n becomes larger. We therefore 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 2: Solving systems by the classical linear homotopy with 1 master and 7 workers

4.2 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 3 shows the scalability of solving eco-16 and cyclic-11 by the polyhedral-linear homotopy and Table 4 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 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 are very close to being perfect. Notably illuminating is the almost perfect scalability in solving the whole system of cyclic-11 in Table 3. This shows that when the polyhedral-linear homotopy is used our strategy for the dynamic workload distribution is quite balanced.

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

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: The scalability of solving systems by the classical linear homotopy

4.3 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 the paths.

4.4 Conclusion and future work

Polyhedral-linear and classical linear homotopies can be used to solve large polynomial systems efficiently. With multi-processors, HOM4PS-2.0para can solve systems that have not been solved in the past within tolerable time. As we can see from Table 3, the ratios of computing mixed cells do not exhibit near perfect scalabilities as the ratios of tracing cures 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.

References

[1] 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.

[2] 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.

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

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

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

[6] T. Gunji, S. Kim, M. Kojima, A. Takeda, K. Fujisawa and T. Mizutani (2004), "PHoM - A polyhedral homotopy continuation method", *Computing*, **73**, 57-77.

[7] 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.

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

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

[10] 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.

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

[12] 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).

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

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

[15] 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.

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

[17] 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.

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

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

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

[21] 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).

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

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

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

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

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