# Interfacing with the Numerical Homotopy Algorithms in *PHCpack*⋆

Anton Leykin[1] and Jan Verschelde[2]

Department of Mathematics, Statistics, and Computer Science
University of Illinois at Chicago, 851 South Morgan (M/C 249)
Chicago, IL 60607-7045, USA.
[1]leykin@math.uic.edu, http://www.math.uic.edu/~leykin
[2]jan@math.uic.edu, http://www.math.uic.edu/~jan

**Abstract.** *PHCpack* implements numerical algorithms for solving polynomial systems using homotopy continuation methods. In this paper we describe two types of interfaces to *PHCpack*. The first interface *PHCmaple* originally follows OpenXM, in the sense that the program (in our case Maple) that uses *PHCpack* needs only the executable version `phc` built by the package *PHCpack*. Following the recent development of *PHCpack*, *PHCmaple* has been extended with functions that deal with singular polynomial systems, in particular, the deflation procedures that guarantee the ability to refine approximations to an isolated solution even if it is multiple. The second interface to *PHCpack* was developed in conjunction with MPI (Message Passing Interface), needed to run the path trackers on parallel machines. This interface gives access to the functionality of *PHCpack* as a conventional software library.

## 1 Introduction

In various fields of science and engineering one must solve polynomial systems, see for example [14], or the case studies in [22, Chapter 9], or [24]. Computer algebra packages like Maple have a convenient worksheet interface which allows to document very precisely the derivation of the polynomial systems using the language of the application area. We designed *PHCmaple* [5] (based on a small Maple procedure in [20], applying our experience with OpenXM [11]) to give a Maple user access to the functionality of a numerical polynomial system solver `phc` (**p**olynomial **h**omotopy **c**ontinuation), a program built by *PHCpack* [25].

Besides a carefully documented problem formulation, the user will need to analyze and interpret the results returned by the solver. The visualization capabilities of a computer algebra system, combined with high level facilities to manipulate and export data in various formats, extend the usefulness of the solver. So the interaction with computer algebra is not only seen as a natural, but as a vital part of the solving process.

⋆ This material is based upon work supported by the National Science Foundation under Grant No. 0134611 and Grant No. 0410036. *Date:* 22 June 2006.

The interaction between computer algebra systems and numerical solvers is one practical side of symbolic-numeric computation, concerned with the implementation of hybrid methods [3]. Examples of hybrid methods are the algorithms developed in the fields of numerical polynomial algebra [23] and numerical algebraic geometry [21] [22]. The algorithms at the heart of numerical algebraic geometry are homotopy continuation methods (see e.g. [1], [10], or [14]), which can be seen as the combination of two methods. Homotopy methods embed the system to be solved in a suitable family of systems, connecting the given problem with a system which is trivial or at least easier to solve. Once this family (called the homotopy) is created, numerical path following methods (also known as continuation methods) track solution paths starting at solutions of the easier problem leading to the solutions of the given problem. As the number crunching is usually all done in hardware floating point arithmetic, the performance of the numerical solver depends critically on the ability of the symbolic homotopy method to capture the structure of the given problem.

Originally designed [25] to offer a wide variety of homotopy algorithms, *PHCpack* has grown into a platform for numerical algebraic geometry, most notably extended with features [20] to deal with positive dimensional solution sets of polynomial systems, implementing a so-called numerical irreducible decomposition [19]. While the *pack* in its name suggests *PHCpack* to be in the glorious tradition of highly successful software like LAPACK [2], its main target is the standalone executable program `phc` which currently is available on a wide range of platforms: PCs running Linux and Windows, computers running MacOS X, SUN workstations running Solaris and IBM machines running AIX. The recent "parallel PHCpack" project is described in [7].

The Maple interface *PHCmaple* was used in [13]. In [16], Maple and *PHCpack* were combined to develop a prototype implementation of new algorithms in numerical jet geometry. In addition to problem formulation and result analysis, this algorithm prototyping is our third motivation for interfaces like *PHCmaple*.

In this paper we present an overview of *PHCmaple*, we refer to [5] for details on its original design and to [9] for the added interface to the new deflation algorithms [8] to recondition multiple isolated roots of polynomial systems. While *PHCmaple* is a user oriented interface, in this paper we document an alternative interface, developed for programming purposes, in particular for use with MPI [17], for tracking solution paths on parallel machines.

The programmer interface to *PHCpack* is characterized by two important features inspired by the *PHCmaple* user interface. Like *PHCmaple* relies only on the executable `phc`, the programmer interface is concentrated in one single routine. Moreover, as the user of *PHCmaple* enjoys the existing data manipulation facilities of Maple, the internal data structures of *PHCpack* for representing polynomials and solutions to systems, are available to the user of the programmer interface so the programmer calling on *PHCpack* should not define similar data structures. In this sense, the programmer interface to *PHCpack* resembles very much a conventional software library.

## 2 The Evolution of the Interfaces to *PHCpack*

In this section we briefly describe the chronological evolution of the interfaces to the capabilities in *PHCpack*.

1. **OpenXM calls the blackbox solver.** The first interface is still available via OpenXM (Open message eXchange protocol for Mathematics [11], see also [12] and [15]) and only needs an executable file of the program. Just as one calls the blackbox solver of *PHCpack* as `phc -b input output`, a simple system call achieves the same effect.
2. **A simple Maple procedure calls the blackbox solver.** On [20, page 114], a simple Maple 7 procedure (less than 20 lines long) applies the experience of the first interface, calling `phc -b`.
3. **A functional C interface to the Ada routines in *PHCpack*.** To process the output of the Pieri homotopies in *PHCpack* to compute feedback laws to control a linear system, a dedicated C interface was written, used in [26] and described in an online appendix (available at the second author's web site). The main C program calls the Ada routines in *PHCpack* which then call another C function to process the output.
4. ***PHCmaple* gives access to the tools of `phc`.** The main executable program of *PHCpack* can be used as a blackbox or as a toolbox, calling the program with the appropriate options and selecting the desired actions from the menu. Via input redirections, it also just takes the executable version to gain access to the tools offered by *PHCpack*. In [5], we presented *PHCmaple*, a Maple interface to *PHCpack*.
5. **Using *PHCpack* as a state machine.** The dedicated C interface was not adequate for the parallel implementation of the path tracking routines in *PHCpack*. Therefore, a new interface was developed for use in [27], [4] [6] and [28], which describe the progress by parallel *PHCpack* [7].
   The Ada function `use_c2phc`

   ```
   function use_c2phc ( job : integer;
                        a : C_intarrs.Pointer;
                        b : C_intarrs.Pointer;
                        c : C_dblarrs.Pointer ) return integer;
   ```

   is available to the C programmer as

   ```
   extern void adainit( void );
   extern int _ada_use_c2phc ( int job, int *a, int *b, double *c );
   extern void adafinal( void );
   ```

   The first parameter `job` specifies the action requested from `phc`. The meaning of the other parameters `a`, `b`, and `c` depends on the `job`. We will describe this interface in greater detail below.

The chronological evolution pictures two distinct trends in interfacing with *PHCpack*: (1) using the executable originated with OpenXM [11]; and (2) calling the compiled code, motivated by the use of MPI [17].

## 3 Overview of *PHCmaple*

*PHCmaple* is available for download at `www.math.uic.edu/~leykin/PHCmaple/` and works with Maple version 8 or higher. At the moment the software is developed and tested only on the Windows distribution of Maple.

The goal of *PHCmaple* is to provide computer algebra users with a convenient interface to the

- blackbox solver of `phc`;
- homotopy path tracking facilities;
- deflation procedure;
- routines that create and manipulate witness sets for positive-dimensional components;
- factorization/decomposition capabilities of `phc`.

Using the following procedures one can deal with *isolated solutions* of square systems (with a finite number of solutions): to run the black-box solver execute **solve**, which returns approximations to all complex isolated roots of a square system; refines the solutions to any specified precision with **refine**, which also provides a way to set certain parameters in order to fine-tune the solver; **track** a subset of the solutions set of the start system to the corresponding solutions of the target system and visualize the results with **drawPaths**; for singular isolated solutions one might consider applying **deflationStep**, the implementation of the first-order *deflation* procedure [8], which given a polynomial system and an approximation to one of its multiple isolated solutions produces a new system of equations that has the same solution, but with lower multiplicity. See [9] for more details about the implementation of the deflation algorithm and examples for its use by *PHCmaple*.

The positive-dimensional solution sets of general polynomial systems can be represented by means of *witness sets*, computing which reduces the problem to the isolated solution case.

The following functions of *PHCmaple* serve this purpose: construct an embedded system with **embed** in assumption that the dimension of its solution set is known; **cascade** runs the so-called *cascade of homotopies* for an embedded system, it computes the list of *witness sets* for the components of the solution set in every dimension; after producing the witness sets **filter** the points in lower-dimensional witness sets belonging to higher-dimensional components; to produce a *numeric irreducible decomposition* of a pure-dimensional solution component **decompose** its witness set; absolute factorization capability for multivariate polynomial is given by **factor**.

The new, recently added **eqnbyeqn** routine launches the equation-by-equation solver [18], which produces the witness sets similarly to **cascade**, though using a different method. These witness sets are returned already filtered.

## 4 Using *PHCpack* in C programs

The function `use_c2phc` serves as a gateway to the full functionality of *PHCpack*, concentrated in one single routine. When designing this interface, we viewed *PHCpack* as a state machine. Like we input coins into a vending machine, make a selection pushing a button to then collect our selected beverage, the programmer calls `use_c2phc` with the appropriate `job` number to read in a polynomial system, select the type of homotopy and various other options to then finally activate the path trackers which will compute the solutions and write to the output.

The `use_c2phc` was written to get access to the path tracking routines (written in Ada) by a main program in C, which calls the communication primitives of MPI. It was used in a series of papers, starting with [27] (parallel Pieri homotopies), continuing with [6] and [4] (parallel factorization), and most recently used in [28] (parallel polyhedral homotopies). As `use_c2phc` called on various homotopy algorithms in *PHCpack*, the number of different `job` numbers grew and the direct use of the gateway by the main parallel program became too tedious. So we developed an additional layer between the main parallel program and the `use_c2phc` function.

This additional layer forms the programmer interface to *PHCpack*. It consists of a collection of header files (suffix `.h`) offering the programmer meaningful names for the various jobs performed by `use_c2phc`, hiding the precise `job` number in the definition of the function listed in the header files. Details about this evolving interface can be found in the source code distribution of *PHCpack*.

## 5 Future Developments

In this paper we described a user and a programmer interface to *PHCpack*.

While currently *PHCmaple* still operates by making system calls to the standalone program `phc`, a more efficient interface will apply the `use_c2phc` in a dynamic link library.

Besides adding more functionality to `use_c2phc`, a very useful extension of its argument list will be the addition of two functions to allow the user to define so-called straight line programs to evaluate and differentiate the polynomial systems generated by the homotopy.

The interfaces we described will serve as a model to use *PHCpack* in other computer algebra systems like Axiom or Macaulay 2 for example, as well as in scientific computing systems like Octave or Scilab. As C seems to be the least common denominator language for computer programming, the programmer interface could lead to bindings to other languages or used to build other dedicated interfaces.

## References


1. E.L. Allgower and K. Georg. *Introduction to Numerical Continuation Methods*, volume 45 of *Classics in Applied Mathematics*. SIAM, 2003.
2. E. Anderson, Z. Bai, C. Bischof, L.S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammerling, A. McKenny, and D. Sorensen. *LAPACK User's Guide*. SIAM, 3rd edition, 1999. Available online via `http://www.netlig.org/lapack.`
3. R.M. Corless, E. Kaltofen, and S.M. Watt. Hybrid methods. In J. Grabmeier, E. Kaltofen, and V. Weispfenning, editors, *Computer Algebra Handbook*, pages 112–125. Springer–Verlag, 2002.
4. A. Leykin and J. Verschelde. Decomposing solution sets of polynomial systems: a new parallel monodromy breakup algorithm. Accepted for publication in *The International Journal of Computational Science and Engineering*.
5. A. Leykin and J. Verschelde. PHCmaple: A Maple interface to the numerical homotopy algorithms in PHCpack. In Quoc-Nam Tran, editor, *Proceedings of the Tenth International Conference on Applications of Computer Algebra (ACA'2004)*, pages 139–147, 2004.
6. A. Leykin and J. Verschelde. Factoring solution sets of polynomial systems in parallel. In Tor Skeie and Chu-Sing Yang, editors, *Proceedings of the 2005 International Conference on Parallel Processing Workshops. 14-17 June 2005. Oslo, Norway. High Performance Scientific and Engineering Computing*, pages 173–180. IEEE Computer Society, 2005.
7. A. Leykin, J. Verschelde, and Zhuang Y. Parallel homotopy algorithms to solve polynomial systems. Proceedings of ICMS'06, this volume.
8. A. Leykin, J. Verschelde, and A. Zhao. Newton's method with deflation for isolated singularities of polynomial systems. To appear in *Theoretical Computer Science*.
9. A. Leykin, J. Verschelde, and A. Zhao. Evaluation of Jacobian matrices for Newton's method with deflation to approximate isolated singular solutions of polynomial systems. In D. Wang and L. Zhi, editors, *SNC 2005 Proceedings. International Workshop on Symbolic-Numeric Computation. Xi'an, China, July 19-21, 2005*, pages 19–28, 2005.
10. T.Y. Li. Numerical solution of polynomial systems by homotopy continuation methods. In F. Cucker, editor, *Handbook of Numerical Analysis. Volume XI. Special Volume: Foundations of Computational Mathematics*, pages 209–304. North-Holland, 2003.
11. M. Maekawa, M. Noro, K. Ohara, Y. Okutani, N. Takayama, and Tamura Y. Openxm – an open system to integrate mathematical softwares. Available at `http://www.OpenXM.org/`.
12. M. Maekawa, M. Noro, K. Ohara, N. Takayama, and Y. Tamura. The design and implementation of OpenXM-RFC 100 and 101. In K. Shirayanagi and K. Yokoyama, editors, *Computer mathematics. Proceedings of the Fifth Asian Symposium (ASCM 2001) Matsuyama, Japan 26 - 28 September 2001*, volume 9 of *Lecture Notes Series on Computing*, pages 102–111. World Scientific, 2001. Available at `http://www.math.kobe-u.ac.jp/OpenXM/ascm2001/ascm2001/ascm2001.html.`
13. M.M. Maza, G.J. Reid, R. Scott, and W. Wu. On approximate triangular decomposition I. Dimension zero. In D. Wang and L. Zhi, editors, *SNC 2005 Proceedings. International Workshop on Symbolic-Numeric Computation. Xi'an, China, July 19-21, 2005*, pages 19–28, 2005.

14. A. Morgan. *Solving polynomial systems using continuation for engineering and scientific problems*. Prentice-Hall, 1987.
15. M. Noro. A computer algebra system: Risa/Asir. In M. Joswig and N. Takayama, editors, *Algebra, Geometry, and Software Systems*, pages 147–162. Springer–Verlag, 2003.
16. G. Reid, J. Verschelde, A. Wittkopf, and W. Wu. Symbolic-numeric completion of differential systems by homotopy continuation. In M. Kauers, editor, *Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation (ISSAC'05), July 24-27 2005, Beijing, China*, pages 269–276. ACM, 2005.
17. M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra. *MPI - The Complete Reference Volume 1, The MPI Core*. Massachusetts Institute of Technology, 2nd edition, 1998. Available via `http://www-unix.mcs.anl.gov/mpi/`.
18. A.J. Sommese, J. Verschelde, and C.W. Wampler. Solving polynomial systems equation by equation. Submitted for publication.
19. A.J. Sommese, J. Verschelde, and C.W. Wampler. Numerical decomposition of the solution sets of polynomial systems into irreducible components. *SIAM J. Numer. Anal.*, 38(6):2022–2046, 2001.
20. A.J. Sommese, J. Verschelde, and C.W. Wampler. Numerical irreducible decomposition using PHCpack. In M. Joswig and N. Takayama, editors, *Algebra, Geometry, and Software Systems*, pages 109–130. Springer–Verlag, 2003.
21. A.J. Sommese, J. Verschelde, and C.W. Wampler. Introduction to numerical algebraic geometry. In *Solving Polynomial Equations. Foundations, Algorithms and Applications*, volume 14 of *Algorithms and Computation in Mathematics*, pages 301–337. Springer–Verlag, 2005.
22. A.J. Sommese and C.W. Wampler. *The Numerical solution of systems of polynomials arising in engineering and science*. World Scientific Press, Singapore, 2005.
23. H.J. Stetter. *Numerical Polynomial Algebra*. SIAM, 2004.
24. B. Sturmfels. *Solving Systems of Polynomial Equations*. Number 97 in CBMS Regional Conference Series in Mathematics. AMS, 2002.
25. J. Verschelde. Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Trans. Math. Softw.*, 25(2):251–276, 1999. Software available at `http://www.math.uic.edu/~jan`.
26. J. Verschelde and Y. Wang. Computing dynamic output feedback laws. *IEEE Transactions on Automatic Control*, 49(8):1393–1397, 2004.
27. J. Verschelde and Y. Wang. Computing feedback laws for linear systems with a parallel Pieri homotopy. In Y. Yang, editor, *Proceedings of the 2004 International Conference on Parallel Processing Workshops, 15-18 August 2004, Montreal, Quebec, Canada. High Performance Scientific and Engineering Computing*, pages 222–229. IEEE Computer Society, 2004.
28. J. Verschelde and Y. Zhuang. Parallel implementation of the polyhedral homotopy method. Accepted for publication in the proceedings of *The 8th Workshop on High Performance Scientific and Engineering Computing (HPSEC-06)*, Columbus, Ohio, USA, August 18, 2006.