HOM4PS-2.0: A software package for solving polynomial systems by the polyhedral homotopy continuation method

Tsung-Lin Lee[*] Tien-Yien Li[†] and Chih-Hsiung Tsai[‡]

January 15, 2008

Abstract

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) Combining the polyhedral and linear homotopies in one step, (3) New way of dealing with the curve jumping. Numerical results show that this revision leads to a spectacular speed-up, ranging up to the 50's, over its original version on all bench mark systems, especially for large ones. It surpasses established packages in this category, such as PHCpack [25] and PHoM [8], in speed by huge margins.

1 Introduction

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

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

where $a = (a_1, \ldots, a_n) \in \mathbb{N}^n$, $c_{j,a} \in \mathbb{C}^* = \mathbb{C} \backslash \{0\}$ 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 $P(x)$.

[*]Department of Mathematics, Michigan State University, East Lansing, MI 48824, email: leetsung@msu.edu.

[†]Department of Mathematics, Michigan State University, East Lansing, MI 48824, email: li@math.msu.edu. Research supported in part by NSF under Grant DMS-0411165.

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

Based on Bernshtein's combinatorial root count [1], the *polyhedral homotopies* are established in 1995 [9] to approximate all the isolated zeros of $P(x) = (p_1(x), \ldots, p_n(x))$ by the homotopy continuation method. It yields a drastic improvement over the classical linear homotopies for solving sparse polynomial systems. The software package HOM4PS, developed over the years by a group led by T.Y.Li at Michigan State University, implemented this approach for solving polynomial systems. While the detailed description of the algorithms in HOM4PS was not formally published, the code is widely considered to be much superior in speed to the published codes PHCpack [25] and PHoM [8] which also implemented the polyhedral homotopy method for solving polynomial systems.

In this article, we shall elaborate the details of the most updated version of HOM4PS, we call it HOM4PS-2.0, which greatly upgrades its original version, leading to a spectacular speed-up as shown in § 6.

There are three major revisions in HOM4PS-2.0:

- **Mixed cell computations**

  When the polyhedral homotopy is employed to find all isolated zeros of $P(x) = (p_1(x), \ldots, p_n(x))$, the process of locating all the *mixed cells* during the mixed volume computation plays a crucially important role [13, 14, 15]: 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. Calculating the mixed cells (and thus the mixed volume) of the support of $P(x)$ consumes a large part of the computation and therefore dictates the efficiency of the method as well as the scope of its applications. In 2005, a software package, **MixedVol** [7], emerged which led the existing codes for the mixed volume computation by a great margin in speed. However, soon after **MixedVol** was published, T. Mizutani, A. Takeda and M. Kojima [19] developed a more efficient algorithm which overshadowed **MixedVol** in speed by a big amount. Most recently T.L.Lee and T.Y.Li [12] embedded the novel idea of *dynamic enumerations* of mixed cells in [19] into **MixedVol** and a new code, **MixedVol-2.0**, was produced which regains the lead by a substantial margin. Naturally, we adopt this new algorithm for the mixed cell computation in HOM4PS-2.0 and will outline the algorithm in § 2.

- **Combining the polyhedral and linear homotopies in one step**

  The polyhedral homotopy method implemented in HOM4PS for solving polynomial system $P(x) = (p_1(x), \ldots, p_n(x))$ consists of two main steps: (1) A polynomial system $Q(x) = (q_1(x), \ldots, q_n(x)) = 0$ having the same monomials as $P(x)$ but with randomly chosen coefficients is solved by polyhedral homotopies. (2) A linear homotopy $H(x,t) = (1-t)\gamma Q(x) + tP(x)$, $t \in [0,1]$ with generically chosen $\gamma \in \mathbb{C}$, is used to find all isolated zeros of $P(x)$. By Bernshtein's theorem [1], the number of solution paths which need to be traced in the first step is the mixed volume of the support of $P(x)$. This number remains the same for the second step. All together the total number of homotopy paths that need to be traced is twice the mixed volume of the support of $P(x)$. In HOM4PS-2.0, those two steps were

combined in one step so that the total number of curves that need to be traced is simply the mixed volume of the support of $P(x)$. While this idea was originally suggested in [9], it had not been successfully implemented in HOM4PS because of the involved numerical difficulties in efficiency and stability. In HOM4PS-2.0, we successfully dealt with the problems by applying the transformation $s = \ln t$, yielding a family of smooth solution paths of a homotopy $\bar{H}(x, s) = 0$, each parametrized by $s \in (-\infty, 0]$. We will explain the details in § 3. This strategy was independently developed in [10].

- **Dealing with the 'Curve Jumping'**

  Theoretically, with probability 1, two homotopy paths do not cross each other. But, in practice, when tracing two homotopy paths that are very close to each other, it is possible that the *curve jumping* can happen and may thus result in the missing of zeros. Though a sophisticatedly designed path tracing algorithm seldom encounters this problem, very seldom indeed, we have zero tolerance for the occurrence of a missing zero. In HOM4PS, the numerically attained zeros were compared with each other in a quite straightforward manner to determine possible curve jumping, followed by retracing the corresponding paths with smaller step sizes. The detecting method used there could potentially cost a substantial amount of computing time for large systems where over millions of homotopy paths need to be traced. In the new version, HOM4PS-2.0, solutions are divided into many groups so that only solutions in the same group need to be compared with each other. This technique provides quite a big saving in the computing time, especially for large systems. More details follow in § 4.

In addition to those key revisions given above, § 5 lists several new aspects in our algorithms in HOM4PS-2.0 which appeared in HOM4PS in a less sophisticated manner. This includes schemes in evaluating polynomials and their partial derivatives in § 5.1, in scaling the coefficients of the polynomial systems in § 5.2 and in treating the end games in § 5.3. Numerical results are shown in § 6. The speed-ups of our new package HOM4PS-2.0 over its original version HOM4PS on those bench mark polynomial systems are shown in § 6.2. As the speed-up increases when the size of the system becomes bigger, it can reach over 50's for large systems. Furthermore, HOM4PS-2.0 can handle much larger systems in a tolerable time than its original version. In § 6.3, we compare our new code with the existing packages, PHCpack [25] and PHoM [8] which also implemented the polyhedral homotopy method for solving polynomial systems. As exhibited, HOM4PS-2.0 leads in speed by huge margins.

2 Mixed cell computations

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 $j = 1, \ldots, n$. 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 $\{a_1, a_1'\} \subset S_1, \ldots, \{a_n, a_n'\} \subset S_n$ 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

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

and α is called the *inner normal* of this mixed cell. For $1 \le i \le n$, $\hat{\mathbf{e}} = \{\hat{a}, \hat{a}'\} \subset \widehat{S}_i$, is called *a lower edge of* $\widehat{S}_i$ if there exists $\hat{\alpha} = (\alpha, 1) \in \mathbb{R}^{n+1}$ 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 $\widehat{S}_i$ by $L(\widehat{S}_i)$. For k distinct integers $\{i_1, \ldots, i_k\} \subset \{1, \ldots, n\}$,

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

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 $j = 1, \ldots, k$

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

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

A main strategy for finding mixed cells is the extension of level-k subfaces $\widehat{E}_k$ of $\hat{S} = \left(\hat{S}_1, \ldots, \hat{S}_n\right)$ starting from $k = 1$ and an extensible $\widehat{E}_k$ when $k = n - 1$ yields mixed cells of $S = (S_1, \ldots, S_n)$, induced by elements in $\widehat{S}_{i_n}$. In [5, 6, 7, 16], the *order* of this extension $i_1, i_2, \ldots$ is predetermined and fixed, that is, one always starts from extending lower edge $\hat{\mathbf{e}}_1$ of $\hat{S}_1$ to a level-2 subface $(\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2)$ with $\hat{\mathbf{e}}_2 \in L(\widehat{S}_2)$, then extend $(\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2)$ to a level-3 subface $(\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2, \hat{\mathbf{e}}_3)$ with $\hat{\mathbf{e}}_3 \in L(\widehat{S}_3) \ldots$ etc. Software package **MixedVol** [7] for the mixed cells computation that was adopted in the original HOM4PS was developed along this line of approach.

In [19], 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

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

one searches among $M := \{ \widehat{S}_l : l \in \{1, \ldots, n\} \backslash \{i_1, \ldots, i_k\}\}$ for $\widehat{S}_{i_{k+1}}$ that has minimal number of suitable points where only lower edges consisting of points among

them can possibly extend $\widehat{E}_k$ to a level-$(k+1)$ subface. The main strategy suggested in [19] for finding such $\widehat{S}_{i_{k+1}}$ is the removal of those points in each $\widehat{S}_l \in M$ which have no chances to be part of a lower edge in $L(\widehat{S}_l)$ that can extend $\widehat{E}_k$, and select the one with minimal remaining points as $\hat{S}_{i_{k+1}}$.

For level-k subface $\widehat{E}_k = (\hat{\mathbf{e}}_{i_1}, \ldots, \hat{\mathbf{e}}_{i_k})$ where $\hat{\mathbf{e}}_{i_j} = \{\hat{a}_{i_j}, \hat{a}'_{i_j}\} \in L(\widehat{S}_{i_j})$ 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 $\widehat{S}_l$, and consider the set of constraints:

$$\begin{array}{rcll} \langle \widehat{a}_\mu, \widehat{\alpha} \rangle & = & \langle \widehat{a}'_\mu, \widehat{\alpha} \rangle & \forall \ \ \mu \in Q \\ & \leq & \langle \widehat{a}, \ \widehat{\alpha} \rangle & \forall \ \widehat{a} \in \widehat{S}_\mu \backslash \{\widehat{a}_\mu, \widehat{a}'_\mu\} \\ \langle \widehat{b}_v, \widehat{\alpha} \rangle & \leq & \langle \widehat{b}, \widehat{\alpha} \rangle & \forall \ \ \widehat{b} \in \widehat{S}_l \backslash \{\widehat{b}_v\} \end{array} \tag{3}$$

in terms of unknowns $\widehat{\alpha} = (\alpha, 1) \in \mathbb{R}^{n+1}$. Apparently, when the set of constraints in (3) is infeasible, then there is no $\hat{b}_\mu$ in $\widehat{S}_l$ such that $\{\hat{b}_v, \hat{b}_\mu\}$ can extend $\widehat{E}_k$ 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), consider the linear programming (LP) problem

$$\begin{array}{ll} \text{(P)} & \text{Max} \quad \langle \mathbf{r}, \alpha \rangle \\ & \text{Subject to} \quad (3) \end{array}$$

where $\mathbf{r} \in \mathbb{R}^n$ is any fixed vector. By the duality theorem, the feasibility of the inequalities in (3) 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 [19]. 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 [12]. Employing the polyhedral homotopy method for solving polynomial systems, locating all mixed cells always plays a critically important role [13, 14, 15]. 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 Constructing the polyhedral-linear homotopy

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

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

let $Q(x) = (q_1(x), \ldots, q_n(x)) = 0$ be a polynomial system having the same monomials as $P(x)$ but with randomly chosen coefficients, i.e., $q_j(x) = \sum_{a \in S_j} \bar{c}_{ja} x^a$ where $\bar{c}_{ja}$ are randomly chosen complex numbers. In HOM4PS this system is first solved 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 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, the so-called cheater's homotopy [17] (or the coefficient-parameter continuation [21]). It is known that all isolated solutions of $P(x) = 0$ can be obtained by following the smooth solution paths of $H(x,t) = 0$ emanating from the solutions to $Q(x) = 0$ found above.

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

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

where

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

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

$$\hat{h}_j(x,t) = \sum_{a \in S_j} [(1-t)\bar{c}_{ja} + t c_{ja}] x^a t^{(<\hat{a}, \hat{\alpha}> - \beta_j)}, \quad \text{for } j = 1, \ldots, n. \tag{4}$$

Note that we still have $\hat{H}(x,1) = P(x)$, because $x(1) = y(1)$. 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 (4) 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 $100,000 * t^{99999}$ 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 this sort 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, the difference between the computing time of these two approaches is almost negligible most of the time notwithstanding the number of paths which need to be followed differs by a half between them.

In HOM4PS-2.0, we address this problem by applying the transformation $s = \ln t$ in (4), resulting in 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

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

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

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

Therefore at $s = -\infty$, $\bar{H}(x, -\infty)$ 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 $|\det(a_{11} - a_{12}, \ldots, a_{n1} - a_{n2})|$ 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 not detail the standard procedure for solving binomial systems here, see [13, 15].

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

$$\frac{d\bar{H}}{ds}(x(s), s) = \frac{d}{ds}\hat{H}(x, e^s) = \hat{H}_x \frac{dx}{ds} + \hat{H}_t e^s = 0. \tag{5}$$

It follows that $\dfrac{dx}{ds} \mid_{s=-\infty} = 0$ 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 $e^{\,s_0*(<\hat{a},\hat{\alpha}>-\beta_j)}$ for $a \in S_j \setminus \{a_{j1}, a_{j2}\}$ 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 $s*(<\hat{a},\hat{\alpha}> - \beta_j)$ in the polynomial

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

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

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

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

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

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 (5),

$$\frac{dx}{ds} = -\hat{H}_x^{-1} \hat{H}_t \, e^s, \quad \text{with } \hat{H}(x, t) \text{ as in (4) and } t = e^s.$$

So $\frac{dx}{ds}$ is small for large negative s_0, which justifies the adoption of a large step size at s_0, say $\delta_0 = \frac{-s_0}{3}$, 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 $\delta_k = \min\{\delta_{k-1}, \frac{-s_k}{3}\}$, 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 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 [9]. However, this idea has not been 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.

4 On curve jumping

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 1 in § 6.1, 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 $z = (z_1, \ldots, z_n) \in \mathbb{C}^n$ 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 $\{0, 1, 2, \ldots, 9\}$. 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

$$rank_{\mathbb{C}} \frac{\partial (p_1, \ldots, p_n)}{\partial (x_1, \ldots, x_n)} (z) = n - dim\ Z. \tag{6}$$

To differentiate those cases, the algorithm developed in [11] 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 (6). For the rank revealing of the Jacobian, we use the scheme developed in [18] 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 [11]. Curve jumping is only allowed to exist if $\bar{x}$ is nonsingular.

5 Miscellaneous

5.1 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 $j = 1, \ldots, n$,

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

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 $P = (p_1(x_1,x_2,x_3), p_2(x_1,x_2,x_3), p_3(x_1,x_2,x_3))$ where

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

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 $\bar{H}_x(x,s)$.

Table T

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

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 monomials $x_n, x_n^2, \ldots, x_n^{MaxDeg(n)}$. Since $\bar{H}_x(x,s)$ contains the partial derivatives of $x^a =$

$x_1^{a_1} \cdots x_n^{a_n}$ that appear in $\bar{H}(x,s)$, and since $\bar{H}_s(x,s)$ contains the same $x^a = x_1^{a_1} \cdots x_n^{a_n}$ 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 $\bar{H}_x(x,s)$. The value of a monomial evaluated at a point $x = (x_1, \ldots, x_n)$ can easily be obtained from table T. For example, for $n = 3$, the quantity of $x_1^2 x_2^3 x_3$ is $T(1,2) * T(2,3) * T(3,1)$.

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

Table T

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	

and those powers that are involved in any monomial evaluations, no matter how often they appear, will never be repeatedly computed.

5.2 Scaling of the coefficients

Certain polynomial systems, such as cohn 2 and cohn 3 [4], have large coefficients. Large magnitudes in coefficients will result in large magnitudes in tangent vectors of the homotopy paths. This will effect 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 [20]. We shall illustrate our scaling method by way of an example.

Example Consider the following system of two equations in two unknowns

$$\begin{aligned} 8000\, x_1^2 x_2^2 - 2000\, x_1 + 1 &= 0 \qquad (7) \\ 5000\, x_1 x_2 - 30 &= 0. \qquad (8) \end{aligned}$$

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 (7) by 10^{c_3} and multiply (8) by 10^{c_4}. This gives

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

Or,

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

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,

$$\begin{aligned} r &= (2c_1 + 2c_2 + c_3 + \log(8000))^2 + (c_1 + c_3 + \log(2000))^2 + c_3^2 \\ &\quad +(c_1 + c_2 + c_4 + \log(5000))^2 + (c_4 + \log(30))^2 \\ &\quad +(c_1 + 2c_2 + \log(8000) - \log(2000))^2 + (2c_1 + 2c_2 + \log(8000))^2 \\ &\quad +(c_1 + \log(2000))^2 + (c_1 + c_2 + \log(5000) - \log(30))^2. \end{aligned} \tag{9}$$

While in [20] 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 (9) as

$$\begin{aligned} r &= \left\| \underbrace{\begin{pmatrix} 2 & 2 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}}_{\mathrm{A}} \underbrace{\begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix}}_{x} - \underbrace{\begin{pmatrix} -\log(8000) \\ -\log(2000) \\ 0 \\ -\log(5000) \\ -\log(30) \\ \log(2000) - \log(8000) \\ -\log(8000) \\ -\log(2000) \\ \log(30) - \log(5000) \end{pmatrix}}_{b} \right\|_2^2 \\ &= \| Ax - b \|_2^2 , \end{aligned} \tag{10}$$

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 $x_2 = 10^{c_2} z_2$.

5.3 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 [24] 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

$$x_j(s) = a_j(1 - e^s)^{\frac{\omega_j}{m}}(1 + \sum_{i=1}^{\infty} a_{ij}(1 - e^s)^{\frac{i}{m}}), \quad j = 1, \ldots, n \tag{11}$$

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. [22] 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 (11) yields

$$\log |x_j(s)| = \log |a_j| + \frac{\omega_j}{m} \log(1 - e^s) + \sum_{i=1}^{\infty} b_{ij}(1 - e^s)^j. \tag{12}$$

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 (12),

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

Therefore $\dfrac{\omega_j}{m}$ 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 [25], 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.

6 Numerical results

To demonstrate the performance of HOM4PS-2.0 and to compare it with the existing packages HOM4PS, PHCpack [25] as well as PHoM [8] which implemented the polyhedral homotopy method, we will focus on those size-expandable bench mark systems listed in Table A. 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 code as well as its Matlab interface are available at: `http://www.math.msu.edu/~li/Software.htm`

For some of the systems listed in Table A, 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, in such situations, rather than employing the polyhedral homotopy for solving those systems, they should be solved directly by following the total degree number of solution paths of the classical linear homotopies $H(x,t) = (1-t)\gamma\, Q(x) + t\, P(x) = 0$ with generic $\gamma \in \mathbb{C}$ where $Q(x) = (q_1(x), \ldots, q_n(x))$ with

eco-n [20] 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 [23] 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 $= 2^n$
$2x_{n+1} + 2x_n + \cdots + 2x_2 + 2x_1 - 1 = 0$ $2x_{n+1}^2 + 2x_n^2 + \cdots + 2x_2^2 + x_1^2 - x_1 = 0$ $2x_nx_{n+1} + 2x_{n-1}x_n + \cdots + 2x_2x_3 + 2x_1x_2 - x_2 = 0$ $2x_{n-1}x_{n+1} + 2x_{n-2}x_n + \cdots + 2x_1x_3 + x_2^2 - x_3 = 0$ $\vdots$ $2x_2x_{n+1} + 2x_1x_n + 2x_2x_{n-1} + \cdots + 2x_{n/2}x_{(n+2)/2} - x_n = 0$ (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 [24] 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 A The polynomial systems

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

and randomly chosen $(a_1, \ldots, a_n, b_1, \ldots, b_n) \in \mathbb{C}^{2n}$ [15]. In this way, with no polyhedral homotopy method involved in solving the systems requires no costly, very costly for large systems indeed, mixed cell computations. Moreover, homotopies being straightly linear in the homotopy parameter t may avoid the $s = ln\, t$ transformation for the parameter used in the linear-polyhedral homotopy combinations. This can also reduce a considerable amount of cpu time for large systems. We also implemented the algorithm for solving polynomial systems by the classical linear homotopies given above as an option in HOM4PS-2.0, just as in PHCpack.

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

System	CPU time	Mixed volume	# of curve jumpings	# of isolated solutions
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-15	7m03s	32,768	2	32,768
katsura-16	16m25s	65,536	-	65,536
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	-	5,994
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	-	14,401
reimer-9	7h52m40s	3,628,800	14	86,415

Table 1: The performance of HOM4PS-2.0.

Listed in Table 1 is the performance of HOM4PS-2.0 on solving all those bench mark systems in Table A. 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 curving jumping never appears for systems in each system category with sizes smaller than that were 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 [8], multiple retracings were required very often for the software package PHoM [8] 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 14,401. 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 § 5.3 as the cpu times in the table show.

The results displayed in Table 1 are the results that solved all the polynomial systems by the polyhedral homotopy method. As mentioned before, we may solve katsura-n and reimer-n systems by the classical linear homotopis because the mixed volume of each system agrees with its total degree. As a comparison, we list in Table 2 the results of solving those systems by both the classical linear homotopy option and the polyhedral 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 such 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. 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 ahead of time, 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 homotopy in the table indicate that nonlinear homotopies can be costly for large systems.

System	Total Degree	CPU time		# of isolated solutions
		Linear	**Polyhedral**	
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	14,401
reimer-9	3,628,800	7h52m40s	8h47m42s	86,415

Table 2: Comparison of the classical linear homotopy and the polyhedral homotopy in HOM4PS-2.0

6.2 HOM4PS-2.0 vs. HOM4PS

Table 3 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 of this amount of paths need 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: Comparison of HOM4PS and HOM4PS-2.0

6.3 HOM4PS-2.0 vs. PHCpack and PHoM

Listed in Table 4 is the comparison of the performance of HOM4PS-2.0 and PHCpack [25]. The implementation of solving polynomial systems by the classical linear homotopies is also available in PHCpack. Therefore the comparisons listed in Table 4 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 for noon-n systems) are the results by using the classical linear homotopy option in each package. Our powerful code MixedVol-2.0 [12] 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 [7] for mixed cell computations.

System	Total degree	CPU time		Speed-up ratio	# of isolated solutions
		PHCpack	**HOM4PS-2.0**		
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	5,994
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	-	14,401
reimer-9	3,628,800	-	7h52m40s	-	86,415

Table 4: Comparison of HOM4PS-2.0 and PHCpack

Table 5 compares the performance of HOM4PS-2.0 and PHoM [8]. The implementation of the classical linear homotopy for solving polynomial systems is not available in PHoM. Therefore all the results listed in Table 5 used the polyhedral homotopy method on all the systems.

System	Total degree	CPU time		Speed-up ratio	# of isolated solutions
		PHoM	**HOM4PS-2.0**		
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	-	5,994
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	-	86,415

Table 5: Comparison of HOM4PS-2.0 and PHoM

Table 6 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.

System	Maximum solvable 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 6: Maximum sizes of polynomial systems that can solved by PHCpack, PHoM, and HOM4PS-2.0 within 12 hours of cpu time

References

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

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

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

[7] 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.

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

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

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

[11] 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.

[12] T. L. Lee and T. Y. Li, "Mixed volume computation, A revisit", (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, (pp. 209-304), 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] 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.

[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 http://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.