

Solving Polynomial Systems in Python

phcpy: a scripting interface for PHCpack

Jan Verschelde

University of Illinois at Chicago
Department of Mathematics, Statistics, and Computer Science

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

www.phcpyack.org

Python devroom, FOSDEM 2019, 3 February, Brussels, Belgium

Outline

1 Introduction

- PHCpack and phcpy
- project history of phcpy
- design, implementation, documentation

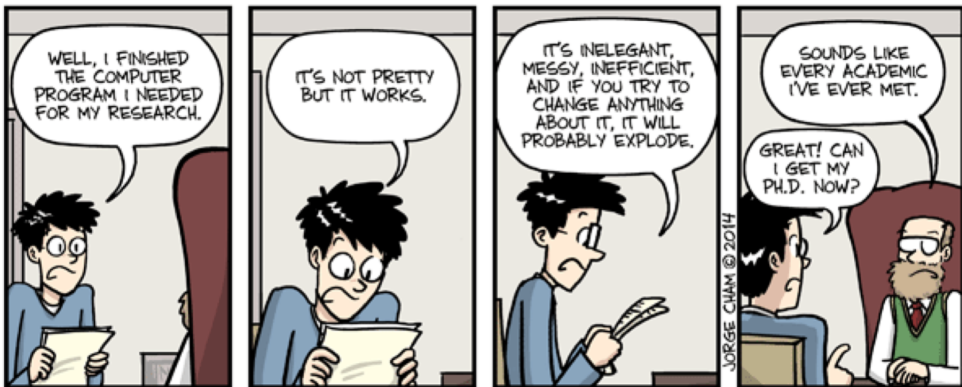
2 Solving Polynomial Systems

- an example from mechanical design
- step-by-step solution path tracking

3 Cloud Computing

- a first web interface: CGI scripting
- a second web interface: JupyterHub
- conclusions and future developments

the software originated in my 1996 PhD thesis ...



WWW.PHDCOMICS.COM

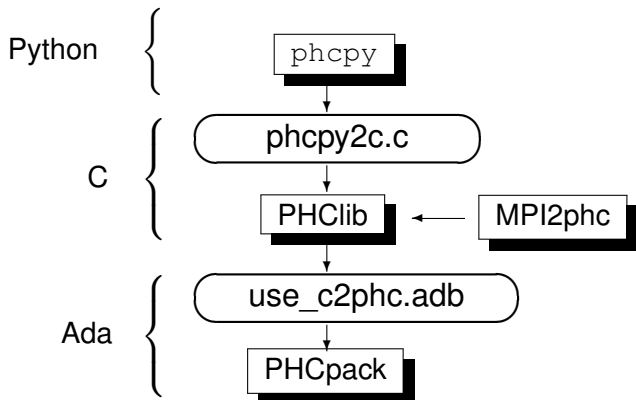
Homotopy Continuation Methods for Solving Polynomial Systems.
K.U. Leuven, 1996.

ACM Transactions on Mathematical Software archived
version 1.0 of PHCpack as Algorithm 795, 1999.

milestones in the project history of phcpy

- 1 Fall 2006 workshop on software for algebraic geometry, IMA.
IMA = Institute for Mathematics and its Applications
- 2 PhD thesis of Kathy Piret in 2008.
- 3 Sage has the interface `phc.py`, developed by William Stein, Marshall Hampton, and Alex Jokela.
- 4 EuroSciPy 2013 proceedings paper describes version 0.1.4
- 5 NSF award ACI 1440534 funds a Sustainable Software Element.
- 6 Xiangcheng Yu defines a first web interface, described with Nathan Bliss and Jeff Sommars in the CASC 2015 proceedings.
CASC = Computer Algebra in Scientific Computing
- 7 Summer 2017, setup of JupyterHub with Jasmine Otto.
- 8 Tutorial at CASC 2017, Beijing, China, in September 2017.
- 9 One week SageMath coding sprint at IMA in October 2017 with N. Bliss, T. Brysiewicz, T. Duff, M. Hampton, and J. Sommars.

design: application of a façade pattern



Goal: export all functionality of PHCpack to the Python programmer.

extending Python using shared object files

Compiled code can be called without loss of efficiency.

The initialization of `phcpy` imports `phcpy2c2.so`, a shared object.

We are using a Python implementation written in C, built with `gcc`.

Four steps in making “boilerplate” software are sketched below:

- Include `Python.h`.
- Add `PyObject *function_name()` wrappers for each function.
- Add a `PyMethodDef ModuleMethods[]` table, with entries for each function.
- Add a module initializer function.

The Python `distutils` package helps with compilation.

Static linking, including the whole archive, makes that the shared object can be used on systems that do not have the `gnu-ada` compiler.

documenting your code with Sphinx

Goal: export all functionality of PHCpack to the Python programmer.

Python programmers are informed by providing good documentation, which typically consists in four parts;

(1) getting started introduction; (2) tutorial with use cases; (3) user's manual; and (4) a reference manual.

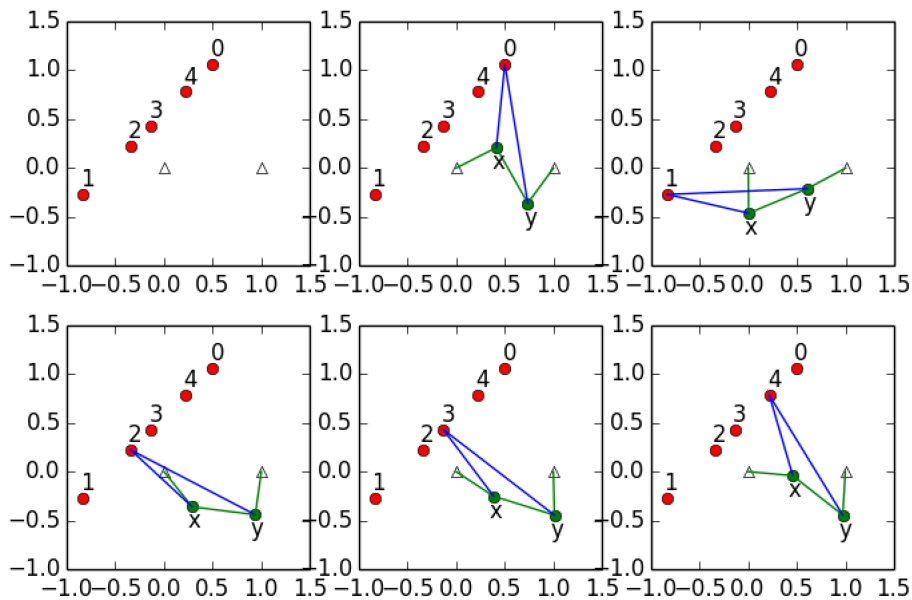
Sphinx makes the reference manual automatic, as documentation strings of functions in each module are included.

The `sphinx-quickstart` is an interactive tool that gives you a workable template in a first quick session.

One can start writing the user's manual using the code of the test functions in each Python module.

Finding great use cases for the tutorial are typically the hardest part, restructured text is so much easier to use than \LaTeX .

design a four bar mechanism through five points



a blackbox solver

If the file `input` contains

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

then `phc -b input output` puts the solutions in the file `output`.

The corresponding `phcpy` session:

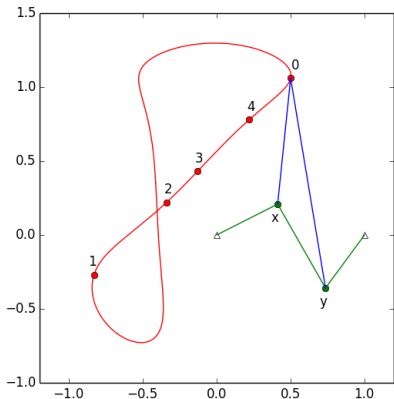
```
>>> from phcpy.solver import solve
>>> p = ['x**2 + 4*y**2 - 4;', '2*y**2 - x;']
>>> s = solve(p)
```

The `s` contains all *isolated* solutions of `p`.

solving a use case of the phcpy tutorial

We benefit from Python's computational ecosystem.

use case from the phcpy tutorial:



reproduces J. Mech. Design paper

Three steps:

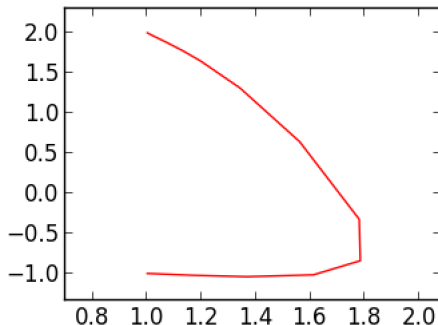
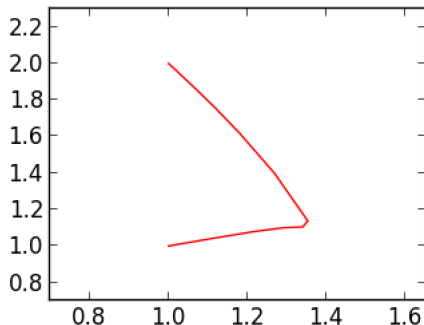
- 1) Given the precision points, formulate the polynomials, using SYMPY.
- 2) Apply the solver `s = solve(p)`.
- 3) There are 36 complex solutions. We extract the real solutions and visualize using MATPLOTLIB.

step-by-step solution path tracking

What contribution does Python make to the solving?

Well, solutions are computed by deforming polynomials with known solutions to the system we want to solve.

Paths of solutions are discretized. In a scripting environment, we give control to the user, to compute the next point on a path.



And now we have a research question: why do paths have this shape?

CGI scripting


Python comes with batteries included.

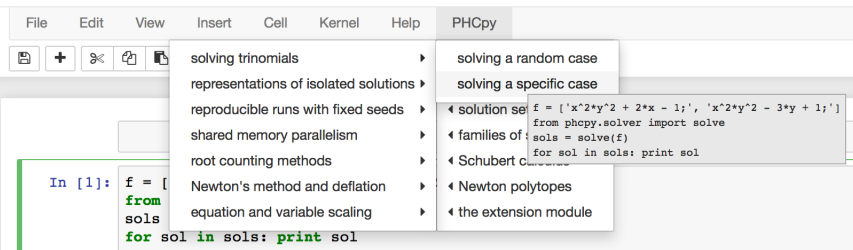
- 1 Posting and processing of HTML forms:
 - ▶ pure Python code prints the HTML code; and
 - ▶ the input of the forms is processed with Python functions.
- 2 `MySQLdb` does the management of user data:
 - ▶ names and encrypted passwords,
 - ▶ generic, random folder names to store data files,
 - ▶ file names with polynomial systems solved.
- 3 With the module `smtplib` we define email exchanges:
 - ▶ automatic 2-step registration process,
 - ▶ automatic password recovery protocol.

Test offline on `localhost` till confident to deploy.

The above approach was applied in our first web interface.

phcpy in a SageMath kernel of a Jupyter notebook

 jupyter code_snippet Last Checkpoint: 4 minutes ago (autosaved)



The screenshot shows the Jupyter notebook interface with the PHCpy menu open. The menu items are:

- solving trinomials
- representations of isolated solutions
- reproducible runs with fixed seeds
- shared memory parallelism
- root counting methods
- Newton's method and deflation
- equation and variable scaling
- solving a random case
- solving a specific case
- ◀ solution set
- ◀ families of
- ◀ Schubert classes
- ◀ Newton polytopes
- ◀ the extension module

The code cell contains the following Python code:

```
In [1]: f = [  
from  
sols  
for sol in sols: print sol
```

```
PHCv2.4.64 released 2019-01-21 works!  
total degree : 16  
2-homogeneous Bezout number : 8  
  with partition : { x }{ y }  
general linear-product Bezout number : 8  
  based on the set structure :  
    { x }{ x }{ y }{ y }  
    { x }{ x }{ y }{ y }  
mixed volume : 4  
stable mixed volume : 4  
t : 1.0000000000000000E+00  0.0000000000000000E+00  
m : 1  
the solution for t :  
  x : 4.86132470489966E-01  0.0000000000000000E+00  
  y : 3.42578353006690E-01 -2.28597478256455E-100  
== err : 3.499E-17 = rco : 8.882E-01 = res : 4.857E-17 =
```

- Code snippets suggest typical applications, and guide the novice user.
- Solve polynomials by pointing and clicking.

Jupyter and JupyterHub

The Jupyter notebook supports language agnostic computations, supporting execution environments in several dozen languages. With JupyterHub, we can run the code in a Python Terminal session, in a Jupyter notebook running Python, or in a SageMath session.

For the user administration, we recycled our first web interface.

With JupyterHub, we provide user accounts on our server.

- At login time, a new process is spawned.
- Users have generic, random login names.
- Actions of users must be isolated from each other.

The setup requires some system administration expertise.

conclusions and future developments

The current version of phcpy is 0.9.2.

- Exports the functionality of PHCpack to the Python programmer.
- Offers new functionality: a step-by-step path tracker.
- Available in the cloud in a SageMath kernel of a Jupyter notebook.

Future developments, beyond version 1.0.0 ...

- Improve systematic interface code development.
- Interactive pipelined parallel computations.
- Promote computational science research projects.