Lecture III: Software and Applications

Jan Verschelde

Department of Math, Stat & CS
University of Illinois at Chicago
Chicago, IL 60607-7045, USA

Email: `jan@math.uic.edu`

URL: `http://www.math.uic.edu/~jan`

Homotopy Methods for Solving Polynomial Systems

tutorial at ISSAC'05, Beijing, China, 24 July 2005

Lecture III Outline

1. Examples, Applications, Benchmarks — ***formats***
2. Using phc in blackbox or toolbox mode — ***options***
3. Illustrations of `phc`, Exercises — ***hands on***

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`.

This material is based upon work supported by the National Science Foundation under Grant No. 0134611 and Grant No. 0410036.
Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Examples, Applications, Benchmarks

Input format for `phc`:

```
3  8
  x11*x22 - x21*x12;
  x12*x23 - x22*x13;
  x13*x24 - x23*x14;
```

The `i` and `I` represent the imaginary unit $\sqrt{-1}$.

`e` and `E` are used for floating-point numbers in scientific notation.

Impose on `phc` an order of the symbols, e.g.: `x` before `y`, giving the null polynomial `x + y - x - y` at the start of the input.

Solution Formats

```
THE SOLUTIONS :
1 2
============================================================
solution 1 :
t :  1.00000000000000E+00   0.00000000000000E+00
m : 1
the solution for t :
 x : -3.23606797749979E+00   0.00000000000000E+00
 y :  0.00000000000000E+00  -1.27201964951407E+00
== err :  1.174E-15 = rco :  1.079E-01 = res :  4.441E-16 ==
```

Complex values are listed as a sequence of real and imaginary numbers, `t` is the continuation parameter `t`, `m` is the multiplicity.

`err` is the magnitude of correction $|\Delta \mathbf{x}|$, `rco` is an estimate for inverse of condition number, and `res` is the magnitude of $|f(\mathbf{x})|$.

Using phc in blackbox or toolbox mode

`PHC` = Polynomial Homotopy Continuation in `PHCpack`.

We write `PHCpack` when we intend the whole package,
i.e.: source code, examples, makefiles, executables, etc.

`phc` is the program in executable form, on Linux, Windows, MacOS X, or SunOS, runs from the command line.

Two blackbox solvers:

- `phc -b`: original, for isolated solutions;
- `phc -a`: new equation-by-equation solver.

`phc` without option runs program interactively, after first showing all options.

Most commonly used options

- for isolated solutions:

`phc -r`	root counts and start systems
`phc -p`	polynomial continuation
`phc -v`	validation with Newton's method

- for solution sets:

`phc -c`	embeddings and cascade
`phc -f`	filter and factor
`phc -w`	witness set intersection

Newton's Method – exercise 1

```
2
  x^2 + y - 3;                    save as file
  x + 0.125*y^2 - 1.5;              "ojika1"
```

- `phc -b` look at #steps along a path,
  interpret `rco` for each root.
- `phc -v` using multiprecision arithmetic,
  how many iterations to have 32 decimal places right?
- `phc -v` deflate at the singular roots.
- `phc -v` compute multiplicity for each root.

Multihomogeneous Bézout Numbers – exercise 2

```
3
 0.5*(x^2 + 4*x*y + y^2 ) + x^2*y^2 - 1;        save as
 0.5*(y^2 + 4*y*z + z^2 ) + y^2*z^2 - 1;      "molecular1"
 0.5*(z^2 + 4*z*x + x^2 ) + z^2*x^2 - 1;
```

- `phc -r` create 3-homogeneous start system, save as file `"molecular1_start"`.
- `phc -p` solve using `"molecular1_start"`.
- `phc -b` verify what black-box solver does.
- `phc -p` change coefficients at random, verify that #roots stays constant.

Solving Binomial Systems – exercise 3

```
3
 x^2*y^3*z - 1;                    save as
 x*y^2*z^3 - 1;                 "binonomial"
 x^3*y*z^2 - 1;
```

- `phc -b` how many solutions?
- `phc -r` best possible Bézout number?

Polyhedral Methods – exercise 4

```
2
 x^3*y + x*y^2 + 1;                    save as
 x^4 + x*y + 1;                        "cayley"
```

- `phc -m` use dynamic lifting and Cayley trick
  to compute the Minkowski polynomial $5\lambda_1^2 + 8\lambda_1\lambda_2 + 4\lambda_2^2$.
- `phc -m` use static lifting to create a random coefficient start system,
  save it as `"cayley_start"`.
- `phc -p` solve using `"cayley_start"`.
- `phc -b` verify the calculations.

Factorization Methods – exercise 5

```
3  8
  x11*x22 - x21*x12;                    save as
  x12*x23 - x22*x13;                   "minors24"
  x13*x24 - x23*x14;
```

- `phc -c` add 5 random hyperplanes, save as `"minors24e5"`.
- `phc -b` solve `"minors24e5"`.
- `phc -f` factor using monodromy breakup certified by linear traces.
- `phc -f` factor using combinatorial enumeration of linear traces.
- Repeat for larger systems of adjacent minors of general 2-by-n matrix.

Computing Witness Sets – exercise 6

```
3
 (x1^2 - x2)*(x1 - 0.5);                    save as
 (x1^3 - x3)*(x2 - 0.5);                   "twistp4i"
 (x1*x2 - x3)*(x3 - 0.5);
```

- `phc -c` add 1 random hyperplane, save as `"twistp4i_e1"`.
- `phc -c` run cascade, extract from output witness set `"twistp4i_ws1"`, and candidate isolated points `"twistp4i_sw0"`.
- `phc -f` filter `"twistp4i_sw0"` with respect to `"twistp4i_ws1"`.
- `phc -a` runs equation-by-equation solver.

Special Parameter Values – exercise 7

```
3 4
 0.5*(x^2 + 4*x*y + y^2 ) + a*(x^2*y^2 - 1);        save as
 0.5*(y^2 + 4*y*z + z^2 ) + a*(y^2*z^2 - 1);      "molecular"
 0.5*(z^2 + 4*z*x + x^2 ) + a*(z^2*x^2 - 1);
```

- Add $\begin{cases} \frac{\partial f}{\partial \mathbf{x}} \boldsymbol{\lambda} = \mathbf{0} & \frac{\partial f}{\partial \mathbf{x}} \text{ is Jacobian, } \boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3) \\ \langle \mathbf{h}, \boldsymbol{\lambda} \rangle = 1 & \mathbf{h} \text{ is random 3-vector in } \mathbb{C}^3 \end{cases}$ save as `"special"`.
- `phc -b` solve `"special"`.
- Verify special values for `a`, either by Newton with deflation,

  or by computing a witness set.