# Solving Polynomial Systems in the Cloud with Polynomial Homotopy Continuation

Jan Verschelde
joint with Nathan Bliss, Jeff Sommars, and Xiangcheng Yu

University of Illinois at Chicago
Department of Mathematics, Statistics, and Computer Science
`https://kepler.math.uic.edu`

The 17th Workshop on Computer Algebra in Scientific Computing
14-18 September 2015, RWTH Aachen University, Aachen, Germany

## Outline

1. Polynomial Homotopy Continuation
    - solving systems with deformations
    - the software PHCpack and its interfaces
2. Solving in the Cloud
    - a web interface to phc
    - what we currently have
3. Classifying the Solved Systems
    - searching a database of polynomial systems
    - the graph isomorphism problem
    - benchmarking the canonization of systems

# Solving Polynomial Systems in the Cloud

1. Polynomial Homotopy Continuation
   - solving systems with deformations
   - the software PHCpack and its interfaces

2. Solving in the Cloud
   - a web interface to phc
   - what we currently have

3. Classifying the Solved Systems
   - searching a database of polynomial systems
   - the graph isomorphism problem
   - benchmarking the canonization of systems

## polynomial homotopy continuation methods

$\mathbf{f}(\mathbf{x}) = \mathbf{0}$ is a polynomial system we want to solve,
$\mathbf{g}(\mathbf{x}) = \mathbf{0}$ is a start system ($\mathbf{g}$ is similar to $\mathbf{f}$) with known solutions.

A homotopy $\mathbf{h}(\mathbf{x}, t) = (1 - t)\mathbf{g}(\mathbf{x}) + t\,\mathbf{f}(\mathbf{x}) = \mathbf{0}$, $t \in [0, 1]$,
to solve $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ defines solution paths $\mathbf{x}(t)$: $\mathbf{h}(\mathbf{x}(t), t) \equiv \mathbf{0}$.

Numerical continuation methods track the paths $\mathbf{x}(t)$, from $t = 0$ to 1, applying predictor-corrector algorithms.

Polynomial homotopy continuation methods are hybrid:

- Symbolic: definition of a homotopy $\mathbf{h}(\mathbf{x}, t) = \mathbf{0}$.
  We exploit the sparse structure of $\mathbf{f}$ when constructing $\mathbf{g}$.
- Numeric: path tracking with predictor-corrector algorithms.
  Verify approximate solutions with special care for diverging paths.

# Solving Polynomial Systems in the Cloud

## the software PHCpack

PHCpack is a package for Polynomial Homotopy Continuation.
ACM Transactions on Mathematical Software archived version 1.0 as Algorithm 795, vol. 25, no. 2, pages 251–276, 1999.

*blackbox solver:*
`phc -b` computes all isolated solutions of a polynomial system.

External software package integrated in PHCpack:

- Fast mixed volume computation by MixedVol of Gao, Li, and Wu, Algorithm 846 of ACM TOMS, vol. 31, pages 555–560, 2005.
- Double double and quad double arithmetic with the QD Library of Hida, Li, and Bailey published in the 15th IEEE Symposium on Computer Arithmetic, pages 155–162. IEEE, 2001.

Interfaces to Macaulay2 (Gross, Petrovic), Maple (Leykin), MATLAB (Guan), Python (Piret), Sage (Hampton, Jokela, Stein).

Version 2.4 supports GPU accelerated path trackers (Yoffe, Yu).

## interfaces to the software PHCpack

Three (pure) types of interfaces to the solvers in PHCpack:

1. Command line with interactive menus:

```
phc -b input output
```

2. Python scripting interface with `phcpy`:

```
>>> from phcpy.solver import solve
>>> help(solve)
>>> f = ['x*y + 3*x - 4;', 'x^2 + y^2 - 1;']
>>> s = solve(f)
```

3. A Graphical User Interface (GUI).

   We view a web interface as a GUI that runs in a browser.

# Solving Polynomial Systems in the Cloud

1. Polynomial Homotopy Continuation
   - solving systems with deformations
   - the software PHCpack and its interfaces

2. Solving in the Cloud
   - a web interface to phc
   - what we currently have

3. Classifying the Solved Systems
   - searching a database of polynomial systems
   - the graph isomorphism problem
   - benchmarking the canonization of systems

## motivation for a cloud service

In some disciplines, cloud computing has become the norm.

Benefits for the user (but there are risks as well):

- No installation is required, just sign up.

  Installing software can be complicated and a waste of time, especially if one want to perform a single experiment.
  The user should not worry about upgrading to newer versions.

- We offer a computing service.

  The web server is hosted by a powerful computer, which can be extended with the addition of compute servers.

- Files and data are stored and managed for the user.

  The input and output files are managed at the server.
  For larger problems, storage space can become an issue.

## Macaulay2 in the cloud

`PHCpack.m2` is a package distributed with Macaulay2.

Macaulay2 runs in the cloud as well.

An example session with truncated output is below:

```
Macaulay2, version 1.8.1

i1 : loadPackage "PHCpack";

i2 : R = CC[x,y,z];

i3 : S = {x+y+z-1, x^2 + y^2 - 1, x+2*y-3};

i4 : solveSystem(S)

o4 = {{.6+.8*ii, 1.2-.4*ii, -.8-.4*ii},
      {.6-.8*ii, 1.2+.4*ii, -.8+.4*ii}}
```

## Solving Polynomial Systems in the Cloud

1. Polynomial Homotopy Continuation
   - solving systems with deformations
   - the software PHCpack and its interfaces
2. Solving in the Cloud
   - a web interface to phc
   - what we currently have
3. Classifying the Solved Systems
   - searching a database of polynomial systems
   - the graph isomorphism problem
   - benchmarking the canonization of systems

## what we currently have

A web interface to the blackbox solver of `phc` is running at `https://kepler.math.uic.edu`.

1. The server `kepler` runs Red Hat Linux.
2. Apache is the web server.
3. Our database is MySQL.
4. Python is the scripting language.

All software is free and open source.

The web service was also deployed and tested on a Mac OS X.

In its current state, the setup of the web interface is minimal,
but, most importantly: It works!

## Apache

The web server runs Apache.

- One `index.html` leads to the login Python script.
- The `cgi-bin` directory contains all scripts.

The setup process is automatically executed at a reboot.

Registration is done automatically via a google email account.

## five Python scripts

The original, proof-of-concent version had less than 1,500 lines of Python code.

A simplified view of the original design:

- `cookie_login` prints first login screen
  - calls `register` for a first time user; or
  - calls `phc_solver`
- `register` sends email to first time user
- `activate` runs when user clicks in email
- `contact` is optional to send emails about the service
- `phc_solver` solves polynomial systems

## MySQL

MySQL is called in Python through the module `MySQLdb`.

The database manages two tables:

- `users`: data about users, encrypted passwords;
- `polys`: references to systems and solutions.

Mathematical data are not stored in the database:

- Every user has a folder, a generated 40 character string.
- With every system there is another generated 40 character string.

# Solving Polynomial Systems in the Cloud

## motivation to classify polynomial systems

Some interesting questions:

- How hard is a ***new*** system compared to ones already solved?
  The research community works with benchmark systems which are although representative for
  - the difficulties of the problems in the field; and
  - the capabilities of the software;

  but not really meaningful or directly relevant to ***new*** users.

  For example: who knows about the cyclic $n$-roots problems?

- Has the ***same*** system been solved already?
  If we would have a large database of solved systems,
  then how to search the database?

  ***New*** users will unlikely know the names of benchmark problems.

- Is the system on input ***similar*** to solved systems?

## the classification problem

If we have solved a system with the same structure,
then the solved system is the start system in a homotopy.

For example, the systems

$$\left\{ \begin{array}{r} x^2 + xy^2 - 3 = 0 \\ 2x^2y + 5 = 0 \end{array} \right. \quad \text{and} \quad \left\{ \begin{array}{r} 3 + 2ab^2 = 0 \\ b^2 - 5 + 2a^2b = 0 \end{array} \right.$$

are isomorphic to each other. There support sets are

$$\begin{array}{rl} & \{\{(2,0),(1,2),(0,0)\},\{(2,1),(0,0)\}\} \\ \text{and} & \{\{(0,0),(1,2)\},\{(0,2),(0,0),(2,1)\}\}. \end{array}$$

**Definition**

Two sets of support sets are *isomorphic* if there exists a permutation of their equations and variables so that the sets of support sets are identical.

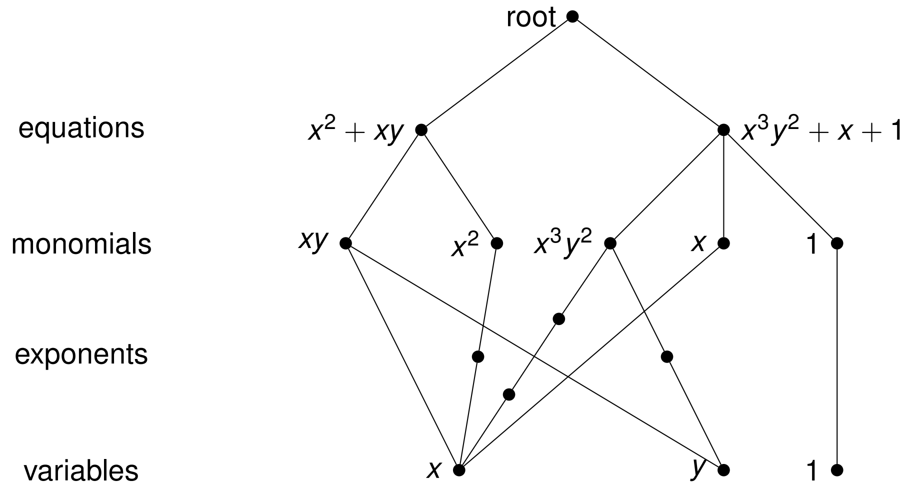
## the isomorphism problem of polynomials

Related work occurs in multivariate cryptography.

- J. Patarin. **Hidden fields equations (HFE) and isomorphism of polynomials (IP): Two new families of asymmetric algorithms.** In U. Maurer, editor, *Advances in Cryptology - EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48. Springer-Verlag, 1996.
- J.-C. Faugère and L. Perret. **Polynomial equivalence problems: Algorithmic and theoretical aspects.** In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 30–47. Springer-Verlag, 2006.
- C. Bouillaguet, P.-A. Fouque, and A. Véber. **Graph-theoretic algorithms for the “Isomorphism of Polynomials” problem.** In T. Johansson and P. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 211–227. Springer-Verlag, 2013.

## Solving Polynomial Systems in the Cloud

## representing a system as a graph



## the graph isomorphism problem

**Definition**

The *graph isomorphism problem* asks whether for two undirected graphs $F, G$ there is a bijection $\phi$ between their vertices that preserves incidence; i.e.: if $a$ and $b$ are vertices connected by an edge in $F$ (respectively $G$), then $\phi(a)$ and $\phi(b)$ are connected by an edge in $G$ (respectively $F$).

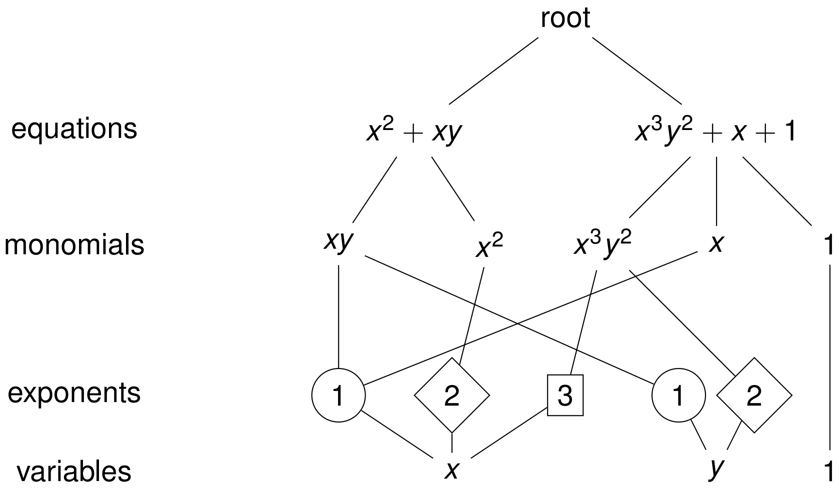
**Proposition**

*The problem of determining whether two sets of support sets are isomorphic is equivalent to the graph isomorphism problem.*

A practical solution: the software package **nauty**.

B.D. McKay and A. Piperno. **Practical graph isomorphism, II.** *Journal of Symbolic Computation*, 60:94–112, 2014.

# canonical graph labelings with nauty



# Solving Polynomial Systems in the Cloud

## benchmarking nauty on cyclic $n$-roots

$$\text{cyclic 3-roots: } \left\{ \begin{array}{r} x_1 + x_2 + x_3 = 0 \\ x_1 x_2 + x_2 x_3 + x_3 x_1 = 0 \\ x_1 x_2 x_3 - 1 = 0. \end{array} \right.$$

Times in milliseconds for small values of $n$,
to compute the canonical form with `nauty`:

| $n$ | time | #nodes | #characters |
|---:|---:|---:|---:|
| 4 | 0.006 | 29 | 526 |
| 6 | 0.006 | 53 | 1,256 |
| 8 | 0.006 | 85 | 2,545 |
| 10 | 0.007 | 125 | 5,121 |
| 12 | 0.007 | 173 | 8,761 |

Note: computing the root counts by `phc -b`
for the cyclic 10-roots problems takes 48.8 seconds.

## cyclic *n*-roots, for large *n*

For larger values of the dimension of the cyclic $n$-root problem, times and sizes of the data start to grow exponentially.

| $n$ | time | #nodes | #characters |
|---:|---:|---:|---:|
| 16 | 0.010 | 293 | 20,029 |
| 32 | 0.045 | 1,093 | 168,622 |
| 48 | 0.265 | 2,405 | 601,702 |
| 64 | 1.200 | 4,229 | 1,427,890 |
| 80 | 4.316 | 6,565 | 2,778,546 |
| 96 | 15.274 | 9,413 | 4,784,390 |
| 112 | 38.747 | 12,773 | 8,595,408 |
| 128 | 80.700 | 16,645 | 13,094,752 |

Note: computing all isolated solutions is no longer possible.
With GPU acceleration, tracking a limited number of paths is possible.

## game theory: Nash equilibria

Totally mixed Nash equilibria for three players with two pure strategies:

$$
\text{Nash 3}: \left\{
\begin{array}{rcl}
c_{1,1}p_2 + c_{1,2}(1-p_2) + c_{1,3}p_3p_2 + c_{1,4}(1-p_3)p_2 & & \\
+ c_{1,5}p_3(1-p_2) + c_{1,6}(1-p_3)(1-p_2) & = & 0 \\
c_{2,1}p_1 + c_{2,2}(1-p_1) + c_{2,3}p_3p_1 + c_{2,4}(1-p_3)p_1 & & \\
+ c_{2,5}p_3(1-p_1) + c_{2,6}(1-p_3)(1-p_1) & = & 0 \\
c_{3,1}p_1 + c_{3,2}(1-p_1) + c_{3,3}p_2p_1 + c_{3,4}(1-p_2)p_1 & & \\
+ c_{3,5}p_2(1-p_1) + c_{2,6}(1-p_2)(1-p_1) & = & 0
\end{array}
\right.
$$

unknowns: $p_1, p_2, p_3$; coefficients: $c_{i,j}$, $i = 1, 2, 3$, $j = 1, 2, \ldots, 6$.

## canonization of Nash equilibria

The system to compute all totally mixed Nash equilibria has supports invariant by the full permutation group.

| $n$ | time | #nodes | #characters |
|---:|---:|---:|---:|
| 4 | 0.006 | 47 | 977 |
| 5 | 0.006 | 98 | 2,325 |
| 6 | 0.007 | 213 | 7,084 |
| 7 | 0.013 | 472 | 18,398 |
| 8 | 0.054 | 1,051 | 51,180 |
| 9 | 0.460 | 2,334 | 134,568 |
| 10 | 4.832 | 5,153 | 331,456 |
| 11 | 73.587 | 11,300 | 872,893 |
| 12 | 740.846 | 24,615 | 2,150,512 |

The systems are not sparse and the number of solutions grows as $n!$.

## a system without symmetry

Another benchmark system is formulated by Katsura:

$$\text{katsura 2}: \left\{ \begin{array}{r} u_0 + 2u_1 + 2u_2 - 1 = 0 \\ u_0^2 + 2u_1^2 + 2u_2^2 - u_0 = 0 \\ 2u_0 u_1 + 2u_1 u_2 + u_2^2 - u_1 = 0. \end{array} \right.$$

Without symmetry in the support sets, the cost of the canonization increases not as fast in the dimension *n*.

| $n$ | time | #nodes | #characters |
|---:|---:|---:|---:|
| 25 | 0.020 | 929 | 24,906 |
| 50 | 0.090 | 3,411 | 112,654 |
| 75 | 0.546 | 7,454 | 254,770 |
| 100 | 1.806 | 13,061 | 495,612 |
| 125 | 4.641 | 20,229 | 793,662 |
| 150 | 10.860 | 28,961 | 1,157,498 |
| 175 | 21.194 | 39,254 | 1,587,115 |
| 200 | 52.814 | 51,111 | 2,082,562 |
| 225 | 98.118 | 64,529 | 2,643,891 |

The number of solutions equals $2^n$.

## concluding remarks

Basic version of web interface to `phc -b` and `phc -p`.

- Built with LAMP stack, Python as scripting language.
- Classification of solved systems via the graph isomorphism.

In the future: automatic exploitation of permutation symmetry.

try it at **https://kepler.math.uic.edu**