

A Short Introduction to `phc`

Reinhard Steffens (steffens@math.uni-frankfurt.de)
<http://www.math.uni-frankfurt.de/~steffens/phc/>

Contents

1	Getting started	2
1.1	What is it good for?	2
1.2	Where can I get it?	2
1.3	Where can I get help?	2
2	File Format	3
2.1	Input	3
2.2	Output	5
3	Different Modes of Usage	6
3.1	The Black-Box Solver	6
3.2	The Tool-Box Mode	7
4	A Short Description of the Solver	7
4.1	The 4 Stages of the Solver	7
4.2	Root Counts and Start Systems	8
4.3	Polynomial Continuation and End Games	9
4.4	How to Detect Higher Dimensional Solution Sets	9
5	Maple, MatLab and C interfaces	10
6	Examples and Exercises	10

1 Getting started

1.1 What is it good for?

Given a system of n polynomial equations over \mathbb{C} in n unknowns with a finite number of common solutions `phc` can

- count the number of solutions by various methods
- compute approximations to all isolated solutions.

If the solution set is higher dimensional `phc` can perform a numerically irreducible decomposition of the variety (see [6]).

1.2 Where can I get it?

- Download: <http://www.math.uic.edu/~jan/download.html>
- Installation: First unzip and unpack the downloaded file ('gunzip (filename)' and 'tar -xf (filename)'). The program doesn't need any installation or configuration and should work right now. Go to the directory where it is located and type './phc'. For a more comfortable usage I recommend linking `phc` in your '/bin' directory or include the `phc` directory in your search path.

1.3 Where can I get help?

The homepage of the programs' author Jan Verschelde

<http://www.math.uic.edu/~jan/>

contains lots of informational material of which only some are listed here.

- The reference manual: www.math.uic.edu/~jan/PHCpack/phcpack.html

- Course material from a tutorial given by the author (You find that in the download section).
- Lots of example files (Go to the homepage and click on 'Demos' in the left column).
- See also the articles [7], [4] and [6] for detailed information.

2 File Format

2.1 Input

Each input file starts with two natural numbers denoting the number of equations and the number of variables. (If these two numbers are the same, which should be the common case, the second one may be omitted.) This is followed by the polynomials of your initial system separated by ';' . E.g. to find the intersection of the circle $x^2 + 4y^2 - 4 = 0$ with the parabola $2y^2 - x = 0$ we write:

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

Note that

- The exponentiation '^' may be written as '**'
- Don't write '= 0'
- Variable names should start with a letter and contain up to 5 characters. (Please don't use '* + - E I' etc. .)
- An E is used to denote the exponent of a floating point number. E.g. 0.015 is written as 1.5E-2.
- The complex unit is denoted by 'I'.

- The internal order of the unknowns is determined by the order in which they occur in the polynomial system. So if x_2 occurs before x_1 you might get confused by the output of the support systems or the mixed subdivisions. To avoid this you can start your polynomial system with a redundant equation like `'x1-x1+x2-x2+x3-x3...;'`.
- Internally graded lexicographical ordering is used for the monomials.

Now you can run `phc` in full mode by typing

```
phc (input file) (output file)
```

If you don't have your system stored in a separate file or if `phc` has problems reading it (this might happen because this program is very picky concerning inputs) you can enter the system by hand while the program runs.

If you just want to use some special tool out of the `phc`-pack or if you want to use the black-box solver you can type one of the following commands:

```
phc -0 : random numbers with zero seed for repeatable runs
phc -a : Solving polynomial systems equation-by-equation
phc -b : Batch or black-box processing
phc -c : Irreducible decomposition for solution components
phc -d : Linear and nonlinear Reduction w.r.t. the total degree
phc -e : SAGBI/Pieri homotopies to intersect linear subspaces
phc -f : Factor pure dimensional solution set into irreducibles
phc -k : realization of dynamic output feedback placing poles
phc -l : Witness Set for Hypersurface cutting with Random Line
phc -m : Mixed-Volume Computation by four lifting strategies
phc -p : Polynomial Continuation by a homotopy in one parameter
phc -q : Tracking Solution Paths with incremental read/write
phc -r : Root counting and Construction of start systems
phc -s : Equation and variable Scaling on system and solutions
phc -v : Validation, refinement and purification of solutions
phc -w : Witness Set Intersection using Diagonal Homotopies
phc -z : strip phc output solution lists into Maple format
```

2.2 Output

The output is rather long and depends on the mode you run `phc`. So here we will only explain the basics that will help you understand what you read. For a quick start read the two remarks at the end of this section.

Initial system:

In the first lines of the output file you will find your initial system of polynomial equations.

Root counting:

The following paragraphs describe the performed root counting. Of course this depends on which methods you chose for that. The description of the root counting is quite complete. E.g. if you compute the mixed volume you will find the lifting vectors used and all mixed cells etc.. Note that if the program runs into problems and has to perform some operation twice or more you will find the results of every approach and not just the successful one in the output file. After that you find some timing information of the root counting.

Start system:

If you perform polynomial continuation you will find all the information concerning the start system in the following paragraphs. (This information is usually stored in a separate file as well so you can use it for further computations.) The solutions of the start system are written in the same format as the solutions to your initial system which can be found later in the output-file. DON'T CONFUSE these two sets of solutions. Again you find the information about the time it took to compute the start system.

Parameters for continuation:

After that you find a list of the parameters used to perform the continuation. This will only be of interest to you if something went wrong and you want to find out why.

Solutions:

Now finally you are given a complete list of all isolated solutions to the initial system. All solutions are written in the following format:

```
solution : 4      :          start residual : 1.921E-14
t : 1.0000000000000000E+00  0.0000000000000000E+00
```

```

m : 1
the solution for t :
x : 8.52582667329085E-01 5.38308418212131E-01
y : 7.30705684264524E-02 -1.00398009431747E+00
== err : 1.101E-16 = rco : 3.979E-01 = res : 1.921E-14 = complex regular ==

```

The important information you find in the lines which start with `x` and `y`. The first number denotes the real part and the second the imaginary part of the corresponding variable written as a floating point number. So in the above case $(x, y) = (0.852 + i 0.538, 0.073 - i)$ is an approximation to a root of the initial system. `m` denotes the multiplicity of the root. The other values give numerical information about the convergence and other things at this particular root. We don't want to go into the details here.

Note that if a homotpy path diverges the corresponding 'solution' will also be listed. You can spot them by their very unreasonable numerical values or by the words 'failure' and 'no solution'.

Summary:

After the solutions you find a summary of the number and type of the computed solutions and again some timing information.

Remarks:

1. Newer versions of `phc` append the solution set to the input file.
2. Type `phc -z (output file)` to extract the solutions from the output and write them into Maple format. (This works only with newer versions of `phc`.)

3 Different Modes of Usage

3.1 The Black-Box Solver

The black-box solver is started by typing '`phc -b (input-file) (output-file)`'. If you don't specify the input or output the programm will ask you for it.

No other input or specification is required.

This is the most comfortable mode since you don't have to take care of any parameter settings. For simple problems the black-box solver should be your first choice.

Note: The black-box solver uses the mixed volume to estimate the number of solutions. If your system has solutions with zero components the black-box solver may NOT FIND them.

3.2 The Tool-Box Mode

The tool-box mode gives you the opportunity to use components of `phc` separately and to specify the parameters which are used during the computations. This requires deeper knowledge of the program and of the mathematics behind the computations. We don't want to explain every component here and treat just some of them in the example and exercise section. A certain program component can be started by typing `phc -*` where `*` is the identifier of the desired tool. E.g. type `phc -m` to compute the mixed volume.

You can get a full list of all separate tools by typing `phc`. This list may vary depending on which version you have installed on your computer.

4 A Short Description of the Solver

4.1 The 4 Stages of the Solver

1. Preconditioning

- Coefficient Scaling
- Reduction of degrees

2. Root counting

- Bezout \rightarrow product homotopy
- Bernstein (Mixed Volume) \rightarrow coefficient homotopy
- Schubert \rightarrow SAGBI/Pieri homotopy

3. Homotopy continuation

- Tuning of continuation parameters
- Choice of predictor - corrector

4. Validation

- Refining the roots
- Analysis of path directions

4.2 Root Counts and Start Systems

Again there are various methods that `phc` uses to estimate the number of roots including Bezout number, multihomogenous Bezout number and mixed volume. The mixed volume will be discussed in the example and exercise section. Note that the mixed volume just counts the solutions in $(\mathbb{C}^*)^n$ and that all solution counting in general just gives an upper bound on the exact number.

After the root count `phc` constructs a new system of polynomial equations which has the same support set like the initial system but whose solutions are known in advance. The number of solutions should equal the number obtained by the root count. This start system and its solutions are then used to perform the polynomial continuation in the next step.

4.3 Polynomial Continuation and End Games

Suppose the system we want to solve is $f_i(x) = 0$ and the start system is denoted by $g_i(x) = 0$. Then we introduce a new complex variable t and set

$$h_i(x, t) := (1 - t) g_i(x) + t f_i(x)$$

We know the solution set of $h_i(x, t)$ for $t = 0$ and we want the set for $t = 1$. To obtain this set we raise t in little steps and for each new t we compute the solutions of $h_i(x, t)$ numerically (e.g with Newton's method) starting with the previous solution from last value of t .

When the start system and the transformation procedure, called a homotopy, are chosen suitably, the endpoints of these solution paths are guaranteed to include all isolated solutions of the target system.

The mathematics behind these computations is very well described in [2] and [5]

As t approaches 1 the diverging paths have to be separated from the paths converging to solutions. Also we want to decide whether we found a singular or a regular solution. This is done in so called 'end games' which we don't want to describe in detail here.

4.4 How to Detect Higher Dimensional Solution Sets

In the general case the solution set of a system of polynomial equations may consist of several components of different dimensions. Let n be the number of variables. To detect a component of dimension $n - 1$ cut the solution set with a generic hyperplane of dimension 1. This line meets all components of dimension $n - 1$ in a finite number of points, which can be computed using continuation. The line misses all lower dimensional components entirely. Now move the line around to collect samples of the solution set. To find components of dimension $n - k$ cut with a k -dimensional hyperplane and proceed as above (but don't forget about the higher dimensional solutions you found before). For details see [6].

5 Maple, MatLab and C interfaces

Scripts that call the `phc`-solver from inside one of the above mentioned softwares or programming language can be found in the download section

<http://www.math.uic.edu/~jan/download.html>.

These programs just translate an input from e.g. Maple format to an `phc`-file, then start `phc` independently and after that try to read the output-file and convert it back to the desired format.

See also [3] for a description of PHCmaple.

Note: Some of these scripts may need some fixing.

6 Examples and Exercises

Question 1:

1. Solve the following system using the black-box solver.

$$\begin{aligned}x^2 + 4y^2 - 4 &= 0 \\ 2y^2 - x &= 0\end{aligned}$$

2. Draw both curves with a software of your choice (e.g. Maple) and check if the real solutions you found with `phc` make sense.
3. Can you change the constants of the parabola such there are 4 real intersection points? Verify this with `phc`.

Question 2: Consider the following polynomial.

$$x \prod_{j=1}^{m-1} (x^2 - j)$$

1. How many roots do you expect? Why is this an extremal example regarding Descartes' Rule of Signs?

2. Compute the roots for $m = 3$ using `phc -b`. (The output will be slightly different from what we described above, but since it's short you will find the information you need.)

Question 3:

1. Solve the system

$$\begin{aligned} x^{108} + 1.1 y^{54} - 1.1 y &= 0 \\ y^{108} + 1.1 x^{54} - 1.1 x &= 0 \end{aligned}$$

using the black-box solver. (This may take a few minutes.)

2. While `phc` is working for you, look up Kouchnirenko's Conjecture and apply it to the above system.
3. Compare the prediction of the Conjecture and the number of solutions that `phc` found. See [1] for details.

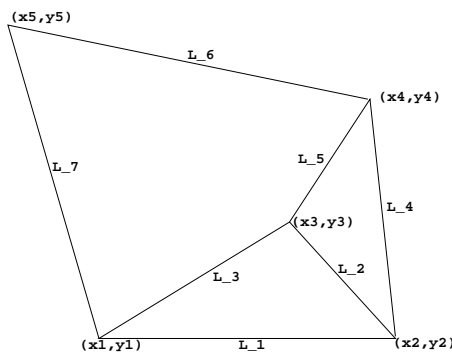
Question 4:

1. Solve the system

$$\begin{aligned} x_1^2 + x_2 - 3 &= 0 \\ x_1 + 0.125 x_2^2 - 1.5 &= 0. \end{aligned}$$

2. Give the multiplicity of the roots and say if they are regular or singular. (You will have to look at the long output file to see that.)
3. Compute the Jacobian of the system and verify the singular solutions.

Question 5: Consider the following graph with prescribed edge lengths



We fix the first and second vertex as $(x_1, y_1) = (1, 1)$ and $(x_2, y_2) = (3, 1)$. Furthermore we give the following values for the edge lengths.

$$L_1 = 2, \quad L_2 = 0.9, \quad L_3 = 1.3, \quad L_4 = 1.5, \quad L_5 = 0.7, \quad L_6 = 2.7, \quad L_7 = 1.9$$

1. Model this situation as a system of polynomial equations.
2. Use `phc -m` to compute the mixed volume of this system.
3. Use `phc` to compute the solutions to this system. How many do you get? Does this agree with the mixed volume?
4. Look up Bernshtein's second theorem (see the courses homepage if you can't find it anywhere else).
5. According to the theorem there has to be a direction w such that $\partial_w f = 0$ has a solution in $(\mathbb{C}^*)^6$. Find such a w .

Question 6: Cut the unit sphere in \mathbb{R}^3 with the following two hyperplanes.

$$\begin{aligned} 4x - 3y + 0.73z &= 0 \\ -x + 2.1y + 7z &= 0.2 \end{aligned}$$

1. Construct a start system for the polynomial continuation using `phc -r`.
2. Perform the continuation with this start system using `phc -p`.

References

- [1] B. Haas. *A Simple Counterexample to Kouchnirenko's Conjecture*. Beiträge Alg. Geom. 43 (2002).
- [2] B. Huber and B. Sturmfels. *A polyhedral method for solving sparse polynomial systems*. Math. Comp. 64, 212, 1541-1555.
- [3] A. Leykin and J. Verschelde. *PHCmaple: A Maple Interface to the Numerical Homotopy Algorithms in PHCpack*. In Proceedings of the Tenth International Conference on Applications of Computer Algebra (ACA'2004), edited by Quoc-Nam Tran, pages 139-147, 2004.

- [4] A. Leykin, and J. Verschelde. *Interfacing with the Numerical Homotopy Algorithms in PHCpack*. Proceedings of ICMS 2006, LNCS 4151, edited by Andrew Iglesias and Nobuki Takayama. Pages 354-360, Springer-Verlag, 2006. <http://www.ima.umn.edu/~leykin/PHCmaple/PHCmaple.pdf>
- [5] 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.
- [6] A. J. Sommese, J. Verschelde, and C. W. Wampler. *Numerical Irreducible Decomposition using PHCpack*. In Algebra, Geometry and Software Systems, edited by M. Joswig and N. Takayama, pages 109-130, Springer-Verlag 2003.
- [7] J. Verschelde. *Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation*. ACM Trans. on Math. Softw. 25(2):251-276, 1999.