PHCpack: a general-purpose solver for polynomial systems by homotopy continuation

Jan Verschelde
Mathematical Sciences Research Institute

Polynomial systems occur in a wide variety of application domains. Homotopy continuation methods are reliable and powerful methods to compute numerically approximations to all isolated complex solutions. During the last decade considerable progress has been accomplished on exploiting structure in a polynomial system, in particular its sparsity. In this paper the structure and design of the software package PHC is described. The main program operates in several modes, is menu-driven and file-oriented. This package features a great variety of root-counting methods among its tools. The outline of one black-box solver is sketched and a report is given on its performance on a large database of test problems. The software has been developed on four different machine architectures. Its portability is ensured by the gnu-ada compiler.

Categories and Subject Descriptors: D.3.2 [**Programming Languages**]: Language Classification—*Ada*; G.1.5 [**Numerical Analysis**]: Roots of Nonlinear Equations—*Systems of equations, Polynomials, methods for*; G.2.1 [**Discrete Mathematics**]: Combinatorics—*Counting problems*; G.4 [**Mathematics of Computing**]: Mathematical Software

General Terms: Algorithms, Theory

Additional Key Words and Phrases: homotopy continuation, polynomial systems, start system, root count, Bézout number, mixed volume, Bernshteĭn's theorem, polyhedral homotopy, enumerative geometry, Schubert calculus.

1. INTRODUCTION

The presented software package PHC implements homotopy continuation methods to compute numerically approximations to all isolated solutions of a system of n

---

Address: MSRI, 1000 Centennial Drive, Berkeley, CA 94720-5070, USA.
E-mail addresses: jan@msri.org, jan@math.msu.edu and jan.verschelde@na-net.ornl.gov.
URLs: http://www.msri.org/people/members/jan and http://www.mth.msu.edu/~jan.
The software originated from the doctoral research of the author and was supported by the Fund for Scientific Research Flanders (Belgium) through Grant G.0261.96 to the NINES Research Group, Department of Computer Science, K.U.Leuven. Several upgrades were made during the author's post-doctoral fellowships at the Department of Computer Science of the Katholieke Universiteit Leuven, at the Department of Mathematics of Michigan State University, and at the Mathematical Sciences Research Institute. Supported in part by NSF under Grant DMS - 9804846 at Department of Mathematics, Michigan State University, East Lansing, MI 48824-1027, USA.

polynomial equations in n unknowns.

The name *Polynomial Homotopy Continuation* unites the three key concepts of the method. Since we solve *polynomial* systems we exploit the algebraic structure to count the roots and to construct a start system. By *continuation* methods the known solutions of the start system are extended to the desired solutions of the target system. This deformation is defined by the *homotopy*, that is a family of systems connecting start and target system.

The following intuitive reasoning might shed a light on the hardness of the problem. Not surprisingly, evaluating a multivariate polynomial can be done in *polynomial time*, i.e., in time proportional to a polynomial function in the dimension, degrees and number of terms. Computing one solution is *NP-hard*, because we may apply the following non-deterministic algorithm: guess a root by some oracle and verify whether it satisfies the equations. This verification runs in polynomial time. Computing *all* solutions is harder, because there exists no polynomial-time algorithm to verify whether a guessed number of solutions gives the right number of solutions. This counting problem is said to be *#P-hard*. Hence our problem is intractable [Garey and Johnson 1979] for growing dimension and increasing degrees. Consequently, the computations are restricted to a fixed dimension n and the complexity is measured in terms of the output size, that is the number of solutions. See [Blum, Cucker, Shub, and Smale 1997] for the complexity of Bézout's theorem.

The central concept in polynomial homotopy continuation is the *root count*[1], because it determines the number of solution paths that need to be traced. Recent research has striven to develop sharp root counts that lead to homotopies with an optimal number of paths. The root count is also a vital instrument in validating numerical results. This term encompasses Bézout numbers, mixed volumes, and combinatorial counts from the Schubert calculus in enumerative geometry.

The history of homotopy continuation for polynomial systems can be roughly divided into two eras, each spanning about one decade. The first decade was focussed on applying Bézout's theorem for counting the solutions. Milestone publications are the introductory paper [Li 1987], the book [Morgan 1987], and the survey [Watson 1986]. Publicly available software packages are CONSOL [Morgan 1987] and HOMPACK [Watson, Billups, and Morgan 1987], recently upgraded to Fortran 90 [Watson, Sosonkina, Melville, Morgan, and Walker 1997]. During the last ten years, root-counting methods have been developed to exploit the structure of a polynomial system. The novel methods are of a symbolic-numeric nature [Emiris 1998]. Progress in homotopy continuation for polynomial systems [Li 1997] benefited from the interaction between combinatorics, algebraic geometry and applied mathematics [Sturmfels 1998]. The polyhedral methods have brought homotopy continuation into the literature on computational algebraic geometry [Cox, Little, and O'Shea 1998]. New publicly available software packages are Pelican [Huber 1995] and PHC. Recently, optimal homotopies were presented for computing linear subspace intersections in enumerative geometry, see [Sottile 1997], [Huber, Sottile, and Sturmfels 1998] and [Verschelde 1998].

[1] The term *root count* was coined by Canny and Rojas. They presented in [Canny and Rojas 1991] the mixed volume as of important practical significance for solving polynomial systems.

The aim of this paper is to give an overview on how the algorithms in PHC are used in practice to solve polynomial systems. In the next section related software packages are mentioned. PHC offers a great variety of root-counting methods, as explained in section three. The fourth section contains some basics about polynomial continuation. Sections five, six and seven describe the general flow, the operation modes of the main program, and the internal structure of the package. The software is portable, computer-compiler experiences are given in section eight. In sections nine and ten the outline of the black-box solver is presented, along with its practical performance on a large collection of applications. Before the conclusions, information on how to obtain and install the software is listed.

2. RELATED SOFTWARE

To indicate related software dedicated to solving polynomial systems by homotopy continuation, four different packages that are publicly available are briefly mentioned. We also refer to a program for computing mixed volumes. In closing this section we indicate some large-scale related software projects.

HOMPACK [Morgan, Sommese, and Watson 1989] [Watson, Billups, and Morgan 1987] and CONSOL [Morgan 1987] are written in Fortran 77. HOMPACK is a general package for homotopy continuation with a polynomial driver. It has been parallelized [Allison, Chakraborty, and Watson 1989], [Harimoto and Watson 1989], and extended with an end game [Sosonkina, Watson, and Stewart 1996]. A Fortran 90 version appeared recently [Watson, Sosonkina, Melville, Morgan, and Walker 1997]. The package POLSYS_PLP [Wise, Sommese, and Watson 1998] for constructing partitioned linear-product start systems is intended to be used in conjunction with HOMPACK90. The code for CONSOL is contained in [Morgan 1987, Appendix 6]. Morgan, Sommese and Wampler [1991, 1992a, 1992b] developed techniques to handle end-point singularities.

Malajovitch created pss to apply homotopy continuation with verification by α-theory. The program contains facilities for parallel continuation. Originally written in C, the newest version [Malajovich 1996] is programmed in C++. Pelican [Huber 1995; Huber 1996] implements in C the polyhedral methods of [Huber and Sturmfels 1995]. Gao has created Fortran software for polyhedral continuation, with facilities to compute the affine roots [Gao, Li, and Wang 1997].

The computation of mixed volumes is a crucial step in the resolution of sparse polynomial systems. The C program mvlp [Emiris 1994; Emiris and Canny 1995] computes mixed volumes, see [Giordano 1996] for a distributed version. For a general resultant-based polynomial-system solver, we refer to [Wallack, Emiris, and Manocha 1998].

In recent years, the attention to software for solving polynomial systems increased largely. FRISCO [The FRISCO Consortium 1996] is a three-year project funded by the European Commission under the Esprit Reactive LTR Scheme (project no. 21.024). A demo of the software produced by the predecessor project PoSSo is available at [The Pisa Team of PoSSo 1993].

3. ROOT COUNTS AND START SYSTEMS

The computation of a root count is identified with the resolution of a generic system. In this sense, we call root-counting a symbolic computation mirroring this resolution. The basic root-counting principles for dense, sparse, and determinantal systems are exemplified next.

The use of multi-homogenization was proposed in [Morgan and Sommese 1987a; Morgan and Sommese 1987b]. Li, Sauer and Yorke [1987a, 1987b] introduced random product homotopies, see also [Li and Wang 1991].

EXAMPLE 3.1. Consider a 2-dimensional generalized eigenvalue problem, represented by a polynomial in λ with as coefficients 2-by-2 matrices. A linear equation is added to scale the eigenvectors.

$$F(\mathbf{x}) = \left\{ \begin{array}{rcl} \left(\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \lambda^2 + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \lambda + \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \right) \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} & = & \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 & = & 0 \end{array} \right. \quad (1)$$

The total degree D equals $3 \times 3 \times 1$ and overshoots the number of roots. For this problem we see that the components of the eigenvector occur linearly, whereas the degree of the eigenvalue equals two. By separating the unknowns in a partition Z, a 2-homogeneous Bézout number is obtained as follows

$$Z = \{\{\lambda\}, \{x_1, x_2\}\} \qquad \begin{bmatrix} \overbrace{2}^{\{\lambda\}} & \overbrace{1}^{\{x_1,x_2\}} \\ 2 & 1 \\ 0 & 1 \end{bmatrix} \qquad B_Z = 2 \times 1 + 2 \times 1 + 0 \times 1 = 4. \quad (2)$$

The matrix contains the degrees of the polynomials w.r.t. the sets in Z. The Bézout number B_Z is computed as the generalized permanent of this matrix. This computation models the resolution of the following linear-product system

$$F^{(0)}(\mathbf{x}) = \left\{ \begin{array}{rcl} \overbrace{(\alpha_{11}\lambda + \alpha_{12})(\alpha_{13}\lambda + \alpha_{14})}^{\{\lambda\}} \overbrace{(\alpha_{15}x_1 + \alpha_{16}x_2 + \alpha_{17})}^{\{x_1,x_1\}} & = & 0 \\ (\alpha_{21}\lambda + \alpha_{22})(\alpha_{23}\lambda + \alpha_{24})(\alpha_{25}x_1 + \alpha_{26}x_2 + \alpha_{27}) & = & 0 \\ \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 & = & 0. \end{array} \right. \quad (3)$$

Any random number generator will yield α-coefficients of the linear-product system $F^{(0)}(\mathbf{x}) = \mathbf{0}$ so that it has exactly four regular solutions. This leads to a homotopy with an optimal number of solution paths. ■

A heuristic method developed for constructing a good partition Z of the set of unknowns is outlined in [Verschelde 1996]. Besides that, an exhaustive enumeration as in [Wampler 1992] of all partitions is available in PHC. In case the number of independent roots equals B_Z, interpolation can be used to construct a start system [Verschelde, Beckers, and Haegemans 1991].

The idea of [Verschelde and Haegemans 1993] is that not every polynomial should be modeled by the same partition. This leads to partitioned linear-product start systems. General linear-product start systems were constructed in [Verschelde and Cools 1993b] and applied to symmetric polynomial systems in [Verschelde and Cools 1994]. The key condition is that a linear-product start system must contain all monomials of the target system. Theoretically, these homotopy methods can be considered as a special case of the polyhedral homotopy methods. In practice, we sometimes prefer product start systems, for solving a random linear-product system can be performed much more efficiently than solving a random coefficient system. PHC supports the construction of both partitioned and general linear-product start systems.

Efficient algorithms to construct general linear-product start systems are elaborated by Li, T. Wang and X. Wang [1996]. Morgan, Sommese and Wampler [1995] treated general product decompositions that do not restrict to linear factors. Recent coding efforts on partitioned linear-product start systems are reported by Wise, Sommese and Watson [1998].

The start solutions in linear-product homotopies are obtained by solving linear systems. In polyhedral homotopy methods [Huber and Sturmfels 1995] [Verschelde, Verlinden and Cools 1994], the start solutions are solutions to binomial systems.

Example 3.2. To solve a system that has two terms in any of its equations, unimodular transformations are applied to transform the system into a triangular structure. For the example below, $\mathbf{x} = \mathbf{y}^U$ abbreviates the substitution $(x_1, x_2) \leftarrow (y_1 y_2^{-1}, y_1^{-1} y_2^2)$.

$$F(\mathbf{x}) = \left\{ \begin{array}{l} x_1^2 x_2^1 - 1 = 0 \\ x_1^4 x_2^3 - 1 = 0 \end{array} \right. \quad U = \left[\begin{array}{rr} 1 & -1 \\ -1 & 2 \end{array} \right] \quad F(\mathbf{x} = \mathbf{y}^U) = \left\{ \begin{array}{r} y_1 - 1 = 0 \\ y_1 y_2^2 - 1 = 0 \end{array} \right. \tag{4}$$

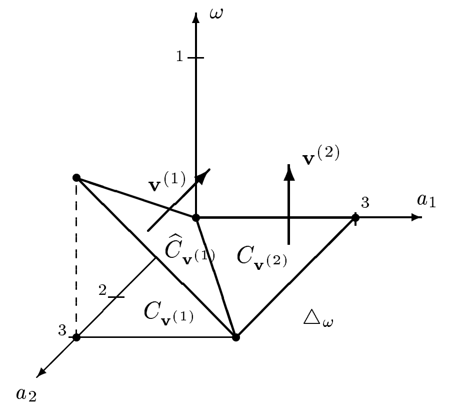
We see that $F(\mathbf{x}) = \mathbf{0}$ has two regular solutions. Geometrically, we have computed the area of a parallelogram spanned by the origin, the points $(2, 1)$ and $(4, 3)$, and their sum. This area equals the determinant of the matrix that has in its columns the spanning vectors of the parallelogram. Multiplying by U triangulates this matrix:

$$\left[\begin{array}{rr} 1 & -1 \\ -1 & 2 \end{array} \right] \left[\begin{array}{rr} 2 & 4 \\ 1 & 3 \end{array} \right] = \left[\begin{array}{rr} 1 & 1 \\ 0 & 2 \end{array} \right]. \tag{5}$$

As $\det(U) = 1$, U is called unimodular and preserves volume. Consequently, the transformation $\mathbf{x} = \mathbf{y}^U$ does not change the number of solutions. Note that the total degree and 2-homogeneous Bézout number equal respectively 21 and 10. ■

The above example is the sparsest case. The fewer monomials the fewer roots we expect [Khovanskiĭ 1991]. In general, we apply Bernshteĭn's theorem [Bernshteĭn 1975] and count the number of roots by the mixed volume of the Newton polytopes. By means of a regular subdivision, polyhedral homotopies are constructed that start at systems corresponding to the cells in the subdivision. Figure 1 illustrates the case where all Newton polytopes are the same, which is the case of Kushnirenko's theorem [1976]. The root count is obtained as the volume of the Newton polytope shared by all polynomials in the system.

The program features four different lifting methods: implicit, static, dynamic, and symmetric lifting. Implicit lifting refers to the algorithms used in the proof of David Bernshteĭn [Bernshteĭn 1975]. The method in [Huber and Sturmfels 1995] is called static, to make the distinction with dynamic lifting, an algorithm that has been developed in [Verschelde, Gatermann, and Cools 1996] to construct regular triangulations of polytopes incrementally with low lifting values. Symmetric



$$F(x_1, x_2) = \left\{ \begin{array}{l} x_1^3 x_2^3 + c_{11} x_1^3 + c_{12} x_2^3 + c_{13} = 0 \\ x_1^3 x_2^3 + c_{21} x_1^3 + c_{22} x_2^3 + c_{23} = 0 \end{array} \right.$$

$$\widehat{F}(x_1, x_2, t) = \left\{ \begin{array}{l} x_1^3 x_2^3 t^0 + c_{11} x_1^3 t^0 + c_{12} x_2^3 t^1 + c_{13} t^0 = 0 \\ x_1^3 x_2^3 t^0 + c_{21} x_1^3 t^0 + c_{22} x_2^3 t^1 + c_{23} t^0 = 0 \end{array} \right.$$

$$\mathbf{v}^{(1)} = [1, -1, 3]^T \qquad \mathbf{v}^{(2)} = [0, 0, 1]^T$$

Fig. 1. A regular triangulation of the Newton polytope of F with polyhedral homotopy $\widehat{F}$.

lifting was presented in [Verschelde and Gatermann 1995]. To construct regular subdivisions, both integer and floating-point lifting functions are available in PHC and elaborated with recursion. The Cayley trick [Gel'fand, Kapranov, and Zelevinsky 1994] defines in its polyhedral version [Sturmfels 1994] a polytope whose volume equals the mixed volume of the considered configuration of polytopes. In PHC, this trick is implemented by means of dynamic lifting. When many polynomials share the same exponents, this method is more efficient than static lifting.

With mixed volumes we restrict the counting to solutions that have all their components different from zero. Extensions to count and compute all isolated affine roots are described in [Rojas 1994; Rojas 1996], [Huber and Sturmfels 1997], [Li and Wang 1996], [Rojas and Wang 1996], [Gao, Li and Wang 1997], and in [Emiris and Verschelde 1997].

A new class of homotopy methods solves geometric problems whose intersection conditions are modeled by polynomial equations that arise from expanding determinants. Gröbner and SAGBI bases translate questions concerning ideals and subalgebras to monomial equations. The monomial orderings induced by weight vectors provide recipes to set up homotopies that are flat deformations, i.e.: preserve the structure of the solution set. These are the key ideas for the Gröbner and SAGBI homotopies introduced in [Huber, Sottile and Sturmfels 1998] to enumerate all p-planes that intersect mp given m-planes in general position in $\mathbb{C}^{m+p}$. See [Ravi, Rosenthal and Wang 1996], [Rosenthal and Willems 1998], for the relevance to the pole placement problem in control theory and for related computational experiments see [Rosenthal and Sottile 1998] [Sottile 1998], and [Verschelde 1998]. A third type of homotopies presented in [Sottile 1997] and [Huber, Sottile and Sturmfels 1998] has an intrinsic geometric meaning and is briefly described next.

Example 3.3. A classical problem in enumerative geometry [Kleiman and Laksov 1972] deals with finding the two lines in projective 3-space that meet four given lines in general position. For the configurations as in Figure 2 we have to solve the following system:

$$\det(X|L_i) = 0, \quad \Longleftrightarrow \quad \det \left(\begin{array}{cc|cc} x_{11} & 0 & c_{11}^{(i)} & c_{12}^{(i)} \\ x_{21} & 0 & c_{21}^{(i)} & c_{22}^{(i)} \\ 0 & x_{32} & c_{31}^{(i)} & c_{32}^{(i)} \\ 0 & x_{42} & c_{41}^{(i)} & c_{42}^{(i)} \end{array} \right) = 0, \quad i = 1,2,3,4 \qquad (6)$$

The special choice of coordinates so that L_1 is spanned by the first two and L_2 by the last two basis vectors admits the choice of local coordinates for the solution X. The best Bézout number for this system equals 6 and the mixed volume equals 4, whereas there are only two solutions.

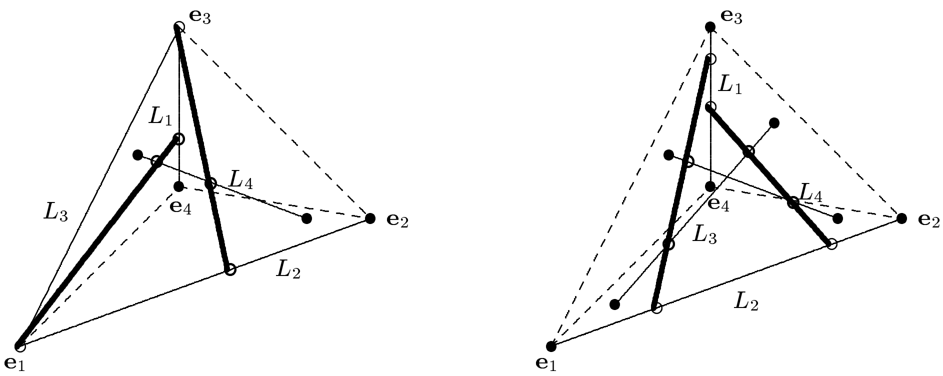


Fig. 2. In $I\!P^3$ two thick lines meet four given lines L_1, L_2, L_3, and L_4 in a point. At the left we see a special configuration and the general configuration is at the right.

The so-called Pieri-homotopy starts at the special configuration, displayed at the left of Figure 2 and moves the third input line in general position. To reach the two solutions of this problem, it suffices to follow the solution paths defined by this homotopy. ■

The start systems and root counts presented here are optimal for three different classes of polynomial systems. This classification is only a sample and by no means exhaustive. We expect that future research developments will extend this list of root counters and homotopies.

4. POLYNOMIAL CONTINUATION AND END GAMES

Solution paths of polynomial homotopies do not turn back as the continuation parameter t increases, due to the regularity of the paths, as discussed in [Li and Sauer 1987]. Therefore an increment-and fix predictor-corrector method is appropriate: after each increase of t, t remains fixed while correcting the solution $\mathbf{x}$ by Newton's method. Figure 3 sketches two possible predictor schemes in the path tracker.

The clustering of solution paths is avoided by tightening the tolerances of the corrector to enforce quadratic convergence of Newton's method in every step.

Only as $t \rightarrow 1$, we may have to deal with paths converging to singular solutions and with paths diverging to infinity. To this end, several *end games* were proposed

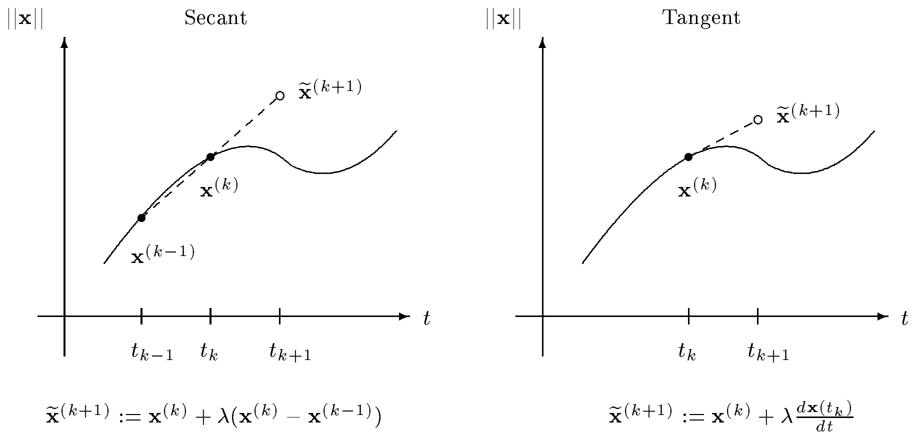


Fig. 3. The secant and tangent predictor with λ as step length.

by Morgan, Sommese and Wampler [1991,1992a,1992b] and by Sosonkina, Watson and Stewart in [1996]. Polyhedral end games [Huber and Verschelde 1998] provide a certificate of divergence that allows to separate diverging paths from the rest, without first having to compute the actual values of the diverging paths accurately. Next we summarize the idea of [Huber and Verschelde 1998].

A solution path is represented by the following power series expansion:

$$\left\{ \begin{array}{rcl} x_i(s) & = & a_i s^{\omega_i}(1 + O(s)) \\ t(s) & = & 1 - s^m \end{array} \right. \qquad t \approx 1, \quad s \approx 0. \tag{7}$$

The winding number m is lower than or equal to the multiplicity of the solution. We see that for a solution diverging to infinity or to a zero-component solution: $\omega_i \neq 0$. According to David Bernshteĭn's second theorem [1975], this solution corresponds to a solution of the face system defined by the direction ω. This face certifies the divergence.

To check whether a solution path really diverges is equivalent to the test on the value for ω_i. A first-order approximation of ω_i can be computed by

$$\frac{\log|x_i(s_1)| - \log|x_i(s_0)|}{\log(s_1) - \log(s_0)} = \omega_i + O(s_0), \tag{8}$$

with $0 < s_1 < s_0$. The above formula assumes the correct value of the winding number m. To compute m, solution paths are sampled geometrically with ratio h as $s_k = h^{k/m} s_0$. The errors on the estimates for ω_i are

$$e_i^{(k)} = (\log|x_i(s_k)| - \log|x_i(s_{k+1})|) - (\log|x_i(s_{k+1})| - \log|x_i(s_{k+2})|) \tag{9}$$

$$= c_1 h^{k/m} s_0 (1 + O(h^{k/m})). \tag{10}$$

An estimate for m is derived from two consecutive errors $e_i^{(k)}$. Extrapolation improves this estimate.

A parallel development to make resultants deal with situations when the mixed volume overshoots the number of roots is described in [Rojas 1997].

5. THE FOUR STAGES OF THE SOLVER

The root count provides important information about the amount of computational work that is required to solve the problem. It suffices to multiply the root count with the estimated time needed to follow one solution path.

In Figure 4, the four stages of the solver are displayed.

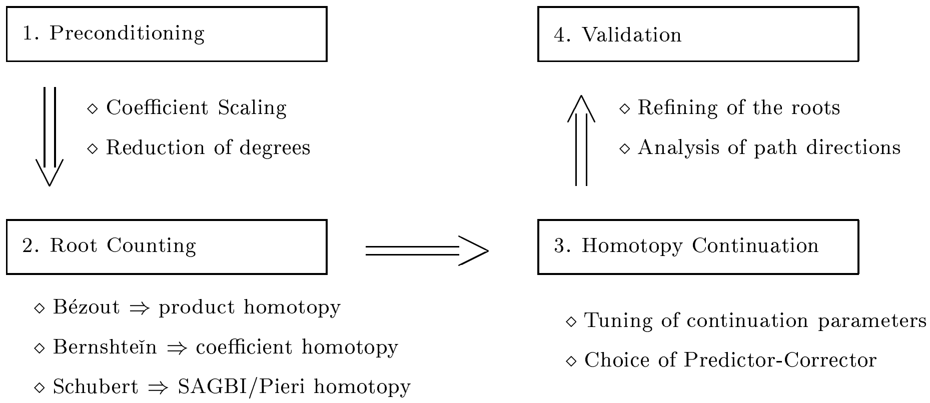


Fig. 4. The four stages in the solver.

The aim of preconditioning is to bring the system in a form more suitable to homotopy continuation. In the second stage, a root-counting method is applied to construct a start system. The tuning of continuation parameters and path following by means of predictor-corrector methods is performed in the third stage. The post-processing stage consists in the validation of the computed results. Basic validation includes for instance the computation of local condition numbers whereas more elaborate validation procedures eventually require continuation.

6. EXECUTION MODES AND TOOLS

Since we have to respect a strict processing order and we may expect computationally lengthy jobs, PHC is organized as a menu-driven and file-oriented program.

The simplest way to solve systems by PHC is to type

```
phc -b input output
```

when `input` is the name of the input file that contains the system. This mode is the so-called *black-box mode* and requires no other input than the polynomial system. Results can be found in the file `output`. One particular choice for a black-box solver is outlined in section nine.

The second mode is the *full mode* where PHC runs through all stages of the solver and asks the user to confirm the default settings while giving the opportunity to modify the settings interactively. This mode is invoked by default, just by typing `phc` after the prompt.

Some stages may be skipped, whereas more than one root-counting method can be invoked before the construction of a homotopy. Therefore, the *tool mode* has been created, see Figure 5. Another advantage of working with tools is that intermediate results, such as a mixed subdivision and a random coefficient start system, can be valuable stepping stones in the resolution of a large and difficult system.

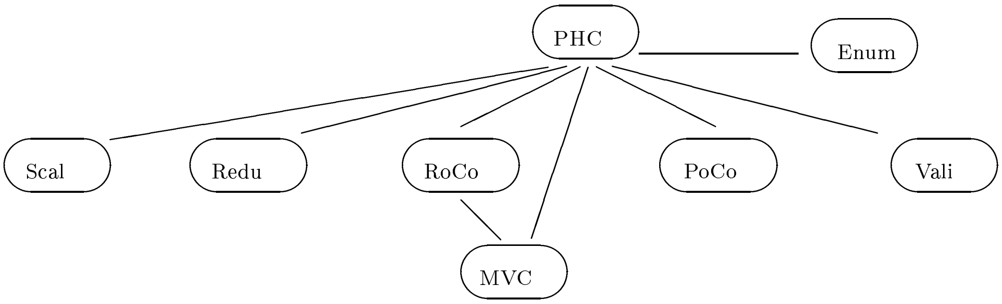


Fig. 5. Schematic overview of the tools offered by the package PHC.

Table 1 gives an overview of the tools and the options of PHC to invoke them.

stage	acronym	description of the tool	option
1	`scal`	coefficient scaling	`phc -s`
	`redu`	reduction of degrees	`phc -d`
2	`roco`	root counts and start systems	`phc -r`
	`mvc`	mixed-volume computation	`phc -m`
3	`poco`	polynomial continuation	`phc -p`
4	`vali`	validation of results	`phc -v`
x	`enum`	enumerative geometry	`phc -e`

Table 1. Overview of tools, acronyms and options of PHC.

The need for a separate tool for `mvc` comes from the amount of computational work that is not negligible for computing mixed volumes and performing polyhedral continuation. The reduction `redu` tool applies S-polynomials as described in [Verschelde and Cools 1992]. This technique generalizes the linear reduction on the coefficient matrix of the system [Morgan 1987].

7. THE INTERNAL DESIGN: THE LIBRARIES OF PHCPACK

There are four large components of the software system: the mathematical library, the homotopy continuation routines, the root-counting methods, and the interface packages.

The sources of PHCpack are organized in the following tree of libraries:

```
Ada                           : Ada sources of PHC
 |-- System                   : 0. UNIX dependencies, e.g.: timing
 |-- Math_Lib                 : 1. general mathematical library
 |      |-- Numbers           : 1.1. number representations
 |      |-- Matrices          : 1.2. matrices and linear-system solvers
 |      |-- Polynomials       : 1.3. multivariate polynomial systems
 |      |-- Supports          : 1.4. support sets and linear programming
 |-- Homotopy                 : 2. homotopy and solution lists
 |-- Continuation             : 3. path-tracking routines
 |-- Root_Counts              : 4. root counts and homotopy construction
 |      |-- Product           : 4.1. linear-product start systems
 |      |-- Implift           : 4.2. implicit lifting
 |      |-- Stalift           : 4.3. static lifting
 |      |-- Dynlift           : 4.4. dynamic lifting
 |      |-- Symmetry          : 4.5. exploitation of symmetry relations
 |-- Schubert                 : 5. numerical Schubert calculus
 |-- Main                     : 6. main dispatcher
```

The structure reflects the discrete and continuous nature of the program.

The concept of information hiding has been applied more deeply than just separating the four stages of the solver. Next are some examples on how PHC deals with polynomials.

(1) The continuation is not only separated from the choice of the homotopy, but also from the way polynomials are evaluated. This is done by providing the evaluation and differentiation of the homotopy as parameters of the path trackers.

(2) For the evaluation of polynomials, a multivariate Horner scheme is implemented at the level of the polynomial package. The precise definition is hidden to the client procedures that create and evaluate these polynomials.

(3) A polynomial homotopy can be evaluated more efficiently when the coefficients are parameters to the evaluation routines. The combination with multivariate Horner yields a powerful and flexible coefficient homotopy.

Without using this third data structure, the construction of a random coefficient start system for the cyclic 7-roots problem (924 paths to follow, about 100 cells) by means of polyhedral homotopy continuation took 4h 38m 55s 354ms cpu time. The current version takes only about 15m 49s 391ms cpu time! Polyhedral homotopies are nonlinear in the continuation parameter t and treating t^ω as just one monomial or as a polynomial of degree ω makes the difference in evaluation. This third data structure was created to deal with floating-point lifting values ω implementing a suggestion of T.Y. Li.

Note that coefficient-parameter polynomial continuation [Morgan and Sommese 1989] or cheater's homotopy [Li, Sauer and Yorke 1989], [Li and Wang 1992] is very useful and natural concept.

The computational bottleneck in polynomial continuation is the evaluation of polynomials. The efficiency could improve a lot if the polynomials would be known at compile time, so that optimized in-line evaluators can be used. However, to keep the program user-friendly, compilation of the program must not be required each time a new system has to be solved.

8. ON PORTABILITY: COMPUTERS AND COMPILERS

PHC is written in Ada. Since the gnu-ada compiler (also called GNAT [Ada Core Technologies 1997]) is available for a large number of platforms, the software is portable. The myth that Ada is "big and slow" is disproved in [Syiek 1995]: Ada versions even have a slight edge over their C counterparts.

Initially, the VADS (Verdix Ada Development System) compiler was used on three different machine architectures. The implementation started [Verschelde 1990] on a SUN3/280 and moved [Verschelde and Cools 1993a] to a DECStation 5240. A version for an IBM RS/6000 workstation was made available [Verschelde 1995]. Thirdly, SUN-SPARC machines were used [Verschelde and Cools 1996], along with the gnu-ada compiler version 3.03 in [Verschelde 1996].

Next a report is given on compiler experiences. Because runtime efficiency is crucial, compilation is done with full optimization and with a suppression of runtime checks. Table 2 contains experimental data comparing VADS 6.2.3b against gnu-ada 3.09.

compiler options	user time	sys time	total time	aux. files	executable
VADS: -O -S	40m 9s	5m 8s	45m 17s	42.1 Mb	4.0 Mb
GNAT: -O3 -gnatp	45m 46s	2m 59s	48m 47s	4.1 Mb	2.9 Mb

Table 2. Compiler efficiency (on SPARCserver-1000): Compilation timings and sizes of auxiliary files and executable.

In Table 3 the performance of the generated code is illustrated on one of the benchmark examples.

phc	root counts	start system	continuation	total time
VADS	1m 26s 866ms	24m 20s 589ms	38m 17s 667ms	1h 4m 42s 129ms
GNAT	1m 15s 74ms	15m 49s 391ms	27m 50s 521ms	45m 21s 434ms

Table 3. Performance of generated code, on the cyclic 7-roots problem with the black-box solver compiled with VADS and GNAT. Timings are listed for root counting, construction of start system, the continuation to the target system and the total time.

Although this comparison is by no means thorough, the gnu-ada compiler seems to be the winner, both in compiling and runtime efficiency. Currently, the gnu-ada compiler is maintained by a privately held company Ada Core Technologies (ACT), founded by the creators of the gnu-ada compiler. ACT is committed to provide publicly free releases of their compiler. The most recent public version is numbered 3.11p.

The fourth platform used to develop PHC is a Pentium PC running Linux. The mathematical kernel of PHC has been rewritten using concepts of Ada 95 to incorporate multi-precision facilities. The other major change in the new version is the availability of SAGBI [Verschelde 1998] and Pieri homotopies.

9. PUTTING IT ALL TOGETHER: ONE BLACK-BOX SOLVER

Here the key ingredients are presented to build an overall, general-purpose solver that is reliable and efficient. This allows to solve polynomial systems simply by typing

```
phc -b input output
```

after the prompt.

The outline of the black-box solver is as follows:

1. *Homotopy Construction* :
   1.1. Computation of root counts :
      - D : total degree;
      - B_Z : multi-homogeneous Bézout number, based on a heuristically generated partition;
      - B_S : general linear-product Bézout number, based on a heuristically generated set structure;
      - V : mixed volume, by dynamic lifting, when #different supports $\leq n/2$, by static lifting otherwise.

   1.2. Construction of start system, corresponding to the minimal and least expensive root count.
2. *Polynomial Continuation* :
   2.1. Coefficient and variable scaling.
   2.2. Tracking of the solution paths.
   2.3. Root refinement on the de-scaled solutions.

The computation of root counts by four different methods provides already important information about the problem class. Even though the mixed volume always yields the lowest bound, the construction of the corresponding start system requires continuation and is computationally much more expensive to solve than a linear-product start system. The latter start system is preferred when any of the Bézout numbers equals the mixed volume.

Practical experiments show that scaling the coefficients really helps to smoothen the continuation. For a discussion of scaling see [Morgan, Sommese, and Watson 1989] or [Morgan 1987].

The output file contains:

1. The root counts D, B_Z, B_S, and V.
2. Start system with start solutions.
3. Settings of the path tracker.
4. Results of polynomial continuation on the scaled system.
5. Solutions of original system as output of the root refiner.
6. Timings for the stages and timing summary at the end.

End games are not invoked in the black-box solver, because the determination of the end game operation range tends to be problem dependent. The idea behind this black-box solver is to get quickly an idea of the complexity of the system while providing many root counts. To examine singularities and other degeneracies we recommend switching to the tool mode of PHC.

10. THE TEST DATABASE OF POLYNOMIAL SYSTEMS

The progress on solving polynomial system has always been motivated by the poor performance of the existing technology on practical examples. To make the benchmarking more meaningful and relevant, most systems are taken from practical applications, cited in the literature. We refer to [Traverso 1993] and [Bini and Mourrain 1998] for similar collections.

In Tables 4 and 5 an overview of the application database is given. To save space, the algebraic description is omitted. Either the reference or the PHC distribution file can be consulted. Some systems appear in different guises, such as the cyclic n-roots and the economics problem. It allows us to see how the particular formulation of a problem can influence the solution process.

Some important characteristics of the systems are listed in Tables 6 and 7. Most systems arising from practical applications are *deficient*, i.e.: have fewer roots than the root count. The improvement of the mixed volume compared to the Bézout bounds is in many cases really significant. The black-box solver of PHC does not contain facilities to treat affine roots properly, therefore it may miss some isolated solutions with zero components.

The parameters of the black-box solver have been set to handle all examples of the database. In Tables 8 and 9 timings are listed for a SPARCserver-1000. We count 33 little examples, that require less than one minute to solve. There are 38 larger examples, for which PHC needs between one minute and an hour. Five big examples require more than one hour to solve.

Timings only have a temporary value. They are only good to measure the difficulty of solving one system compared to another one. The exponential gains from selecting a sharp root count are more important. The black-box solver is certainly not the optimal way to solve a particular system, although the timings give a good impression of the general performance of the package.

The demonstration database of polynomial systems is still growing in order to increase the awareness of the importance and relevance of solving polynomial systems to applied mathematics and scientific computing.

In closing some user applications of PHC are mentioned. PHC was used actively by Charles Wampler [1996] to count the roots of various systems in mechanical design. Frank Sottile applied PHC to compute root counts for linear subspace intersections of the Schubert calculus, see [Sottile 1998] for various tables. A third example comes from computer graphics. To show that the 12 lines tangent to four given spheres can all be real, Thorsten Theobald used PHC, choosing appropriate parameters in the algebraic formulation set up by Cassiano Durand.

name	reference	title with description of the application
boon	[Boon 1992]	neurophysiology, posted by Sjirk Boon
butcher	[Traverso 1993]	Butcher's problem, from PoSSo test suite
butcher8	[Boege et al. 1986]	8-variable version of Butcher's problem
camera1s	[Emiris 1997]	camera displacement between 2 positions, frame 1
caprasse	[Traverso 1993]	the system caprasse of the PoSSo test suite
cassou	[Li et al. 1996]	the system of Pierrette Cassou-Noguès
chemequ	[Meintjes and Morgan 1990]	chemical equilibrium of hydrocarbon combustion
cohn2	[Cohn and Deutch 1988]	modular equations for algebraic number fields
cohn3	[Galligo and Traverso 1989]	modular equations for algebraic number fields
comb3000	[Morgan 1987]	Model A combustion chemistry example
conform1	[Emiris 1997]	conformal analysis of cyclic molecules, instance 1
cpdm5	[Gatermann 1990]	5-dimensional system of Caprasse and Demaret
cyclic5	[Björk and Fröberg 1991]	cyclic 5-roots problem
cyclic6	[Björk and Fröberg 1991]	cyclic 6-roots problem
cyclic7	[Backelin and Fröberg 1991]	cyclic 7-roots problem
cyclic8	[Björk and Fröberg 1994]	cyclic 8-roots problem
d1	[Van Hentenryck et al. 1997]	a sparse system, known as benchmark D1
des18_3	[Nauheim 1998]	a "dessin d'enfant", called des18_3
des22_24	[Nauheim 1998]	a "dessin d'enfant", called des22_24
discret3s	[Traverso 1993]	system discret3, scaled by average coefficients
eco5	[Morgan 1987]	5-dimensional economics problem
eco6	[Morgan 1987]	6-dimensional economics problem
eco7	[Morgan 1987]	7-dimensional economics problem
eco8	[Morgan 1987]	8-dimensional economics problem
extcyc5	[Verschelde and Gatermann 1995]	extended cyclic 5-roots to exploit symmetry
extcyc6	[Verschelde and Gatermann 1995]	extended cyclic 6-roots to exploit symmetry
extcyc7	[Verschelde and Gatermann 1995]	extended cyclic 7-roots to exploit symmetry
extcyc8	[Verschelde and Gatermann 1995]	extended cyclic 8-roots to exploit symmetry
fourbar	[Morgan and Wampler 1990]	four-bar design problem, so-called 5-point problem
fbrfive4	[Wampler 1996]	Four-bar linkage through 5 points, n=4 version
fbrfive12	[Wampler 1996]	Four-bar linkage, coupler curve through 5 points
gaukwa2	[Stroud and Secrest 1966]	Gaussian quadrature formula 2 knots, 2 weights
gaukwa3	[Stroud and Secrest 1966]	Gaussian quadrature formula 3 knots, 3 weights
gaukwa4	[Stroud and Secrest 1966]	Gaussian quadrature formula 4 knots, 4 weights
geneig	[Chu et al. 1988]	generalized eigenvalue problem
heart	[Nelsen and Hodgkin 1981]	heart-dipole problem
i1	[Van Hentenryck et al. 1997]	Benchmark i1 from Interval Arithmetic Benchmarks
ipp	[Morgan and Sommese 1987b]	six-revolute-joint problem of mechanics

Table 4. An overview of the test database, part I. Besides the name of the polynomial system, a reference to the literature and a short description is mentioned.

11. OBTAINING AND INSTALLING PHC

The current second release of PHC is available at the web pages of the author. The distribution contains the Ada sources with makefiles to install with the gnu-ada compiler, the database of test examples, and executable versions for Unix workstations SUN, SGI, PCs running Linux and Solaris. For other platforms the gnu-ada compiler is needed.

The affiliations of the author will change. In case the current web addresses are obsolete, or to report other (installation) problems, please feel free to contact the author sending an e-mail message to `jan.verschelde@na-net.ornl.gov`.

12. CONCLUSIONS AND FUTURE DEVELOPMENTS

PHC offers a general-purpose solver for polynomial systems that features recent research advances in root-counting methods. The software package is portable via the gnu-ada compiler. Practical evidence for the performance of its black-box solver has been given on a large set of applications.

name	reference	title with description of the application
ipp2	[Wampler and Morgan 1991]	6R inverse position problem
katsura5	[Boege et al. 1986]	a problem of magnetism in physics
kinema	[Bellido 1992]	robot kinematics problem
kin1	[Van Hentenryck et al. 1997]	kinematics problem
ku10	[Steenkamp 1982]	10-dimensional system of Ku
lorentz	[Li 1987]	equilibrium of 4-dimensional Lorentz attractor
lumped	[Li and Wang 1991]	lumped-parameter chemically reacting system
mickey	[Verschelde and Cools 1996]	Mickey-mouse example as illustration
noon3	[Noonburg 1989]	neural network, Lotka-Volterra system, n=3
noon4	[Noonburg 1989]	neural network, Lotka-Volterra system, n=4
noon5	[Noonburg 1989]	neural network, Lotka-Volterra system, n=5
proddeco	[Morgan et al. 1995]	system with a product-decomposition structure
puma	[Morgan and Shapiro 1987]	hand position and orientation of PUMA robot
quadfor2	[Verschelde and Gatermann 1995]	Gaussian quadrature with 2 knots and weights
quadgrid	[Sweldens 1994]	interpolating quadrature formula on a grid
rabmo	[Moore and Jones 1977]	optimal multi-dimensional quadrature formulas
rbpl	[Mourrain 1993]	parallel robot, the so-called left-hand problem
rbpl24	[Mourrain 1996]	parallel robot with 24 real solutions
redcyc5	[Emiris 1994]	reduced cyclic 5-roots problem
redcyc6	[Emiris 1994]	reduced cyclic 6-roots problem
redcyc7	[Emiris 1994]	reduced cyclic 7-roots problem
redcyc8	[Emiris 1994]	reduced cyclic 8-roots problem
redeco5	[Morgan 1987]	reduced 5-dimensional economics problem
redeco6	[Morgan 1987]	reduced 6-dimensional economics problem
redeco7	[Morgan 1987]	reduced 7-dimensional economics problem
redeco8	[Morgan 1987]	reduced 8-dimensional economics problem
rediff3	[Iserles 1995]	3-dimensional reaction-diffusion problem
reimer5	[Traverso 1993]	The 5-dimensional system of Reimer
rose	[Traverso 1993]	a general economic equilibrium model
s9_1	[Nauheim 1998]	small system from constructive Galois theory
sendra	[Traverso 1993]	the system sendra of the PoSSo test suite
solotarev	[Traverso 1993]	the system solotarev of the PoSSo test suite
sparse5	[Verschelde and Gatermann 1995]	5-dimensional sparse symmetric polynomial system
speer	[Gatermann 1990]	the system of E.R. Speer
trinks	[Traverso 1993]	system of Trinks from the PoSSo test suite
virasoro	[Schrans and Troost 1990]	the construction of Virasoro algebras
wood	[Moré et al. 1981]	system derived from optimizing Wood function
wright	[Wright 1985]	system of A.H. Wright

Table 5. An overview of the test database, part II. Besides the name of the polynomial system, a reference to the literature and a short description is mentioned.

A recent exciting development concerns homotopies that solve problems in enumerative geometry, see [Sottile 1997] and [Huber, Sottile and Sturmfels 1998]. This class of homotopies is already incorporated in the current second release of the package. As future (maybe futuristic) long-term project we dream of a comprehensive framework for constructing homotopies, integrating the PACKage approach of numerical analysis with the general-purpose solvers of computer algebra.

ACKNOWLEDGMENTS

The author acknowledges the technical assistance, usage of PHC, concern, comments and advice of the following people:

name	n	D	B_Z	B_S	V	#sols
boon	6	1024	344	216	20	8
butcher	7	4608	2090	605	24	5
butcher8	8	4608	1461	587	26	16
cameral1s	6	64	20	20	20	20
caprasse	4	144	62	94	48	48
cassou	4	1344	368	361	24	16
chemequ	5	108	56	44	16	16
cohn2	4	900	468	358	124	18
cohn3	4	1080	484	358	213	102
comb3000	10	96	66	28	16	16
conform1	3	64	16	16	16	16
cpdm5	5	243	243	243	242	157
cyclic5	5	120	120	106	70	70
cyclic6	6	720	720	588	156	156
cyclic7	7	5040	5040	4200	924	924
cyclic8	8	40320	40320	30365	2560	1152
d1	12	4068	320	896	192	48
des18_3	8	324	544	241	46	46
des22_24	10	256	128	82	42	42
discret3s	8	256	128	128	128	128
eco5	5	54	20	16	8	8
eco6	6	162	48	36	16	16
eco7	7	486	112	80	32	32
eco8	8	1458	256	176	64	64
extcyc5	5	120	120	106	70	70
extcyc6	6	720	720	588	156	156
extcyc7	7	5040	5040	4200	924	924
extcyc8	8	40320	40320	30365	2560	1152
fbrfive12	12	4096	96	96	36	36
fbrfive4	4	256	96	194	36	36
fourbar	4	256	96	96	80	36
gaukwa2	4	24	11	11	5	2
gaukwa3	6	720	225	225	49	6
gaukwa4	8	40320	6769	6769	729	24
geneig	6	243	10	10	10	10
heart	8	576	193	193	121	4
i1	10	59049	452	437	66	66
ipp	8	256	96	96	64	48

Table 6. Characteristics of the polynomial systems, part I. The dimension is n. D is the total degree of the system. B_Z is an m-homogeneous Bézout number, based on a partition generated by a heuristic method. B_S is a generalized linear-product Bézout number, based on a set structure generated by a heuristic method. V is the mixed volume. #sols is number of isolated solutions found by PHC.

Yvan Barbaix
Marc Beckers
Anita Ceulemans
Ronald Cools
Dirk Craeynest
Cassiano Durand
Karel de Vlaminck
Ioannis Emiris
Tangan Gao
Karin Gatermann
Ann Haegemans
Birk Huber
Tae Sung Kim
Dirk Kinnaes
Tien-Yien Li
Tom Michiels
Maurice Rojas
Andrew Sommese
Frank Sottile
Thorsten Theobald
Pierre Verlinden
Charles Wampler
Xiaoshen Wang
Layne Watson
Mengnien Wu
Li Xing

The author thanks the referees for many helpful comments and useful suggestions on the first version.

name	n	D	B_Z	B_S	V	#sols
ipp2	11	1024	576	848	288	16
katsura5	6	32	32	32	32	32
kin1	12	4608	320	896	192	48
kinema	9	64	240	64	64	40
ku10	10	1024	2	2	2	2
lorentz	4	16	14	12	12	11
lumped	4	16	8	11	7	4
mickey	2	4	4	4	4	4
noon3	3	27	29	21	21	21
noon4	4	81	81	73	73	73
noon5	5	243	243	233	233	233
proddeco	4	256	96	96	26	6
puma	8	128	16	32	16	16
quadfor2	4	24	11	11	4	2
quadgrid	5	120	10	10	10	5
rabmo	9	36000	22740	7090	136	16
rbpl	6	486	160	160	160	150
rbpl24	9	576	80	80	80	40
redcyc5	4	24	24	19	14	14
redcyc6	5	120	96	83	26	26
redcyc7	6	720	720	511	132	132
redcyc8	7	5040	3960	3107	320	144
redeco5	5	8	12	8	8	8
redeco6	6	16	28	16	16	16
redeco7	7	32	64	32	32	32
redeco8	8	64	144	64	64	64
rediff3	3	8	8	8	7	7
reimer5	5	720	720	720	720	144
rose	3	216	144	136	136	136
s9_1	8	16	41	10	10	10
sendra	2	49	49	46	46	46
solotarev	4	36	10	8	6	6
sparse5	5	100000	3840	3840	160	160
speer	4	625	384	246	96	43
trinks	6	24	24	18	10	10
virasoro	8	256	3072	256	200	200
wood	4	36	25	16	9	9
wright	5	32	32	32	32	32

Table 7. Characteristics of the polynomial systems, part II. The dimension is n. D is the total degree of the system. B_Z is an m-homogeneous Bézout number, based on a partition generated by a heuristic method. B_S is a general linear-product Bézout number, based on a set structure generated by a heuristic method. V is the mixed volume. #sols is number of isolated solutions found by PHC.

REFERENCES

Allison, D. C. S., Chakraborty, A., and Watson, L. T. 1989. Granularity issues for solving polynomial systems via globally convergent algorithms on a hypercube. *J. of Supercomputing 3*, 5–20.

Backelin, J. and Fröberg, R. 1991. How we proved that there are exactly 924 cyclic 7-roots. In *Proceedings of ISSAC-91* (1991), pp. 101–111. ACM.

Bellido, A. M. 1992. Construction of iteration functions for the simultaneous computation of the solutions of equations and algebraic systems. *Numerical Algorithms 6*, 3–4, 317–351.

Bernshteĭn, D. N. 1975. The number of roots of a system of equations. *Functional Anal. Appl. 9*, 3, 183–185. Translated from *Funktsional. Anal. i Prilozhen.*, 9(3):1–4,1975.

Bini, D. and Mourrain, B. 1998. Polynomial test suite. Available at http://www-sop.inria.fr/saga/POL/.

Björk, G. and Fröberg, R. 1991. A faster way to count the solutions of inhomogeneous systems of algebraic equations, with applications to cyclic n-roots. *J. Symbolic Computation 12*, 3, 329–336.

name	root counts				start system				continuation				total time			
boon	0h	0m	0s	190	0h	0m	5s	868	0h	0m	14s	394	0h	0m	20s	937
butcher	0h	0m	13s	267	0h	0m	29s	23	0h	1m	44s	962	0h	2m	28s	449
butcher8	0h	0m	50s	513	0h	0m	25s	188	0h	4m	20s	602	0h	5m	38s	507
camera1s	0h	0m	8s	258	0h	0m	0s	110	0h	0m	34s	406	0h	0m	44s	682
caprasse	0h	0m	0s	769	0h	0m	10s	4	0h	0m	17s	80	0h	0m	28s	888
cassou	0h	0m	1s	145	0h	0m	10s	688	0h	1m	3s	439	0h	1m	15s	972
chemequ	0h	0m	1s	116	0h	0m	4s	827	0h	0m	6s	886	0h	0m	13s	378
cohn2	0h	0m	3s	989	0h	0m	49s	953	0h	2m	49s	74	0h	3m	46s	619
cohn3	0h	0m	4s	991	0h	1m	12s	618	0h	16m	15s	864	0h	17m	37s	282
comb3000	0h	0m	7s	814	0h	0m	5s	630	0h	0m	18s	162	0h	0m	33s	118
conform1	0h	0m	0s	42	0h	0m	0s	45	0h	0m	3s	880	0h	0m	4s	310
cpdm5	0h	0m	18s	683	0h	2m	27s	225	0h	9m	51s	598	0h	12m	43s	370
cyclic5	0h	0m	0s	562	0h	0m	11s	768	0h	0m	32s	469	0h	0m	45s	993
cyclic6	0h	0m	6s	40	0h	1m	15s	816	0h	2m	44s	292	0h	4m	9s	434
cyclic7	0h	1m	15s	74	0h	15m	49s	391	0h	27m	50s	521	0h	45m	21s	434
cyclic8	0h	14m	41s	38	1h	25m	14s	851	2h	54m	28s	884	4h	35m	54s	367
d1	0h	0m	15s	182	0h	5m	34s	397	0h	13m	30s	348	0h	19m	25s	426
des18_3	0h	3m	51s	209	0h	1m	19s	587	0h	1m	44s	25	0h	6m	57s	913
des22_24	0h	0m	23s	538	0h	0m	50s	660	0h	1m	22s	251	0h	2m	40s	53
discret3s	0h	2m	20s	4	0h	0m	0s	719	0h	56m	20s	922	0h	58m	52s	121
eco5	0h	0m	0s	281	0h	0m	1s	222	0h	0m	2s	829	0h	0m	4s	686
eco6	0h	0m	2s	217	0h	0m	4s	132	0h	0m	6s	771	0h	0m	13s	785
eco7	0h	0m	22s	215	0h	0m	20s	511	0h	0m	34s	19	0h	1m	18s	249
eco8	0h	5m	28s	66	0h	1m	1s	471	0h	1m	55s	472	0h	8m	27s	528
extcyc5	0h	0m	2s	355	0h	0m	36s	607	0h	0m	37s	521	0h	1m	17s	726
extcyc6	0h	0m	30s	15	0h	1m	45s	137	0h	2m	56s	572	0h	5m	15s	706
extcyc7	0h	9m	15s	674	0h	17m	22s	278	0h	30m	29s	581	0h	57m	33s	835
extcyc8	1h	58m	58s	816	1h	36m	57s	179	3h	58m	54s	943	7h	36m	30s	580
fbrfive12	0h	1m	30s	161	0h	1m	6s	686	0h	2m	18s	859	0h	4m	58s	570
fbrfive4	0h	0m	0s	463	0h	0m	17s	283	0h	1m	2s	690	0h	1m	21s	972
fourbar	0h	0m	0s	625	0h	0m	8s	706	0h	2m	1s	20	0h	2m	12s	565
gaukwa2	0h	0m	0s	55	0h	0m	0s	705	0h	0m	1s	460	0h	0m	2s	391
gaukwa3	0h	0m	1s	430	0h	0m	22s	803	0h	0m	52s	615	0h	1m	17s	674
gaukwa4	0h	1m	18s	940	0h	27m	42s	595	0h	52m	30s	671	1h	21m	42s	787
geneig	0h	0m	0s	476	0h	0m	0s	303	0h	0m	11s	314	0h	0m	14s	648
heart	0h	1m	10s	899	0h	3m	14s	236	0h	4m	39s	75	0h	9m	6s	712
i1	0h	0m	37s	869	0h	0m	49s	215	0h	1m	59s	959	0h	3m	29s	913
ipp	0h	1m	4s	541	0h	1m	21s	14	0h	1m	47s	940	0h	4m	16s	287

Table 8. Part I of timing summary on SPARCserver-1000, with the black-box version made with the gnu-ada compiler. The CPU time of the process is expressed in hours, minutes, seconds, and milliseconds.

name	root counts				start system				continuation				total time			
ipp2	0h	8m	21s	346	0h	11m	59s	944	0h	27m	18s	539	0h	47m	48s	632
katsura5	0h	0m	12s	382	0h	0m	0s	8	0h	0m	27s	511	0h	0m	41s	303
kin1	0h	2m	2s	926	0h	5m	39s	399	0h	11m	37s	403	0h	19m	25s	259
kinema	0h	2m	28s	847	0h	0m	0s	22	0h	3m	11s	205	0h	5m	42s	168
ku10	0h	0m	1s	547	0h	0m	2s	253	0h	0m	3s	611	0h	0m	8s	357
lorentz	0h	0m	0s	173	0h	0m	0s	23	0h	0m	6s	633	0h	0m	7s	256
lumped	0h	0m	0s	289	0h	0m	0s	714	0h	0m	1s	975	0h	0m	3s	255
mickey	0h	0m	0s	8	0h	0m	0s	2	0h	0m	0s	175	0h	0m	0s	248
noon3	0h	0m	0s	58	0h	0m	0s	33	0h	0m	4s	639	0h	0m	5s	154
noon4	0h	0m	0s	399	0h	0m	0s	103	0h	0m	51s	160	0h	0m	53s	163
noon5	0h	0m	2s	922	0h	0m	0s	316	0h	7m	9s	741	0h	7m	17s	834
proddeco	0h	0m	0s	299	0h	0m	10s	135	0h	0m	42s	488	0h	0m	54s	26
puma	0h	0m	2s	457	0h	0m	0s	420	0h	0m	34s	43	0h	0m	40s	105
quadfor2	0h	0m	0s	24	0h	0m	0s	183	0h	0m	0s	910	0h	0m	1s	271
quadgrid	0h	0m	4s	451	0h	0m	0s	159	0h	0m	14s	627	0h	0m	20s	443
rabmo	0h	1m	10s	899	0h	3m	14s	236	0h	4m	39s	75	0h	9m	6s	712
rbpl	0h	1m	32s	693	0h	0m	0s	670	0h	10m	24s	859	0h	12m	4s	957
rbpl24	3h	19m	40s	323	0h	0m	1s	331	0h	9m	55s	836	3h	29m	47s	764
redcyc5	0h	0m	0s	348	0h	0m	2s	767	0h	0m	3s	505	0h	0m	6s	995
redcyc6	0h	0m	3s	559	0h	0m	9s	911	0h	0m	17s	549	0h	0m	31s	863
redcyc7	0h	0m	54s	118	0h	1m	56s	885	0h	3m	12s	794	0h	6m	7s	216
redcyc8	0h	9m	58s	283	0h	10m	54s	911	0h	18m	11s	414	0h	39m	14s	893
redeco5	0h	0m	0s	290	0h	0m	0s	3	0h	0m	3s	109	0h	0m	3s	671
redeco6	0h	0m	2s	389	0h	0m	0s	6	0h	0m	7s	958	0h	0m	10s	860
redeco7	0h	0m	22s	751	0h	0m	0s	10	0h	0m	24s	990	0h	0m	48s	725
redeco8	0h	4m	30s	648	0h	0m	0s	18	0h	1m	20s	991	0h	5m	53s	618
rediff3	0h	0m	0s	28	0h	0m	0s	225	0h	0m	0s	568	0h	0m	1s	2
reimer5	0h	0m	0s	913	0h	0m	0s	65	0h	9m	20s	820	0h	9m	30s	537
rose	0h	0m	0s	86	0h	0m	0s	347	0h	2m	26s	320	0h	2m	30s	262
s9_1	0h	0m	0s	663	0h	0m	0s	65	0h	0m	14s	843	0h	0m	17s	344
sendra	0h	0m	0s	50	0h	0m	0s	131	0h	0m	18s	328	0h	0m	19s	474
solotarev	0h	0m	0s	79	0h	0m	0s	365	0h	0m	1s	37	0h	0m	1s	695
sparse5	0h	0m	0s	396	0h	0m	21s	672	0h	3m	45s	456	0h	4m	11s	36
speer	0h	0m	1s	554	0h	0m	52s	9	0h	13m	35s	109	0h	14m	31s	786
trinks	0h	0m	0s	666	0h	0m	1s	855	0h	0m	5s	962	0h	0m	8s	967
virasoro	3h	40m	50s	731	0h	7m	0s	273	0h	7m	42s	993	3h	55m	41s	856
wood	0h	0m	0s	55	0h	0m	0s	761	0h	0m	2s	860	0h	0m	3s	912
wright	0h	0m	1s	332	0h	0m	0s	6	0h	0m	13s	251	0h	0m	15s	210

Table 9. Part II of timing summary on SPARCserver-1000, with the black-box version made with the gnu-ada compiler. The CPU time of the process is expressed in hours, minutes, seconds, and milliseconds.

Björk, G. and Fröberg, R. 1994. Methods to "divide out" certain solutions from systems of algebraic equations, applied to find all cyclic 8-roots. In M. Gyllenberg and L. Persson Eds., *Analysis, Algebra and Computers in Math. research*, Volume 564 of *Lecture Notes in Applied Mathematics*, pp. 57–70. Marcel Dekker.

Blum, L., Cucker, F., Shub, M., and Smale, S. 1997. *Complexity and Real Computation.* Springer–Verlag, New York.

Boege, W., Gebauer, R., and Kredel, H. 1986. Some examples for solving systems of algebraic equations by calculating Groebner bases. *J. Symbolic Computation 2*, 83–98.

Boon, S. 1992. Solving systems of equations. Article 2972 of sci.math.symbolic and Article 3529 of sci.math.num-analysis, 25 June 1992.

Canny, J. and Rojas, J. M. 1991. An optimal condition for determining the exact number of roots of a polynomial system. In *Proceedings of ISSAC-91* (1991), pp. 96–101. ACM.

Chu, M., Li, T. Y., and Sauer, T. 1988. Homotopy method for general λ–Matrix problems. *SIAM J. Matrix Anal. Appl. 9*, 4, 528–536.

Cohn, H. and Deutch, J. 1988. An explit modular equation in two variables for Q[sqrt(3)]. *Math. Comp. 50*, 557–568.

Cox, D., Little, J., and O'Shea, D. 1998. *Using Algebraic Geometry*, Volume 185 of

Graduate Texts in Mathematics. Springer–Verlag, New York.

EMIRIS, I. Z. 1994. *Sparse Elimination and Applications in Kinematics*. Ph. D. thesis, Computer Science Division, Dept. of Electrical Engineering and Computer Science, University of California, Berkeley. Available at http://www.inria.fr/saga/emiris.

EMIRIS, I. Z. 1997. A general solver based on sparse resultants: Numerical issues and kinematic applications. Rapport de recherche no. 3110, INRIA, France. Available via anonymous ftp to ftp.inria.fr.

EMIRIS, I. Z. 1998. Symbolic-numeric algebra for polynomials. In A. KENT AND J. WILLIAMS Eds., *Encyclopedia of Computer Science and Technology*, Volume 39 of *Encyclopedia of Computer Science*, pp. 261–281. Marcel Dekker.

EMIRIS, I. Z. AND CANNY, J. F. 1995. Efficient incremental algorithms for the sparse resultant and the mixed volume. *J. Symbolic Computation 20*, 2, 117–149. Software available at http://www.inria.fr/saga/emiris.

EMIRIS, I. Z. AND VERSCHELDE, J. 1997. How to count efficiently all affine roots of a polynomial system. Rapport de recherche no. 3212, INRIA. To appear in *Discrete Applied Mathematics*.

GALLIGO, A. AND TRAVERSO, C. 1989. Practical determination of the dimension of an algebraic variety. In E. KALTOFEN AND S. M. WATT Eds., *Computers and Mathematics*, pp. 46–52. Berlin, New York: Springer–Verlag.

GAO, T., LI, T. Y., AND WANG, X. 1997. Finding isolated zeros of polynomial systems in C^n with stable mixed volumes. To appear in *J. of Symbolic Computation*.

GAREY, M. AND JOHNSON, D. 1979. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. Freemann, San Francisco.

GATERMANN, K. 1990. Symbolic solution of polynomial equation systems with symmetry. In S. WATANABE AND M. NAGATA Eds., *Proceedings of ISSAC-90* (Tokyo, Japan, 1990), pp. 112–119. ACM.

GEL'FAND, I. M., KAPRANOV, M. M., AND ZELEVINSKY, A. V. 1994. *Discriminants, Resultants and Multidimensional Determinants*. Birkhäuser, Boston.

GIORDANO, T. 1996. Implémention distribuée du calcul du volume mixte. Master's thesis, University of Nice, Sophia-Antipolis.

HARIMOTO, S. AND WATSON, L. T. 1989. The granularity of homotopy algorithms for polynomial systems of equations. In G. RODRIGUE Ed., *Parallel processing for scientific computing* (1989), pp. 115–120. SIAM.

HUBER, B. 1995. Pelican manual. Available at http://www.mrsi.org/people/members/birk.

HUBER, B., SOTTILE, F., AND STURMFELS, B. 1998. Numerical Schubert calculus. *J. of Symbolic Computation 26*, 6, 767–788.

HUBER, B. AND STURMFELS, B. 1995. A polyhedral method for solving sparse polynomial systems. *Math. Comp. 64*, 212, 1541–1555.

HUBER, B. AND STURMFELS, B. 1997. Bernstein's theorem in affine space. *Discrete Comput. Geom. 17*, 2, 137–141.

HUBER, B. AND VERSCHELDE, J. 1998. Polyhedral end games for polynomial continuation. *Numerical Algorithms 18*, 1, 91–108.

HUBER, B. T. 1996. *Solving Sparse Polynomial Systems*. Ph. D. thesis, Cornell University. Available at http://www.msri.org/people/members/birk.

ISERLES, A. 1995. Personal communication at the occasion of the AMS-SIAM Summer Seminar in Applied Mathematics, Park City, Utah, July 17-August 11, 1995, Park City, Utah.

KHOVANSKIĬ, A. 1991. *Fewnomials*, Volume 88 of *Translations of Mathematical Monographs*. AMS, Providence, Rhode Island.

KLEIMAN, S. AND LAKSOV, D. 1972. Schubert calculus. *American Mathematical Monthly 79*, 10, 1061–1082.

KUSHNIRENKO, A. 1976. Newton Polytopes and the Bézout Theorem. *Functional Anal. Appl. 10*, 3, 233–235. Translated from *Funktsional. Anal. i Prilozhen.*, 10(3),82–83,1976.

LI, T. Y. 1987. Solving polynomial systems. *The Mathematical Intelligencer 9*, 3, 33–39.

LI, T. Y. 1997. Numerical solutions of multivariate polynomial systems by homotopy continuation methods. *Acta Numerica 6*, 399–436.

LI, T. Y. AND SAUER, T. 1987. Regularity results for solving systems of polynomials by homotopy method. *Numer. Math. 50*, 3, 283–289.

LI, T. Y., SAUER, T., AND YORKE, J. A. 1987a. Numerical solution of a class of deficient polynomial systems. *SIAM J. Numer. Anal. 24*, 2, 435–451.

LI, T. Y., SAUER, T., AND YORKE, J. A. 1987b. The random product homotopy and deficient polynomial systems. *Numer. Math. 51*, 5, 481–500.

LI, T. Y., SAUER, T., AND YORKE, J. A. 1989. The cheater's homotopy: an efficient procedure for solving systems of polynomial equations. *SIAM J. Numer. Anal. 26*, 5, 1241–1251.

LI, T. Y., WANG, T., AND WANG, X. 1996. Random product homotopy with minimal BKK bound. In J. RENEGAR, M. SHUB, AND S. SMALE Eds., *The Mathematics of Numerical Analysis*, Volume 32 of *Lectures in Applied Mathematics* (Park City, Utah, 1996).

LI, T. Y. AND WANG, X. 1991. Solving deficient polynomial systems with homotopies which keep the subschemes at infinity invariant. *Math. Comp. 56*, 194, 693–710.

LI, T. Y. AND WANG, X. 1992. Nonlinear homotopies for solving deficient polynomial systems with parameters. *SIAM J. Numer. Anal. 29*, 4, 1104–1118.

LI, T. Y. AND WANG, X. 1996. The BKK root count in C^n. *Math. Comp. 65*, 216, 1477–1484.

MALAJOVICH, G. 1996. pss 2.beta, polynomial system solver, version 2.beta. Available at http://www.labma.ufrj.br:80/~gregorio.

ADA CORE TECHNOLOGIES. 1997. *GNAT, User's Guide. GNAT, The GNU Ada 95 Compiler Version 3.09.* ACT. Available at http://www.gnat.com.

THE FRISCO CONSORTIUM. 1996. FRISCO – A Framework for Integrated Symbolic/Numeric Computation. Available at http://www.nag.co.uk/projects/FRISCO.html.

THE PISA TEAM OF POSSO. 1993. PoSSo home page. Available at http://janet.dm.unipi.it/.

MEINTJES, K. AND MORGAN, A. P. 1990. Chemical equilibrium systems as numerical test problems. *ACM Trans. Math. Soft. 16*, 2, 143–151.

MOORE, R. E. AND JONES, S. T. 1977. Safe starting regions for iterative methods. *SIAM J. Numer. Anal. 14*, 6, 1051–1065.

MORÉ, J., GARBOW, B., AND HILLSTROM, K. 1981. Testing unconstrained optimization software. *ACM Trans. Math. Softw. 7*, 1, 17–41.

MORGAN, A. 1987. *Solving polynomial systems using continuation for engineering and scientific problems.* Prentice-Hall, Englewood Cliffs, N.J.

MORGAN, A. AND SHAPIRO, V. 1987. Box-bisection for solving second-degree systems and the problem of clustering. *ACM Trans. Math. Soft. 13*, 2, 152–167.

MORGAN, A. AND SOMMESE, A. 1987a. Computing all solutions to polynomial systems using homotopy continuation. *Appl. Math. Comput. 24*, 2, 115–138.

MORGAN, A. AND SOMMESE, A. 1987b. A homotopy for solving general polynomial systems that respects m-homogeneous structures. *Appl. Math. Comput. 24*, 2, 101–113.

MORGAN, A. P. AND SOMMESE, A. J. 1989. Coefficient-parameter polynomial continuation. *Appl. Math. Comput. 29*, 2, 123–160. Errata: *Appl. Math. Comput.* 51:207(1992).

MORGAN, A. P., SOMMESE, A. J., AND WAMPLER, C. W. 1991. Computing singular solutions to nonlinear analytic systems. *Numer. Math. 58*, 7, 669–684.

MORGAN, A. P., SOMMESE, A. J., AND WAMPLER, C. W. 1992a. Computing singular solutions to polynomial systems. *Adv. Appl. Math. 13*, 3, 305–327.

MORGAN, A. P., SOMMESE, A. J., AND WAMPLER, C. W. 1992b. A power series method for computing singular solutions to nonlinear analytic systems. *Numer. Math. 63*, 391–409.

MORGAN, A. P., SOMMESE, A. J., AND WAMPLER, C. W. 1995. A product-decomposition theorem for bounding Bézout numbers. *SIAM J. Numer. Anal. 32*, 4, 1308–1325.

MORGAN, A. P., SOMMESE, A. J., AND WATSON, L. T. 1989. Finding all isolated solutions to polynomial systems using HOMPACK. *ACM Trans. Math. Softw. 15*, 2, 93–122.

MORGAN, A. P. AND WAMPLER, C. W. 1990. Solving a planar four-bar design problem using continuation. *ASME J. of Mechanical Design 112*, 544–550.

MOURRAIN, B. 1993. The 40 generic positions of a parallel robot. In M. BRONSTEIN Ed., *Proceedings of ISSAC-93* (Kiev, Ukraine, 1993), pp. 173–182. ACM.

MOURRAIN, B. 1996. The handbook of polynomial systems. Available at http://www.inria.fr/saga/POL/index.html.

NAUHEIM, R. 1998. Systems of algebraic equations with bad reduction. *J. Symbolic Computation 25*, 619–641.

NELSEN, C. V. AND HODGKIN, B. C. 1981. Determination of magnitudes, directions, and locations of two independent dipoles in a circular conducting region from boundary potential measurements. *IEEE Trans. Biomed. Engrg. BME-28*, 12, 817–823.

NOONBURG, V. W. 1989. A neural network modeled by an adaptive Lotka-Volterra system. *SIAM J. Appl. Math. 49*, 6, 1779–1792.

RAVI, M. S., ROSENTHAL, J., AND WANG, X. 1996. Dynamic pole placement assignment and Schubert calculus. *SIAM J. Control and Optimization 34*, 3, 813–832.

ROJAS, J. M. 1994. A convex geometric approach to counting the roots of a polynomial system. *Theoret. Comput. Sci. 133*, 1, 105–140.

ROJAS, J. M. 1996. Toric intersection theory for affine root counting. To appear in Journal of Pure and Applied Algebra, vol 136, no 1, March 1999. Available at http://www.cityu.edu.hk/ma/staff/rojas.

ROJAS, J. M. 1997. Toric laminations, sparse generalized characteristic polynomials, and a refinement of Hilbert's tenth problem. In F. CUCKER AND M. SHUB Eds., *Foundations of Computational Mathematics. Selected Papers of a Conference, Held at IMPA in Rio de Janeiro, January 1997* (1997), pp. 369–381. Springer–Verlag. Revised version available at http://www.cityu.edu.hk/ma/staff/rojas.

ROJAS, J. M. AND WANG, X. 1996. Counting affine roots roots of polynomial systems via pointed Newton polytopes. *J. Complexity 12*, 116–133.

ROSENTHAL, J. AND SOTTILE, F. 1998. Some remarks on real and complex output feedback. *Systems and Control Lett. 33*, 2, 73–80. See `http://www.nd.edu/~rosen/pole` for a description of computational aspects of the paper.

ROSENTHAL, J. AND WILLEMS, J. C. 1998. Open problems in the area of pole placement. In V. D. BLONDEL, E. D. SONTAG, M. VIDYASAGAR, AND J. C. WILLEMS Eds., *Open Problems in Mathematical Systems and Control Theory*, Communication and Control Engineering Series, pp. 181–191. Berlin, New York: Springer–Verlag.

SCHRANS, S. AND TROOST, W. 1990. Generalized Virasoro constructions for SU(3). *Nuclear Phys. B 345*, 2–3, 584–606.

SOSONKINA, M., WATSON, L. T., AND STEWART, D. E. 1996. Note on the end game in homotopy zero curve tracking. *ACM Trans. Math. Softw. 22*, 3, 281–287.

SOTTILE, F. 1997. Enumerative geometry for real varieties. In J. KOLLÁR, R. LAZARSFELD, AND D. R. MORRISON Eds., *Algebraic Geometry - Santa Cruz 1995 (University of California, Santa Cruz, July 1995)*, Volume 62, Part I of *Proceedings of Symposia in Pure Mathematics*, pp. 435–447. AMS.

SOTTILE, F. 1998. Real Schubert calculus: polynomial systems and a conjecture of Shapiro and Shapiro. Preprint #1998-066, MSRI.

STEENKAMP, M. C. 1982. Die numeriese oplos van stelsels polinoomvergelykings. Technical report, Nasionale Navorsingsinstituut vir Wiskundige Wetenskappe, Pretoria.

STROUD, A. H. AND SECREST, D. 1966. *Gaussian Quadrature Formulas*. Prentice-Hall series in automatic computation. Prentice-Hall, Englewood Cliffs (N.J.).

STURMFELS, B. 1994. On the Newton polytope of the resultant. *Journal of Algebraic Combinatorics 3*, 207–236.

STURMFELS, B. 1998. Polynomial equations and convex polytopes. *Amer. Math. Monthly 105*, 10, 907–922.

SWELDENS, W. 1994. *The construction and application of wavelets in numerical analysis*. Ph. D. thesis, K.U.Leuven.

SYIEK, D. 1995. C vs Ada: Arguing performance religion. *ACM Ada Letters XV*, 6, 67–69.

TRAVERSO, C. 1993. The PoSSo test suite examples. Available at http://www.inria.fr/saga/POL/index.html.

VAN HENTENRYCK, P., MCALLESTER, D., AND KAPUR, D. 1997. Solving polynomial systems using a branch and prune approach. *SIAM J. Numerical Analysis 34*, 2, 797–827.

VERSCHELDE, J. 1990. Oplossen van stelsels veeltermvergelijkingen met behulp van continueringsmethodes. Bachelor's Thesis, K.U.Leuven.

VERSCHELDE, J. 1995. PHC and MVC: two programs for solving polynomial systems by homotopy continuation. In J. FAUGÈRE, J. MARCHAND, AND R. RIOBOO Eds., *Proceedings of the PoSSo Workshop on Software. Paris, March 1–4, 1995* (1995), pp. 165–175.

VERSCHELDE, J. 1996. *Homotopy Continuation Methods for Solving Polynomial Systems.* Ph. D. thesis, K.U.Leuven, Dept. of Computer Science.

VERSCHELDE, J. 1998. Numerical evidence for a conjecture in real algebraic geometry. Preprint #1998-064, MSRI. Paper and software available at the author's web-pages.

VERSCHELDE, J., BECKERS, M., AND HAEGEMANS, A. 1991. A new start system for solving deficient polynomial systems using continuation. *Appl. Math. Comput. 44*, 3, 225–239.

VERSCHELDE, J. AND COOLS, R. 1992. Nonlinear reduction for solving deficient polynomial systems by continuation methods. *Numer. Math. 63*, 2, 263–282.

VERSCHELDE, J. AND COOLS, R. 1993a. An Ada workbench for homotopy continuation for solving polynomial systems. *The Ada Belgium Newsletter 2*, 1, 23–40.

VERSCHELDE, J. AND COOLS, R. 1993b. Symbolic homotopy construction. *Applicable Algebra in Engineering, Communication and Computing 4*, 3, 169–183.

VERSCHELDE, J. AND COOLS, R. 1994. Symmetric homotopy construction. *J. Comput. Appl. Math. 50*, 575–592.

VERSCHELDE, J. AND COOLS, R. 1996. Polynomial homotopy continuation, a portable Ada software package. *The Ada-Belgium Newsletter 4*, 59–83. Proceedings of the 1996 Ada-Belgium Seminar, 22 November 1996, Eurocontrol, Brussels, Belgium.

VERSCHELDE, J. AND GATERMANN, K. 1995. Symmetric Newton polytopes for solving sparse polynomial systems. *Adv. Appl. Math. 16*, 1, 95–127.

VERSCHELDE, J., GATERMANN, K., AND COOLS, R. 1996. Mixed-volume computation by dynamic lifting applied to polynomial system solving. *Discrete Comput. Geom. 16*, 1, 69–112.

VERSCHELDE, J. AND HAEGEMANS, A. 1993. The *GBQ*-Algorithm for constructing start systems of homotopies for polynomial systems. *SIAM J. Numer. Anal. 30*, 2, 583–594.

VERSCHELDE, J., VERLINDEN, P., AND COOLS, R. 1994. Homotopies exploiting Newton polytopes for solving sparse polynomial systems. *SIAM J. Numer. Anal. 31*, 3, 915–930.

WALLACK, A., EMIRIS, I., AND MANOCHA, D. 1998. MARS: A Maple/Matlab/C resultant-based solver. In O. GLOOR Ed., *Proceedings of ISSAC-98* (Rostock, Germany, 1998), pp. 244–251. ACM.

WAMPLER, C. AND MORGAN, A. 1991. Solving the 6R inverse position problem using a generic-case solution methodology. *Mech. Mach. Theory 26*, 1, 91–106.

WAMPLER, C. W. 1992. Bezout number calculations for multi-homogeneous polynomial systems. *Appl. Math. Comput. 51*, 2–3, 143–157.

WAMPLER, C. W. 1996. Isotropic coordinates, circularity and Bezout numbers: planar kinematics from a new perspective. Proceedings of the 1996 ASME Design Engineering Technical Conference, Irvine, California August 18-22, 1996. CD-ROM edited by McCarthy, J.M., American society of mechanical engineers. Also available as GM Technical Report, Publication R&D-8188.

WATSON, L. T. 1986. Numerical linear algebra aspects of globally convergent homotopy methods. *SIAM Rev. 28*, 4, 529–545.

WATSON, L. T., BILLUPS, S. C., AND MORGAN, A. P. 1987. Algorithm 652: HOMPACK: a suite of codes for globally convergent homotopy algorithms. *ACM Trans. Math. Softw. 13*, 3, 281–310.

Watson, L. T., Sosonkina, M., Melville, R. C., Morgan, A. P., and Walker, H. F. 1997. HOMPACK90: A suite of Fortran 90 codes for globally convergent homotopy algorithms. *ACM Trans. Math. Softw. 23*, 4, 514–549. Available at http://www.cs.vt.edu/~ltw/.

Wise, S., Sommese, A., and Watson, L. 1998. POLSYS_PLP: A partitioned linear product homotopy code for solving polynomial systems of equations. Available at http://www.cs.vt.edu/~ltw/.

Wright, A. H. 1985. Finding all solutions to a system of polynomial equations. *Math. of Comp. 44*, 169, 125–133.

APPENDIX

A. GETTING STARTED

The purpose is to show the package at work on an extremely easy example: the intersection of an ellipse and a parabola. The root count equals four. On the left of Figure 6 is a degenerate configuration as start system to compute the common intersection points.

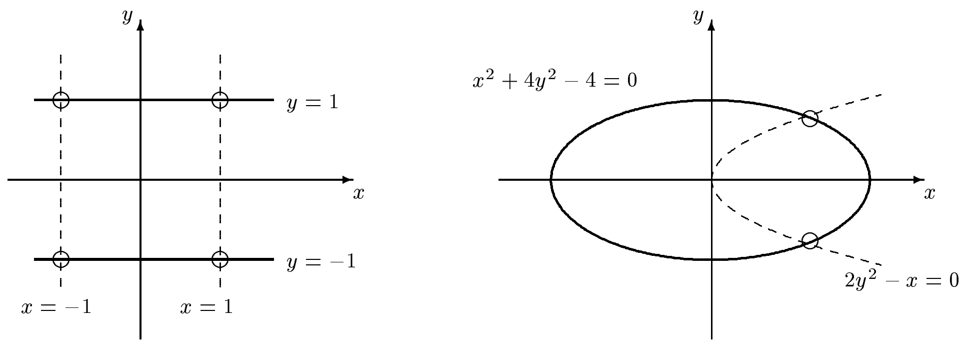


Fig. 6. Intersection of quadratics: a degenerate and a target configuration.

The classical convex-linear homotopy that connects target with start system is

$$H(x,y,t) = (1-t)\gamma \left(\underbrace{\overbrace{\left\{ \begin{array}{r} x^2 - 1 = 0 \\ y^2 - 1 = 0 \end{array} \right.}^{start}}_{F^{(0)}(x,y)=\mathbf{0}} \right) + t \left(\underbrace{\overbrace{\left\{ \begin{array}{r} x^2 + 4y^2 - 4 = 0 \\ 2y^2 - x = 0 \end{array} \right.}^{target}}_{F(x,y)=\mathbf{0}} \right), \quad t \in [0,1]. \tag{11}$$

By choosing a random complex number γ, we can reach all complex solutions even when all start solutions are real, see Figure 7. At $t = 0$ we have the degenerate configuration which is trivial to solve.
By following the solution paths defined by the homotopy we arrive for $t = 1$ at two real and two complex conjugated solutions of the target system. Note that in the x-component we have only two different paths.

It is recommended to put the algebraic formulation in a file to avoid typing it over many times. The file `mickey` of the demonstration database starts as follows

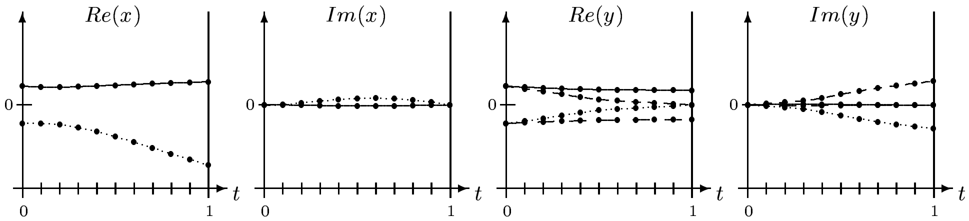


Fig. 7. The different real and imaginary parts of x and y along the solution paths.

```
2
 x**2 + 4*y**2 - 4;
        2*y**2 - x;
```

By typing `phc -b mickey /tmp/mickey.bphc` the black-box solver is invoked. The result can be seen in the file `/tmp/mickey.bphc`.

Next a session with phc in full mode is represented choosing default options.

```
your_machine_prompt: phc mickey /tmp/mickey.phc
Welcome to PHC (Polynomial Homotopy Continuation) Version 2.0.

Running in full mode.  Note also the following options:
  phc -s : Equation and variable Scaling on system and solutions
  phc -d : Linear and nonlinear Reduction w.r.t. the total degree
  phc -r : Root counting and Construction of start systems
  phc -m : Mixed-Volume Computation by four lifting strategies
  phc -p : Polynomial Continuation by a homotopy in one parameter
  phc -v : Validation, refinement and purification of solutions
  phc -e : SAGBI homotopies for a problem in enumerative geometry
  phc -b : Batch or black-box processing

MENU for Scaling Polynomial Systems :
  0 : No Scaling       : leave the menu
  1 : Equation Scaling : divide by average coefficient
  2 : Variable Scaling : change of variables, as z = (2^c)*x
Type 0, 1, or 2 to select scaling, or i for info : 0

MENU for Reducing Polynomial Systems :
  0 : No Reduction            : leave the menu
  1 : Linear Reduction        : triangulate coefficient matrix
  2 : Sparse Linear Reduction : diagonalize coefficient matrix
  3 : Nonlinear Reduction     : S-polynomial combinations
Type 0, 1, 2, or 3 to select reduction, or i for info : 0

MENU with ROOT COUNTS and Methods to Construct START SYSTEMS :
  0. exit - current start system is based on total degree : 4
PRODUCT HOMOTOPIES based on DEGREES ------------------------------
  1. multi-homogeneous Bezout number               (one partition)
  2. partitioned linear-product Bezout number    (many partitions)
```

```
  3. general linear-product Bezout number          (set structure)
  4. symmetric general linear-product Bezout number (group action)
POLYHEDRAL HOMOTOPIES based on NEWTON POLYTOPES ------------------
  5. combination between Bezout and BKK Bound   (implicit lifting)
  6. mixed-volume computation                     (static lifting)
  7. incremental mixed-volume computation        (dynamic lifting)
  8. symmetric mixed-volume computation        (symmetric lifting)
START SYSTEM DEFINED BY USER -----------------------------------
  9. you can give your own start system
Type a number between 0 and 9, eventually preceded by i for info : 0

Homotopy is H(x,t) = a*(1-t)^k * Q(x) + t^k * P(x) = 0, t in [0,1],
      with Q(x) = 0 a start system, and P(x) = 0 the target system.

MENU with settings for homotopy :
  relaxation constant k :  2
  regularity constant a :  9.95793808261866E-01   9.16225486838865E-02
  target value for t    :  1.00000000000000E+00   0.00000000000000E+00
  projective transformation : no
Type k, a, t, or p to change, preceded by i for info.  Type 0 to exit : 0

****************** CURRENT CONTINUATION PARAMETERS *****************
GLOBAL MONITOR :
  1. the condition of the homotopy          : 0
  2. number of paths tracked simultaneously  : 1
  3. maximum number of steps along a path    : 500
  4. distance from target to start end game  : 1.000E-01
  5. order of extrapolator in end game       : 0
  6. maximum number of re-runs               : 1
STEP CONTROL (PREDICTOR) :                    along path : end game
  7: 8. type ( x:Sec,t:Rea ):( x:Sec,t:Rea ) : 0         : 0
  9:10. minimum step size                    : 1.000E-06 : 1.000E-08
 11:12. maximum step size                    : 1.000E-01 : 5.000E-02
 13:14. reduction factor for step size       : 7.000E-01 : 5.000E-01
 15:16. expansion factor for step size       : 1.250E+00 : 1.100E+00
 17:18. expansion threshold                  : 1         : 1
PATH CLOSENESS (CORRECTOR) :                  along path : end game
 19:20. maximum number of iterations         : 4         : 4
 21:22. relative precision for residuals     : 1.000E-09 : 1.000E-11
 23:24. absolute precision for residuals     : 1.000E-09 : 1.000E-11
 25:26. relative precision for corrections   : 1.000E-09 : 1.000E-11
 27:28. absolute precision for corrections   : 1.000E-09 : 1.000E-11
SOLUTIONS (TOLERANCES):                       along path : end game
 29:30. inverse condition of Jacobian        : 1.000E-04 : 1.000E-12
 31:32. clustering of solutions              : 1.000E-04 : 1.000E-12
 33:34. solution at infinity                 : 1.000E+04 : 1.000E+12
*******************************************************************
```

```
Type a number to change (0 to exit), preceded by i for info : 0

MENU for Output Information during Continuation :
  0 : no intermediate output information during continuation
  1 : only the final solutions at the end of the paths
  2 : intermediate solutions at each step along the paths
  3 : information of the predictor: t and step length
  4 : information of the corrector: corrections and residuals
  5 : intermediate solutions and information of the predictor
  6 : intermediate solutions and information of the corrector
  7 : information of predictor and corrector
  8 : intermediate solutions, info of predictor and corrector
Type a number between 0 and 8 to select output information : 0

No more input expected.  See output file for results.

your_machine_prompt:
```

In full mode, the program runs through the scaling and reduction menu, shows the user all root-counting possibilities and, after the choice of start system, invokes the polynomial continuation.

B. REFERENCE MANUAL

B.1 Arguments and options in the command line

The general syntax for calling phc is as follows:

```
phc [-s|-d|-r|-m|-p|-v|-e] [-b] <input file> <output file>
```

As optional arguments the user can gives the names of the input and output file. The name of the input file must be put before the name of the output file. If only one file name is supplied, then it is assumed that it is the input file.

Following is a short description of the options and the acronyms of the tools invoked.

`-s` *Equation and variable Scaling on polynomial system and solution list*: `scal`.

`-d` *Linear and nonlinear Reduction w.r.t. the total degree of the system*: `redu`.

`-r` *Root counting and Construction of start systems*: `roco`.

`-m` *Mixed-Volume Computation by four different lifting strategies*: `mvc`.

`-p` *Polynomial Continuation defined by a homotopy in one parameter*: `poco`.

`-v` *Validation, refinement and purification of computed solution lists*: `vali`.

`-e` *Enumerative geometry for linear subspace intersections*: `enum`

`-b` *Batch or black-box processing.*

To facilitate the use of the tools, it is recommended to use the acronyms to define aliases so that these can be used from the command line. For instance, by creating an executable file `mvc` that contains simply the line

```
phc -m $1 $2
```

tool	input data	output results
scal	target system	scaled system
redu	target system	reduced system
roco	target system	a root count start system start solutions
mvc	target system	mixed volume mixed subdivision start system start solutions
poco	target system start system start solutions	solutions of target system
vali	target system solutions	refined solutions
enum	m-planes	p-planes

Table 10. Input and output of the tools.

the tool mvc can be invoked directly by typing `mvc input output` after the prompt.

In Table 10 the input data and results of the tools are given.

If phc is called without options, then the tools are activated one after the other, in the order as they are listed in Table 10.

B.2 Scaling polynomial systems and solutions

By scaling the coefficients are transformed so that they do not have extreme values. The purpose is to avoid numerical problems.

Equation Scaling means that every polynomial is divided by its average coefficient.

Variable Scaling uses transformations like $z = (2^c)x$. The transformation is such that real solutions remain real. The inverse of the condition number of the linear system that is solved to set up this transformation gives an indication on the condition of the original polynomial system.

Solution Scaling transforms the solutions of a scaled system back into the original coordinate system. Note that in the original coordinates, the solutions can be ill-conditioned.

B.3 Reduction of polynomial systems

The goal of reduction is to rewrite the system into an equivalent one (i.e.: with the same finite solutions) that has a lower total degree, so that fewer solution paths need to be followed. Sometimes reduction can already detect whether a system has no solutions or an infinite number of solutions.

Linear Reduction performs row-reduction on the coefficient matrix of the system.

Sparse Linear Reduction brings the coefficient matrix of the system in a diagonal format.

Nonlinear Reduction replaces polynomials by Subtraction-polynomials to eliminate highest-degree monomials. This type of reduction is more powerful, but also more expensive. Bounds have to be set to limit the combinatorial enumeration.

B.4 Root Counts and Start Systems

Two different root-counting models and start systems are available:

(1) Based on the highest degrees of the polynomials: random linear-product start systems;
(2) Based on the mixed volume of the Newton polytopes: random coefficient start systems.

The first class is well-suited for dense polynomial systems, whereas Newton polytopes provide a good model to capture the sparsity of a system.

The total degree is the product of the degrees of the polynomials in the system. The i-th equation of the start system is a univariate polynomial in the i-th unknown of the same degree as the i-th polynomial in the system that has to be solved. The total degree equals the number of solutions of the start system.

An m-homogeneous Bézout number is based on a partition of the set of unknowns, with m the number of sets in the partition. The corresponding start system is a linear-product system: the i-th equation is the product of linear equations with random coefficients in the unknowns of the set of the partition. The number of factors in the product of the i-th polynomial equals the product of the degrees of the i-th polynomial in the original system w.r.t. every set in the partition. Given a partition, the m-homogeneous Bézout number equals the number of solutions of the corresponding linear-product start system. Before the construction of the start system, an m-homogeneous Bézout number is first computed in a formal way as a generalized permanent of a degree matrix. Either all partitions of the set of unknowns can be evaluated, or the available heuristic procedure for generating a partition can be applied.

A partitioned linear-product Bézout number is based on a tuple of partitions of the set of unknowns. For every polynomial in the system, a different partition can model its structure. The corresponding start system is a linear-product system: the i-th equation is the product of linear equations with random coefficients in the unknowns of the set of the partition. The number of factors in the product for the i-th equation of the start system equals the product of the degrees of the i-th polynomial in the original system w.r.t. every set in the partition. Given a tuple of partitions, the partitioned linear-product Bézout number equals the number of solutions of the corresponding linear-product start system. Before the construction of the start system, a partitioned linear-product Bézout number is first computed in a formal way as a generalized permanent of a degree matrix. A heuristic procedure is available for generating a tuple of partitions.

A general linear-product Bézout number is based on a supporting set structure. A set structure is a tuple of arrays of subsets of unknowns. The corresponding start system is a linear-product system: the i-th equation is the product of linear equations with random coefficient in the unknowns of the set of the i-th array. The number of factors in the product for the i-th equation of the start system equals the number of subsets in the i-th array of the set structure. A set structure is supporting for a polynomial system if every monomial in the system also occurs in the corresponding linear-product start system. Given a supporting set structure, the general linear-product Bézout number equals the number of solutions of the corresponding linear-product start system. Before the construction of the start system, a general linear-product Bézout number is first computed in a formal way as a generalized permanent of a degree matrix. A heuristic procedure is available for generating a supporting set structure.

A symmetric general linear-product Bézout number is based on a symmetric supporting set structure and allows to exploit permutation symmetries in the system. The corresponding linear-product start system has the same symmetric structure, so that in the homotopy, only the generating solution paths need to be traced.

The mixed volume of the tuple of Newton polytopes gives an upper bound for the number of complex isolated solutions with nonzero coefficients. This BKK bound (named to honor Bernshtein, Koushnirenko and Khovanskii) is exact when the coefficients are sufficiently chosen at random or when the polytopes are in generic position w.r.t. each other. By implicit lifting, the mixed volume is computed by lifting up the origin of the first polytope and by computing those facets of the lower hull of the Minkowski sum that are spanned by edge-edge combinations. This procedure to compute the mixed volume uses recursion on the dimension. A random coefficient start system has the same Newton polytopes as the target system, but all coefficients are randomly chosen complex numbers. To solve a random coefficient start system, homotopy continuation is invoked in a recursive way, similar to the computation of the mixed volume. By constructing a set structure for the first k equations and computing the mixed volumes of the remaining $n - k$ equations obtained after elimination, a combined Bézout-BKK bound is computed.

Static lifting is a general procedure to construct regular mixed subdivisions of a tuple of polytopes. For mixed-volume computation, only those cells that are spanned by a tuple of edges contribute to the mixed volume. These cells are the so-called mixed cells in the subdivision. The collection of mixed cells is computed efficiently by pruning in the tree of lifted edge-edge combinations. These mixed cells provide the start systems in the polyhedral homotopy methods used to solve a random coefficient start system. Recursion is applied in case the lifting does not induce at once a fine mixed subdivision.

Dynamic lifting can be used to compute mixed volumes incrementally, i.e.: by adding the points repeatedly to the already constructed subdivision. This method works efficiently when all Newton polytopes are (almost) equal. The Cayley trick is implemented by means of dynamic lifting. This trick computes all cells in a mixed subdivision.

Symmetric lifting allows to exploit permutation symmetries in the tuple of Newton polytopes. A symmetric subdivision is induced by lifting the points in the same orbit up to the same height. The corresponding random coefficient start system has the same symmetric structure, so that in the homotopy, only the generating solution paths need to be traced.

B.5 The artificial-parameter homotopy

The artificial-parameter homotopy has the following form:

$$H(\mathbf{x}, t) = a(1-t)^k Q(\mathbf{x}) + t^k P(\mathbf{x}), \quad t \in [0, 1],$$

with $Q(\mathbf{x}) = \mathbf{0}$ a start system, and $P(\mathbf{x}) = \mathbf{0}$ the target system.

The homotopy parameter k determines the power of the continuation parameter t. Taking $k > 1$ has as effect that at the beginning and at the end of the continuation, changes in t do not change the homotopy as much as with a homotopy that is linear in t so that paths are better to follow. The default value $k = 2$ is recommended.

The homotopy parameter a ensures the accessibility and regularity of the solution paths, i.e.: by choosing a random complex number for a, all paths are regular and do not diverge for $t < 1$.

The target value for the continuation parameter t is by default 1. To create stepping stones in the continuation stage, it is possible to let the continuation stop at $t < 1$, for instance at $t = 0.9$ or even at a complex value for t. The solutions at $t < 1$ will serve at start solutions to take up the continuation in a later stage. In this stage, the same homotopy parameters k and a must be used.

A projective transformation of the homotopy and start solutions makes the equations in the polynomials homogeneous and adds a random hyperplane. The vectors of the start solutions are extended with an additional unknown. For solutions at infinity, this additional unknown equals zero.

B.6 Polynomial Continuation

Polynomial continuation is the computational engine of the solver. The dashboard consists of four blocks of parameters. The user can interactively tune these parameters.

B.6.1 *Monitoring the path tracker*

the condition of the homotopy: This global parameter can be used to tune the parameters according to the general expectancy of how difficult the solution paths will be. Low values of this parameter lead to a loose path tracking, whereas higher values tighten up the path-following.

number of paths tracked simultaneously: In simultaneous path-following, the same discretization for the values of the continuation parameter is used for a number of paths. Clustering of solution paths is checked during continuation for the number of paths that are followed simultaneously. In sequential path-following, i.e., when this parameter equals one, the step sizes are adjusted according to the difficulty of the current path being followed. In this case, a difficult path does not slow down the continuation for the other ones.

maximum number of steps along a path: This parameter bounds the amount of work along a solution path. Continuation for the path being followed stops when the number of steps exceeds this threshold value.

distance from target to start end game: Tracking solution paths is organized in two stages. As $t < 1$, no singularities are likely to occur and paths can be tracked more loosely. When $t \approx 1$, paths may converge to singular solutions or diverge to infinity, so they have to be tracked more carefully. This distance marks the switch towards the second stage, that is the so-called end game.

order of extrapolator in end game: The direction of a diverging solution path provides a normal to a face of the system that certificates the deficiency. In this polyhedral end game, extrapolation is used to determine winding numbers. When the order of the extrapolator equals zero, then the polyhedral end game is turned off.

maximum number of re-runs: Solution paths may be clustered at the end of the continuation. If the number of re-runs is higher than one, then clustered solution paths are re-computed by following the paths more closely. Setting this value to zero turns off this facility.

B.6.2 *Adaptive step-size control*

predictor type: We distinguish between predictors for the solution vector $\mathbf{x}$ and the continuation parameter t. The secant predictor for $\mathbf{x}$ is based on linear extrapolation, whereas the tangent predictor uses the Jacobian matrix to extrapolate. For t, three different predictors are provided. The real predictor simply augments t with the step size. The complex predictor takes values in the complex plane to circumvent values of t for which the corresponding system is badly conditioned. The geometric predictor generates values for t so that the distances to the target value behave like a geometric series. The latter predictor is useful in a polyhedral end game.

minimum step size: This is a lower bound on the step size for the continuation parameter. Continuation for the path being followed stops when a required reduction of the step size yields a step length that is lower than the minimum step size.

maximum step size: This is an upper bound on the step size for the continuation parameter. An expansion of the step size that yields a step length that is larger than this threshold sets the step size to this threshold.

reduction factor for step size: When the corrector does not manage to reach the required accuracy within the allowed number of iterations, the step size will be reduced by multiplication with the reduction factor.

expansion factor for step size: When the corrector has reached the required accuracy within the allowed number of iterations, for a number of consecutive times higher than the expansion threshold, the step size will be enlarged by multiplication by the expansion factor.

expansion threshold: Before enlarging the step size along a path, the number of consecutive successful steps must pass the expansion threshold. A step is successful when the corrector manages to reach the required accuracy within the allowed number of iterations. At the end of the path, this threshold bounds the number of consecutive equal guesses of the winding number m before the polyhedral end game can modify it.

B.6.3 *Path closeness*

maximum number of iterations: The corrector stops when the desired accuracy is reached or when it has exhausted its maximum number of iterations. A low maximum enforces quadratic convergence and keeps the path tracker close to the solution paths. A higher maximum may be needed at the end of the solution path, when quadratic convergence can no longer be obtained due to singularities.

relative precision for residuals: The residual is the norm of the vector obtained after evaluating the current approximation vector in the polynomial system. The corrector stops when the residual divided by the norm of the approximation vector is lower than or equal to the required precision or when another required precision is attained.

absolute precision for residuals: The residual is the norm of the vector obtained after evaluating the current approximation vector in the polynomial system. The corrector stops when the residual is lower than or equal to the required precision or when another required precision is attained.

relative precision for corrections: The correction is the norm of the last vector used to update the current approximation vector. The corrector stops when the correction divided by the norm of the approximation vector is lower than or equal to the required precision or when another required precision is attained.

absolute precision for corrections: The correction is the norm of the last vector used to update the current approximation vector. The corrector stops when the correction is lower than or equal to the required precision or when another required precision is attained.

B.6.4 *Solution tolerances*

inverse condition of Jacobian: A solution is considered as singular when the inverse of the condition number of the Jacobian matrix is lower than the given threshold value. Otherwise, the solution is declared to be regular.

clustering of solutions: Two solutions are considered as clustered when the distance between all corresponding components is lower than the given threshold value.

solution at infinity: A solution is considered to diverge to infinity when its norm exceeds the given threshold value, in case of affine coordinates, or, in case of projective coordinates, when the added coordinate becomes lower than the inverse of the threshold value. Continuation for the path being followed stops when it diverges to infinity.

B.7 Validation

Basic validation consists in the application of Newton's method on the list of solutions. There are facilities to extract the generating solutions when the symmetry group is submitted.

Winding numbers can be computed by homotopy continuation methods. The user must provide a start system with solutions at $t < 1$.

Polyhedral validation is based on the output file of poco, where the polyhedral end game was turned on. This validation puts up a frequency table of computed path directions.

B.8 Enumerative Geometry

A p-plane in $\mathbb{C}^{m+p}$ is represented by an $(m+p)$-by-p matrix that has in its columns the generators of this plane. For numerical stability, we better apply the QR-algorithm to generate an orthogonal representation.

A localization pattern is used to specify the position of the zeros in the representation of a p-plane.

B.9 Input/output formats

Below the two most common formats of the program are described.

B.9.1 *Formats of polynomial systems*

A complex polynomial system is denoted by the dimension followed by as many complex multivariate polynomials as the dimension. The dimension is a positive natural number.

A complex multivariate polynomial is denoted as a sequence of terms, separated by `+` and terminated by the semicolon `;`. The brackets `(` and `)` must be used to isolate a sequence of terms as a factor in a complex multivariate polynomial.

A term can be either a coefficient or a coefficient, followed by `*` and a monomial. If in the latter case the coefficient equals one, then it may be omitted.

A coefficient may be denoted as an integer, a rational, a floating-point or a complex number.

A monomial is a sequence of powers of unknowns, separated by `*`. The power operator is represented by `**` or `^`. It must be followed by a positive natural number. If the power equals one, then it may be omitted.

An unknown can be denoted by at most five characters. The first character must be a letter and the other two characters must be different from `+`, `-`, `*`, `^`, `/`, `;`, `(` and `)`. The letter i means $\sqrt{-1}$, whence it does not represent an unknown. The number of unknowns may not exceed the declared dimension.

Some examples of valid notations of complex multivariate polynomials:

```
x**2*y + 1/2*z*y**2 - 2*z + y**3 + x - 1E9/-8.E-6* y + 3;
x^2*y + z*y^2 - 2*z + y^3 + x - y + 3;
(1.01 + 2.8*i)*x1**2*x2 + x3**2*x1 - 3*x1 + 2*x2*x3 - 3;
(x1^2*x2 + x3^2*x1 - 3*x1 + 2*x2*x3 - 3)*x2**2*(x2-1+i);
```

Some notes concerning the internal representation:

—Graded lexicographical ordering is used for the monomials.

—The internal order of the unknowns is determined by the order in which they occur in the polynomial system. For example, if `x2` is read before `x1`, then the powers of x2 will come first in the representation of the support sets and mixed subdivision.

—The continuation parameter is represented by t. Although t is not a reserved symbol, it is better not to use it to represent unknowns, to avoid confusion.

An input file to phc must begin with a polynomial system in the appropriate format. As example, consider the intersection of a circle with a parabola:

```
2
 x**2 + 4*y**2 - 4;
        2*y**2 - x;
```

B.9.2 *Formats of solutions*

A solution list of a complex polynomial system is denoted by the number of solutions and the dimension, followed by, for each solution, its position in the list, and the solution itself. The solutions are separated by a banner line.

A solution consists of the current value of the continuation parameter t, its multiplicity (or winding number) m, and the solution vector.

A solution vector contains as many lines as the dimension. The i-th line starts with the symbol that represents the i-th unknown, followed by the colon ':' and two floating-point numbers representing respectively the real and imaginary part of the solution component.

As example we list the solution list of the regular solution $(1, i)$ of a 2-dimensional system in the unknowns x and y at $t = 1$.

```
1 2
===========================================================
solution 1 :
t :  1.00000000000000E+00  0.00000000000000E+00
m : 1
the solution for t :
 x : 1.00000000000000E+00  0.00000000000000E+00
 y : 0.00000000000000E+00  1.00000000000000E+00
===========================================================
```

The separation banner between two solutions may contain additional information on output. The following is an example of a last line of a list of solutions as produced by the path-following routines:

```
== #regu : 4 = #sing : 0 = #clus : 0 = #infi : 0 = #fail : 0 ==
```

This means that four paths have yielded four regular solutions. There were no singular and no clustered solutions, no solutions at infinity, nor path failures.

Newton's method used in the root refiner produces separation banners like

```
== err :  9.930E-17 = rco :  1.998E-01 = res :  4.441E-16 = real regular ==
```

to indicate that the norm of the last correction vector of Newton method is $9.930\ 10^{-17}$, that the inverse of the condition number of the Jacobian matrix equals $1.998\ 10^{-1}$, and that the residual equals $4.441\ 10^{-16}$. By its condition number, the solution is regular and, since the imaginary parts of the solution components are sufficiently small, the solution is reported to be real. Separation banners have no meaning to the program when the solution lists are submitted as input.

When a start system is submitted to the program, the list of solutions is expected to be after a banner that contains the word SOLUTIONS.

Index