solving polynomial systems in the cloud with phc

Jan Verschelde

University of Illinois at Chicago
Department of Mathematics, Statistics, and Computer Science
`http://www.math.uic.edu/~jan`
`jan@math.uic.edu`

Graduate Computational Algebraic Geometry Seminar

solving polynomial systems in the cloud with phc

1. Introduction
   - motivation for a cloud service
   - what we currently have
2. A Blackbox Solver
   - what does `phc -b` do?
   - what should `solve` do?
3. Extensions
   - phcpy and PHCpack.m2
   - visualization
   - changing the way we solve polynomial systems

solving polynomial systems in the cloud with phc

1. Introduction
   - motivation for a cloud service
   - what we currently have
2. A Blackbox Solver
   - what does `phc -b` do?
   - what should `solve` do?
3. Extensions
   - phcpy and PHCpack.m2
   - visualization
   - changing the way we solve polynomial 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.

what we currently have

As a result of a joint project with Xiangcheng Yu,
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

Less than 1,500 lines of Python code.

- `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 with phc

what is a blackbox solver?

What to expect from a blackbox solver?

- The polynomials are the *only* input to the solver.
- Parameters that control the execution options are
  - set to *work well on a large class of examples*; and/or
  - tuned automatically during the solving process.
- The output contains various *diagnostics* and checks.

  The user should be
  - warned in case of ill conditioning and nearby singularities;
  - able to verify (or falsify) the computed results.

what does phc -b do?

Polyhedral homotopies are optimal for sparse polynomial systems:

- the root count (the mixed volume) is sharp for generic problems;
- every path in a polyhedral homotopy ends at an isolated root, except for systems that has special initial forms.

The blackbox solver was designed for *square* problems,
that is: as many equations as unknowns.

Special cases:

- linear systems and polynomials in one variable
- binomial systems (exactly two monomials in every polynomial)
    - isolated solutions determined by Hermite normal form;
    - positive dimensional solution sets are monomial maps.

more concretely, what does `phc -b` really do?

Stages in the solving process:

1. Parse and classify the polynomial system.
   - Parse: attempt to gracefully handling of syntax errors.
   - Classify: handle univariate, linear, and binomial directly.
2. If not univariate, linear, or binomial, then two cases remain:
   - the system is square (as many equations as unknowns),
   - the system is overdetermined or underdetermined.

The nonsquare case is still experimental (as is `phc -a`), `phc -a` gives access to an equation-by-equation solver.

the square case

The original blackbox solver has its focus on isolated solutions.

For the square case, we have four types of start systems:

1. start system based on the total degree,
2. one multi-homogenous partition of the set of variables,
3. general linear-product start systems,
4. random coefficient start systems solved by polyhedral homotopies.

The start system with the lowest root count is selected and solved.

Path tracking to the target system gives solutions, to classify

1. detect path clustering, gather multiplicities;
2. frequency tables of forward errors, backward errors (residuals), and estimates for condition numbers.

what should `solve` do?

The `solve` should be *the* blackbox solver.

Goal: without assumptions on the dimension of the solutions.

Following ideas from tropical algebraic geometry we can generalize polyhedral homotopies for positive dimensional sets:

- Compute tropisms based on initial forms with some solutions.
- Develop Puiseux series starting at those initial form solutions.

The cost of the solver should be polynomial in the output size:

- Monitor progress of computation with intermediate results.
- Make predictions on the remaining computation time.

High level parallelism with heterogeneous methods:
$\longrightarrow$ Let several processes running different solution strategies compete!

solving polynomial systems in the cloud with phc

phcpy and PHCpack.m2

The package `phcpy` gives PHCpack a scripting interface,
the development of `phcpy` builds on PHCpack.

The user can build a solution method with scripts.

Several avenues for application in cloud server:

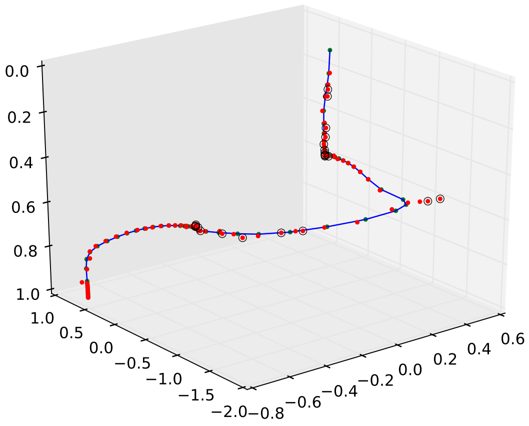
- Upgrade the existing `phc.py` in Sage.
- Scientific Python, with `numpy`, `scipy`, `sympy`, and `matplotlib`.
- Standalone: Graphical User Interface (GUI) to guide the user in addition to the worksheet approach to the web interface.

There are advantages and disadvantages to all avenues.

And then of course, there is Macaulay2, and the package `PHCpack.m2` could be made available in its cloud service as well.

visualization

The new path drawer of Xiangcheng Yu uses `matplotlib` on the output of the path tracker of `phc`, for example:



metadata, data, and expertise

The metadata we store about a polynomial system:

- Provenance, who posed this problem?
- Occurrence in the literature, routine or research?
- Methodology for this problem?

What actual data about the system we want to keep:

- Input: Formulation of the polynomial equations, separating coefficients from parameters.
- Output: Root counts, dimensions and degrees.

Make experience with solving polynomial systems automatic:

- Pattern matching of new system with solve systems.
- Evaluate solution methodologies.