# Hom4PS-3: A Parallel Numerical Solver for Systems of Polynomial Equations Based on Polyhedral Homotopy Continuation Methods

Tianran Chen[1], Tsung-Lin Lee[2], and Tien-Yien Li[1]

[1] Michigan State University, USA
chentia1@msu.edu, li@math.msu.edu
http://www.math.msu.edu/~chentia1/
http://www.math.msu.edu/~li/

[2] National Sun Yat-sen University, Taiwan ROC
leetsung@math.nsysu.edu.tw
http://www.math.nsysu.edu.tw/~leetsung/

**Abstract.** Homotopy continuation methods have been proved to be an efficient and reliable class of numerical methods for solving systems of polynomial equations which occur frequently in various fields of mathematics, science, and engineering. Based on the successful software package Hom4PS-2.0 for solving such polynomial systems, Hom4PS-3 has a new fully modular design which allows it to be easily extended. It implements many different numerical homotopy methods including the Polyhedral Homotopy continuation method. Furthermore, it is capable of carrying out computation in parallel on a wide range of hardware architectures including multi-core systems, computer clusters, distributed environments, and GPUs with great efficiency and scalability. Designed to be user-friendly, it includes interfaces to a variety of existing mathematical software and programming languages such as Python, Ruby, Octave, Sage and Matlab.

**Keywords:** polynomial systems, homotopy continuation, polyhedral homotopy, binomial system.

## 1 Introduction

The problem of solving systems of polynomial equations, or polynomial systems, has been, and will continue to be, one of the most important subjects in both pure and applied mathematics. The need to solve polynomial systems arises naturally and frequently in various fields of science and engineering as documented in [1,13,18]. The *homotopy continuation method* has been established, in recent years, as one of the most reliable and efficient class of numerical methods for finding the full set of isolated solutions to a general polynomial system. There are many mature software implementing this method, including Bertini[3], HOMPACK[20], NAG4M2[12], and etc. See [2,13,16,18] for a survey. One important branch among them is the *polyhedral homotopy* method initiated in [9]. The method has been successfully implemented in software packages PHCpack [19] developed by J. Verschelde at University of Illinois at Chicago Circle, PHoM [8] developed by a group led by M. Kojima at Tokyo Institute of Technology in Japan, and Hom4PS-2.0 [10] developed by a group led by the authors. The efficiency and reliability in real world applications of Hom4PS-2.0 is documented in [10,11,13]. Based Hom4PS-2.0, a new numerical solver for polynomial systems Hom4PS-3[5] is created around the same core mathematical algorithms. Written in the C++ programming language and taking advantage of the object-oriented programming paradigm, Hom4PS-3 has a fully modular structure following modern design principles that allows it to be easily extended by individual "modules". Designed to be user-friendly from the ground up, it includes interfaces to a variety of existing mathematical software and programming languages such as Sage, Python, Ruby, Octave, and Matlab.

In real world applications from science and engineering, there is no shortage in the demand of solving larger and larger polynomial systems. Homotopy continuation methods are particularly suited to handle these large polynomial systems due to its *pleasantly parallel* nature: each isolated solution is computed independently of the others. Hom4PS-3 is designed to take advantage of a variety of parallel hardware architectures including *multi-core* systems, *NUMA* systems, *computer clusters*, *distributed environments*, and *GPUs*. Using parallel computation techniques tailored for each architecture (a symmetric model using Intel TBB and OpenMP on multi-core architectures, a hierarchical model for NUMA architectures, a master-worker model using MPI on clusters, an asynchronous message passing model for distributed environments, and a hybrid "*single-thread-multiple-data*" model for GPUs), excellent efficiency and scalability have been achieved on these systems.

## 2 Functionality

Given an input polynomial system, which can be represented in a number of different formats, Hom4PS-3 solves the polynomial system and outputs a list of complex solutions. This list includes *all* isolated nonsingular solutions of the given system in $\mathbb{C}^n$ as well as isolated singular solutions together with their multiplicity information. Optionally, Hom4PS-3 can also produce "solutions at infinity" by carrying out computation in the complex projective space or weighted projective spaces. For polynomial systems having solution components of positive dimensions, an included module posdim can be used to compute sample or "*witness*" points on solution components of any given dimension. Furthermore, the number of components and their degrees can be computed via the "witness" points.

On a Unix/Linux or similar operating system, one can solve a polynomial system with Hom4PS-3 simply by invoking the command

```
hom4ps-easy FILE
```

on a terminal, where `FILE` is the path of the file that contains the representation of the input polynomial system. This command runs Hom4PS-3 in its "easy mode" in which a predetermined set of parameters for controlling the behavior of the program that is likely appropriate for most situations is used.

The behavior of Hom4PS-3 can be controlled via a long list of switches and parameters given either on the command line or as a configuration file. They control the usage of certain modules, the precision to be used for floating point arithmetic, the strategy for adjusting "step sizes" in the procedure of tracking homotopy paths (Section 3), and many other aspects of the program. A complete list can be found in its reference manual. Both the downloadable packages and complete documentation can be found on the website `http://www.hom4ps3.org`.

## 2.1 Parallel Computation Capabilities

**Multi-core Systems.** A multi-core processor contains multiple processing units, called "cores", each capable of executing program instructions and carrying out computation independently. On a multi-core system, Hom4PS-3 automatically spreads work load across all available processor cores on the system via a multi-thread model. The implementation supports both Intel TBB and OpenMP, two of the most popular programming frameworks for multi-core systems.

**Computer Cluster.** A computer cluster is a group of computers, connected via high speed network, that work together on a single task and can be viewed as a single computer system. Using MPI, the *de facto* standard for communication on clusters, Hom4PS-3 can distribute work load among nodes in the cluster.

**GPU Computing.** GPU computing is the use of graphics processing units, or GPUs, which are originally designed for rendering graphics, to perform general purpose computation in a highly parallel fashion. On platforms where one or more GPU devices are available, Hom4PS-3 can take advantage of these highly parallel hardware on specific tasks involving intensive floating point matrix and vector manipulations such as polynomial and derivative evaluation and "mixed volume" computation (Section 3). The current implementation is built on top CUDA, a popular proprietary GPU programming framework developed by NVidia. Experimental supports for OpenCL, the dominant open standard developed by multiple vendors, are under active development.

## 2.2 Interfaces with Existing Mathematical Software and Programming Languages

**Sage Interface.** Sage is a free open-source mathematical software with features covering many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus. The Sage interface is one of the easiest way to use Hom4PS-3. For example, one can use Hom4PS-3 to solve a extremely simple polynomial system in Sage by using the following commands:

```
import hom4pspy
R.<x,y> = CC['x,y']
f = x^2 - 3*x + 2
g = y^2 - 4*y + 3
hom4pspy.solve_real ( [f,g] )
```

In this example, the first line imports the Hom4PS-3 interface. The next three lines creates a polynomial ring in two variables over the complex (floating point) field and two polynomials using the Sage syntax. The last line solves the polynomial system for the real solutions via Hom4PS-3 and returns a list of `dict` each describing a solution. Of course it is typically used to handle much more complicated and larger systems than this simple example, and with the power of Sage one can perform complicated algebraic construction to build the input system for Hom4PS-3, bridging the world of symbolic and numerical computation.

**Python Interface.** Python is a popular programming language (and the solid foundation on top of which Sage was built). The `hom4pspy` module used in the Sage can also be used separately as a Python interface. The commands

```
import hom4pspy
hom4pspy.solve_real ( ["x^2 - 3*x + 2", "y^2 - 4*y + 3"] )
```

solves the same simple polynomial system, now represented as strings.

**Octave Interface.** (GNU) Octave is a software and programming language designed for numerical computations that is mostly compatible with Matlab. In Octave, with the Hom4PS-3 interface, the commands

```
hom4psoct
hom4psoct.solve_real ( "x^2 - 3*x + 2, y^2 - 4*y + 3" )
```

solves the same simple polynomial system, represented as a single string (due to the lack of sophisticated symbolic manipulation capabilities in Octave).

## 3 Underlying Theory

In the 90's, a considerable research effort in Europe had been directed to the problem of solving polynomial systems in two consecutive major projects, PoSSo (**Po**lynomial **S**ystem **So**lving) and FRISCO (**FR**amework for **I**ntegrated **S**ymbolic and numerical **CO**mputation), supported by the European Commission. Those research projects focused on the development of the well-established Gröbner basis methods within the framework of computer algebra. Their reliance on symbolic manipulation makes those methods seem somewhat limited to relatively small problems. In 1977, Garcia and Zangwill [7] and Drexler [6] independently discovered that the *homotopy continuation method* could be used to find the full set of isolated solutions to a polynomial system numerically. In the last several decades, the method has been quite well developed and proved to be reliable and efficient. Note that *continuation methods* are the method of choice to deal with general nonlinear systems of equations numerically and globally as illustrated by the extensive bibliography listed in [1] where general ideas of the method were discussed.

One of the most important branches of the homotopy continuation method for solving general polynomial systems is the *polyhedral* homotopy method initiated by B. Huber and B. Sturmfels [9]. For an $n \times n$ square polynomial systems

$$P(x_1, \ldots, x_n) = P(\mathbf{x}) = \begin{cases} p_1(\mathbf{x}) = \sum_{\mathbf{a} \in S_1} c_{1,\mathbf{a}} \mathbf{x}^{\mathbf{a}} \\ \quad \vdots \\ p_n(\mathbf{x}) = \sum_{\mathbf{a} \in S_n} c_{n,\mathbf{a}} \mathbf{x}^{\mathbf{a}} \end{cases} \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_n)$, $\mathbf{a} = (a_1, \ldots, a_n)^\top \in \mathbb{N}_0^n$, and $\mathbf{x}^{\mathbf{a}} = x_1^{a_1} \cdots x_n^{a_n}$. Here $S_j$, a finite subset of $\mathbb{N}_0^n$, is called the *support* of $p_j(\mathbf{x})$. For fixed supports $S_1, \ldots, S_n$, it is a basic fact in algebraic geometry that for generic choices of the complex coefficients $c_{j,\mathbf{a}} \in \mathbb{C}^*$ the number of isolated solutions of the system $P(\mathbf{x}) = 0$ in $(\mathbb{C}^*)^n$ is a fixed number. The word "generic" here can be understood as "randomly chosen". Its precise meaning can be found in [4], [9] and [13]. This fixed number also serves as an upper bound on the number of isolated solutions $P(\mathbf{x}) = 0$ can have in $(\mathbb{C}^*)^n$ among all choices of coefficients. In [4], this upper bound, now commonly known as the *BKK bound*, is formulated in terms of *mixed volume*: For convex polytopes $\mathcal{Q}_1, \ldots, \mathcal{Q}_k \subset \mathbb{R}^k$, let $\lambda_1 \mathcal{Q}_1, \ldots, \lambda_k \mathcal{Q}_k$ represent their scaled version, by factors of positive $\lambda_1, \ldots, \lambda_k$ respectively. Then the *Minkowski sum* $\lambda_1 \mathcal{Q}_1 + \cdots + \lambda_k \mathcal{Q}_k$ is also a convex polytope. It can be shown that the volume $\mathrm{Vol}_k(\lambda_1 \mathcal{Q}_1 + \cdots + \lambda_k \mathcal{Q}_k)$ in $\mathbb{R}^k$ is a homogeneous polynomial in $\lambda_1, \ldots, \lambda_k$. The *mixed volume*, denoted by $\mathrm{MVol}(\mathcal{Q}_1, \ldots, \mathcal{Q}_k)$, is defined to be the coefficient of $\lambda_1 \times \lambda_2 \times \cdots \times \lambda_k$ in this polynomial. The theory of BKK bound [4] states that the number of isolated solutions of the system $P(\mathbf{x}) = 0$ in $(\mathbb{C}^*)^n$ for generic choices of the coefficients is the mixed volume of the convex hull of the supports of $p_1, \ldots, p_n$, i.e.

$$\mathrm{MVol}(\mathrm{conv} S_1, \ldots, \mathrm{conv} S_n).$$

We shall restrict our focus on solving a polynomial system $P(\mathbf{x}) = 0$ in (1) with "generic" (nonzero) complex coefficients $c_{j,\mathbf{a}} \in \mathbb{C}^*$. When the system with generic coefficients is solved, one can always use it to solve the system with specifically given coefficients with the same supports by the *Cheater's* homotopy [14] (or [17]).

To solve $P(\mathbf{x}) = 0$ in (1), consider, with a new variable $t$, the homotopy

$$H(x_1, \ldots, x_n, t) = H(\mathbf{x}, t) = \begin{cases} h_1(\mathbf{x}, t) = \sum_{\mathbf{a} \in S_1} c_{1,\mathbf{a}} \mathbf{x}^{\mathbf{a}} t^{\omega_1(\mathbf{a})} \\ \quad \vdots \\ h_n(\mathbf{x}, t) = \sum_{\mathbf{a} \in S_n} c_{n,\mathbf{a}} \mathbf{x}^{\mathbf{a}} t^{\omega_n(\mathbf{a})} \end{cases}. \tag{2}$$

with "lifting" functions $\omega_1, \ldots, \omega_n$, where each $\omega_k : S_k \to \mathbb{Q}$ has randomly chosen images. Note that when $t = 1$, $H(\mathbf{x}, 1) = P(\mathbf{x})$. For $\mathbf{a} \in S_k$, write

$\hat{\mathbf{a}} = (\mathbf{a}, \omega_k(\mathbf{a}))$. In [9], it was shown that if the system $P(\mathbf{x}) = 0$ has isolated solutions in $(\mathbb{C}^*)^n$, then there exists $\hat{\alpha} = (\alpha, 1) \in \mathbb{R}^{n+1}$ with $\alpha = (\alpha_1, \ldots, \alpha_n)$ and a corresponding collection of pairs $\{\mathbf{a}_1, \mathbf{a}_1'\} \subset S_1, \ldots, \{\mathbf{a}_n, \mathbf{a}_n'\} \subset S_n$ such that for each $k = 1, \ldots, n$

$$\langle \hat{\mathbf{a}}_k, \hat{\alpha} \rangle = \langle \hat{\mathbf{a}}_k', \hat{\alpha} \rangle < \langle \hat{\mathbf{a}}, \hat{\alpha} \rangle \quad \text{for all } \mathbf{a} \in S_k \setminus \{\mathbf{a}_k, \mathbf{a}_k'\} \tag{3}$$

and

$$\kappa_\alpha \ := \ \left| \det \left[ \mathbf{a}_1 - \mathbf{a}_1' \ \ldots \ \mathbf{a}_n - \mathbf{a}_n' \right] \right| \ > \ 0.$$

Here $\langle \ , \ \rangle$ stands for the standard inner product in Euclidean space. Let $\mathcal{T}$ be the collection of all such $\alpha$'s, then

$$\sum_{\alpha \in \mathcal{T}} \kappa_\alpha$$

is independent of the choice of the lifting functions $\omega_1, \ldots, \omega_k$. In fact, this number agrees with the number of isolated solutions of the system $P(\mathbf{x}) = 0$ in $(\mathbb{C}^*)^n$, counting multiplicities, known as the BKK bound mentioned before.

Now, for a fixed $\alpha$ in $\mathcal{T}$ along with its corresponding set of pairs $\{\mathbf{a}_1, \mathbf{a}_1'\} \subset S_1, \ldots, \{\mathbf{a}_n, \mathbf{a}_n'\} \subset S_n$, let $\beta_k = \langle \hat{\mathbf{a}}_k, \hat{\alpha} \rangle = \langle \hat{\mathbf{a}}_k', \hat{\alpha} \rangle = \langle \mathbf{a}_k', \alpha \rangle + \omega_k(\mathbf{a}_k')$ for $k = 1, \ldots, n$. Then by (3), for each $k = 1, \ldots, n$,

$$\beta_k < \langle \hat{\mathbf{a}}, \alpha \rangle \quad \text{for all } \mathbf{a} \in S_k \setminus \{\mathbf{a}_k, \mathbf{a}_k'\}. \tag{4}$$

By the change of variables $\mathbf{x} = t^\alpha \bullet \mathbf{y}$, i.e., for $\mathbf{y} = (y_1, \ldots, y_n)$

$$\begin{cases} x_1 = t^{\alpha_1} y_1 \\ \quad \vdots \\ x_n = t^{\alpha_n} y_n, \end{cases} \tag{5}$$

we have, for $\mathbf{a} = (a_1, \ldots, a_n) \in S_k$ and $\hat{\mathbf{a}} = (\mathbf{a}, \omega_k(\mathbf{a}))$

$$\begin{aligned} \mathbf{x}^{\mathbf{a}} t^{\omega_k(\mathbf{a})} &= x_1^{a_1} \ \ldots \ x_n^{a_n} \ t^{\omega_k(\mathbf{a})} \\ &= (t^{\alpha_1} y_1)^{a_1} \ \ldots \ (t^{\alpha_n} y_n)^{a_n} \ t^{\omega_k(\mathbf{a})} \\ &= y_1^{a_1} \ \ldots \ y_n^{a_n} \ t^{a_1 \alpha_1 + \cdots + a_n \alpha_n + \omega_k(\mathbf{a})} \\ &= \mathbf{y}^{\mathbf{a}} \, t^{\langle (\mathbf{a}, \omega_k(\mathbf{a})), (\alpha, 1) \rangle} \\ &= \mathbf{y}^{\mathbf{a}} \, t^{\langle \hat{\mathbf{a}}, \hat{\alpha} \rangle} \end{aligned}$$

with $\hat{\alpha} = (\alpha, 1)$. Substituting the result into $H(\mathbf{x}, t)$ in (2), it follows that

$$\bar{H}^\alpha(\mathbf{y}, t) := H(t^\alpha \bullet \mathbf{y}, t) = \begin{cases} \bar{h}_1^\alpha(\mathbf{y}, t) := h_1(t^\alpha \bullet \mathbf{y}, t) = \sum_{\mathbf{a} \in S_1} c_{1,\mathbf{a}} \, \mathbf{y}^{\mathbf{a}} \, t^{\langle \hat{\mathbf{a}}, \hat{\alpha} \rangle} \\ \vdots \\ \bar{h}_n^\alpha(\mathbf{y}, t) := h_n(t^\alpha \bullet \mathbf{y}, t) = \sum_{\mathbf{a} \in S_n} c_{n,\mathbf{a}} \, \mathbf{y}^{\mathbf{a}} \, t^{\langle \hat{\mathbf{a}}, \hat{\alpha} \rangle} \end{cases}.$$

Though the above expression may contain positive or negative powers of $t$, the minimum exponents of $t$ in each $\bar{h}_k^\alpha$ is actually given by $\beta_k$. Therefore, if

$$H^\alpha(\mathbf{y},t) := \begin{cases} t^{-\beta_1}\bar{h}_1(\mathbf{y},t) &= \sum_{\mathbf{a}\in S_1} c_{1,\mathbf{a}}\,\mathbf{y}^{\mathbf{a}}\,t^{\langle \hat{\mathbf{a}},\hat{\alpha}\rangle-\beta_1} \\ & \vdots \\ t^{-\beta_n}\bar{h}_n(\mathbf{y},t) &= \sum_{\mathbf{a}\in S_n} c_{n,\mathbf{a}}\,\mathbf{y}^{\mathbf{a}}\,t^{\langle \hat{\mathbf{a}},\hat{\alpha}\rangle-\beta_n} \end{cases}, \tag{6}$$

then, by (4), each component of $H^\alpha$ has exactly two terms having no powers of $t$ while all other terms have positive powers of $t$. Hence, when $t = 0$, $H^\alpha(\mathbf{y},0) = 0$ is the "binomial system of equations"

$$\begin{cases} c_{1,\mathbf{a}_1}\mathbf{y}^{\mathbf{a}_1} + c_{1,\mathbf{a}'_1}\mathbf{y}^{\mathbf{a}'_1} = 0 \\ \vdots \\ c_{n,\mathbf{a}_n}\mathbf{y}^{\mathbf{a}_n} + c_{n,\mathbf{a}'_n}\mathbf{y}^{\mathbf{a}'_n} = 0 \end{cases} \tag{7}$$

with $\kappa_\alpha$ nonsingular isolated solutions in $(\mathbb{C}^*)^n$. It is known that such binomial systems can be solved accurately and efficiently via numerical methods [13]. After (7) is solved, these nonsingular solutions obtained can be used as the starting points for following the homotopy paths $\mathbf{y}(t)$ of $H^\alpha(\mathbf{y},t) = 0$, for which $H^\alpha(\mathbf{y}(t),t) = 0$ from $t = 0$ to $t = 1$. Note that the change of variables $\mathbf{x} = t^\alpha \bullet \mathbf{y}$ in (5) yields $\mathbf{x} \equiv \mathbf{y}$ at $t = 1$. Therefore, each end point $\mathbf{y}(1)$ at $t = 1$ of the homotopy path $\mathbf{y}(t)$ of $H^\alpha(\mathbf{y},t) = 0$ is also an end point $\mathbf{x}(1)$ of the homotopy path $\mathbf{x}(t)$ of the homotopy $H(\mathbf{x},t) = 0$ given in (2) which, in turn, provides a solution of the target system $P(\mathbf{x}) = 0$ in (1). Altogether it yields $\kappa_\alpha$ of the isolated solutions of $P(\mathbf{x}) = 0$ in $(\mathbb{C}^*)^n$ along this route. In [9], it was shown that as one follows the homotopy paths defined by $H^\alpha(\mathbf{y},t) = 0$ for all individual $\alpha \in \mathcal{T}$, one obtains all (isolated) solutions of $P(\mathbf{x}) = 0$ in $(\mathbb{C}^*)^n$, justifying, in fact, the BKK bound agrees with $\sum_{\alpha\in\mathcal{T}} \kappa_\alpha$.

Even though the above procedure only addresses the solution set in $(\mathbb{C}^*)^n$ of the target system $P(\mathbf{x}) = 0$ in (1), this method has been extended in [15] so that all isolated zeros of the target system $P(\mathbf{x})$ in $\mathbb{C}^n$ can be obtained. Since its inception, this general method has achieved a great success. It is widely considered to be one of the most efficient, robust and reliable numerical methods for solving systems of polynomial equations. Hom4PS-3 implements this method as its primary tool.

## References

1. Allgower, E., Georg, K.: Introduction to numerical continuation methods, vol. 45. Society for Industrial and Applied Mathematics (2003)
2. Attardi, G., Traverso, C.: The PoSSo library for polynomial system solving. In: Proc. of AIHENP 1995 (1995)
3. Bates, D.J., Hauenstein, J.D., Sommese, A.J., Wampler, C.W.: Numerically Solving Polynomial Systems with Bertini. Society for Industrial and Applied Mathematics (2013)

4. Bernshtein, D.N.: The number of roots of a system of equations. Functional Analysis and its Applications 9(3), 183–185 (1975)
5. Chen, T.R., Lee, T.L., Li, T.Y.: Hom4PS-3: an numerical solver for polynomial systems using homotopy continuation methods, `http://www.hom4ps3.org`
6. Drexler, F.-J.: Eine methode zur berechnung sämtlicher lösungen von polynomgleichungssystemen. Numerische Mathematik 29(1), 45–58 (1977)
7. Garcia, C.B., Zangwill, W.I.: Finding all solutions to polynomial systems and other systems of equations. Mathematical Programming 16(1), 159–176 (1979)
8. Gunji, T., Kim, S., Kojima, M., Takeda, A., Fujisawa, K., Mizutani, T.: PHoM–a polyhedral homotopy continuation method for polynomial systems. Computing 73(1), 57–77 (2004)
9. Huber, B., Sturmfels, B.: A polyhedral method for solving sparse polynomial systems. Mathematics of Computation 64(212), 1541–1555 (1995)
10. Lee, T.L., Li, T.Y., Tsai, C.H.: HOM4PS-2.0: a software package for solving polynomial systems by the polyhedral homotopy continuation method. Computing 83(2), 109–133 (2008)
11. Lee, T.L., Li, T.Y., Tsai, C.H.: HOM4PS-2.0 para: Parallelization of HOM4PS-2.0 for solving polynomial systems. Parallel Computing 35(4), 226–238 (2009)
12. Leykin, A.: NAG4M2: Numerical algebraic geometry for Macaulay2, `http://people.math.gatech.edu/~aleykin3/NAG4M2/`
13. Li, T.Y.: Numerical solution of polynomial systems by homotopy continuation methods. In: Ciarlet, P.G. (ed.) Handbook of Numerical Analysis, vol. 11, pp. 209–304. North-Holland (2003)
14. Li, T.Y., Sauer, T., Yorke, J.: The cheater's homotopy: an efficient procedure for solving systems of polynomial equations. SIAM Journal on Numerical Analysis, 1241–1251 (1989)
15. Li, T.Y., Wang, X.: The BKK root count in $\mathbb{C}^n$. Mathematics of Computation of the American Mathematical Society 65(216), 1477–1484 (1996)
16. Morgan, A.P.: Solving polynomial systems using continuation for engineering and scientific problems. Classics in Applied Mathematics, vol. 57. Society for Industrial and Applied Mathematics (2009)
17. Morgan, A.P., Sommese, A.J.: Coefficient-parameter polynomial continuation. Applied Mathematics and Computation 29(2), 123–160 (1989)
18. Sommese, A.J., Wampler, C.W.: The Numerical solution of systems of polynomials arising in engineering and science. World Scientific Pub. Co. Inc. (2005)
19. Verschelde, J.: Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. ACM Transactions on Mathematical Software (TOMS) 25(2), 251–276 (1999)
20. Watson, L.T., Billups, S.C., Morgan, A.P.: Algorithm 652: Hompack: A suite of codes for globally convergent homotopy algorithms. ACM Transactions on Mathematical Software (TOMS) 13(3), 281–310 (1987)