Solving Polynomial Systems with PHCpack and phcpy

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
4. Short Review
   - research software

MCS 507 Lecture 40
Mathematical, Statistical and Scientific Software
Jan Verschelde, 22 November 2023

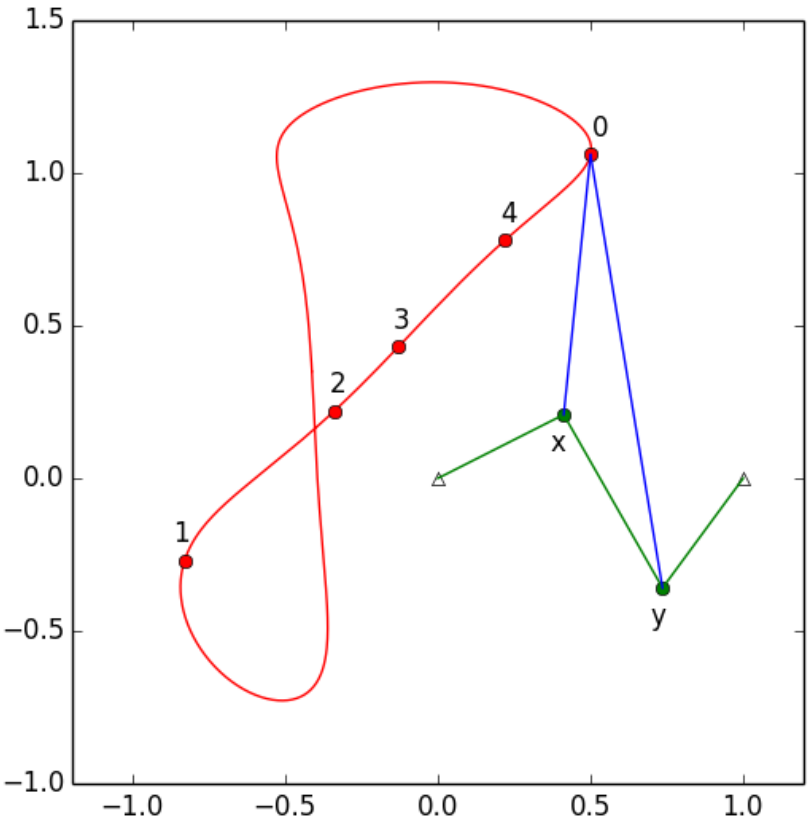
Solving Polynomial Systems with PHCpack and phcpy

Solving Polynomial Systems with PHCpack and phcpy

PHCpack is software for Polynomial Homotopy Continuation

phcpy is a new Python package, available at `www.phcpack.org`

use case from the phcpy tutorial:



reproduces J. Mech. Design paper

jupyter bbsolvesnippet

Logout

File Edit View Insert Cell Kernel Help PHCpy SageMath 8.0

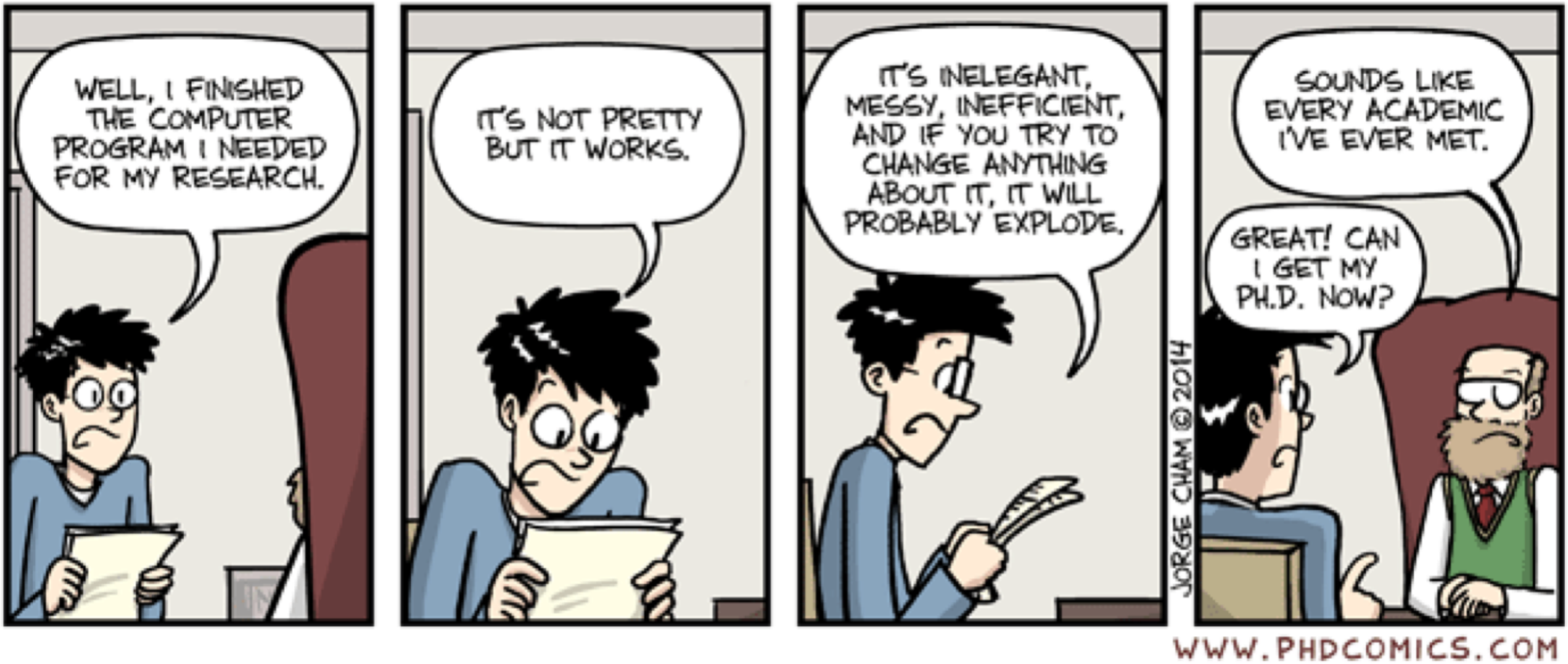
Code

- blackbox solver
- path trackers
- solution sets
- families of systems
- Schubert calculus
- Newton polytopes
- the extension module

```
In [2]: f = ['x*y^2 + y - 3;', 'x^3 - y +
        from phcpy.solver import solve
        sols = solve(f)
        for sol in sols: print sol
```

```
total degree : 9
2-homogeneous Bezout number : 7
  with with partition : { x }{ y }
general linear-product Bezout number : 7
  based on the set structure :
     { x }{ y }{ y }
     { x y }{ x }{ x }
mixed volume : 7
stable mixed volume : 7
t :  1.00000000000000E+00   7.13177119756522E+00
m : 1
the solution for t :
 x : -1.14928524947248E+00  -4.33149270057445E-01
 y :  1.28839810793789E-01  -1.63511747105322E+00
== err :  1.650E-16 = rco :  3.038E-01 = res :  2.220E-16
=
t :  1.00000000000000E+00   2.72148344088863E+00
m : 1
the solution for t :
 x : -1.14928524947248E+00   4.33149270057445E-01
 y :  1.28839810793789E-01   1.63511747105322E+00
```

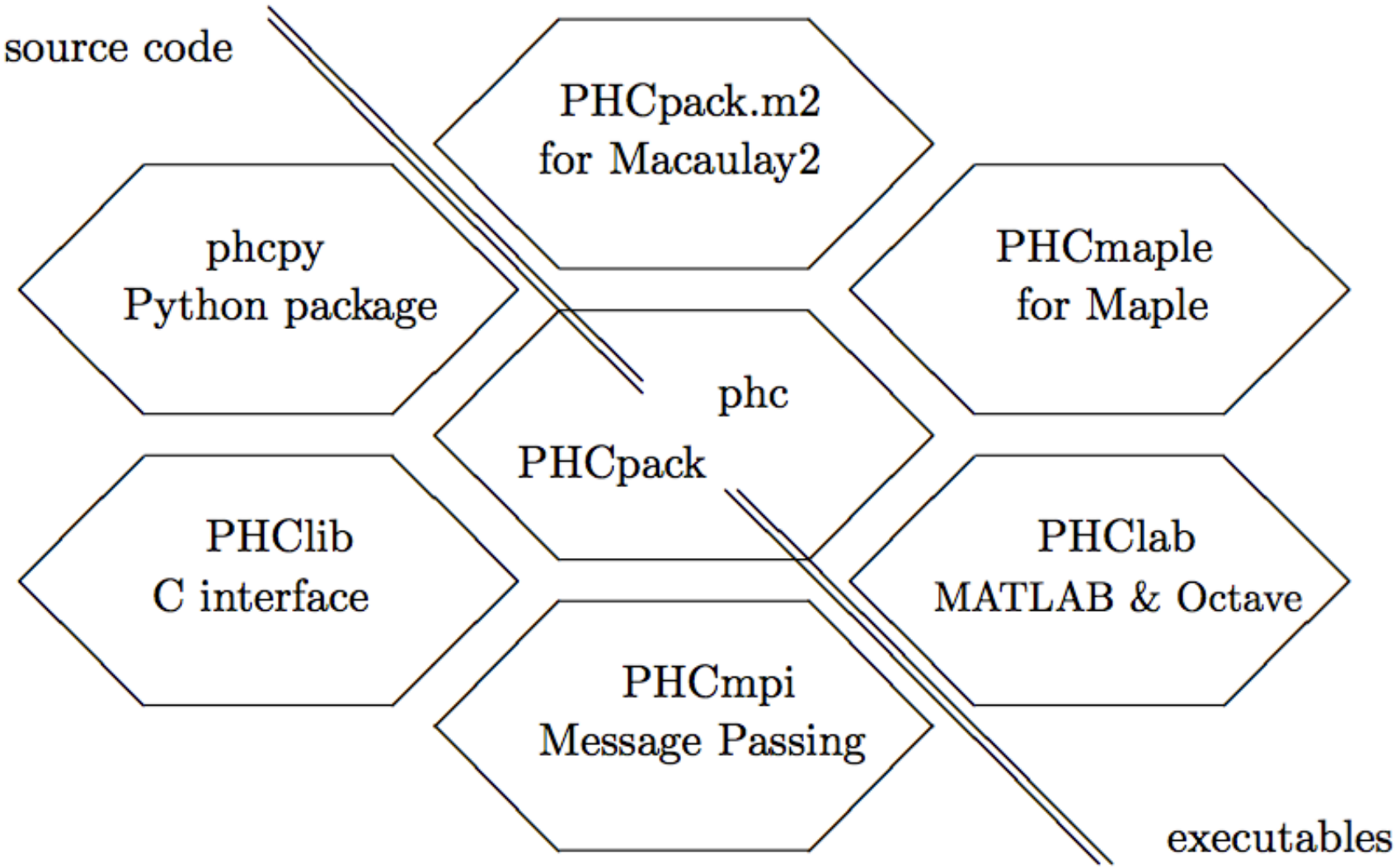
the software originated in my 1996 PhD thesis ...



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.

interfaces


source code
PHCpack.m2
for Macaulay2
phcpy
Python package
PHCmaple
for Maple
phc
PHCpack
PHClib
C interface
PHClab
MATLAB & Octave
PHCmpi
Message Passing
executables

Solving Polynomial Systems with PHCpack and phcpy

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
4. Short Review
   - research software

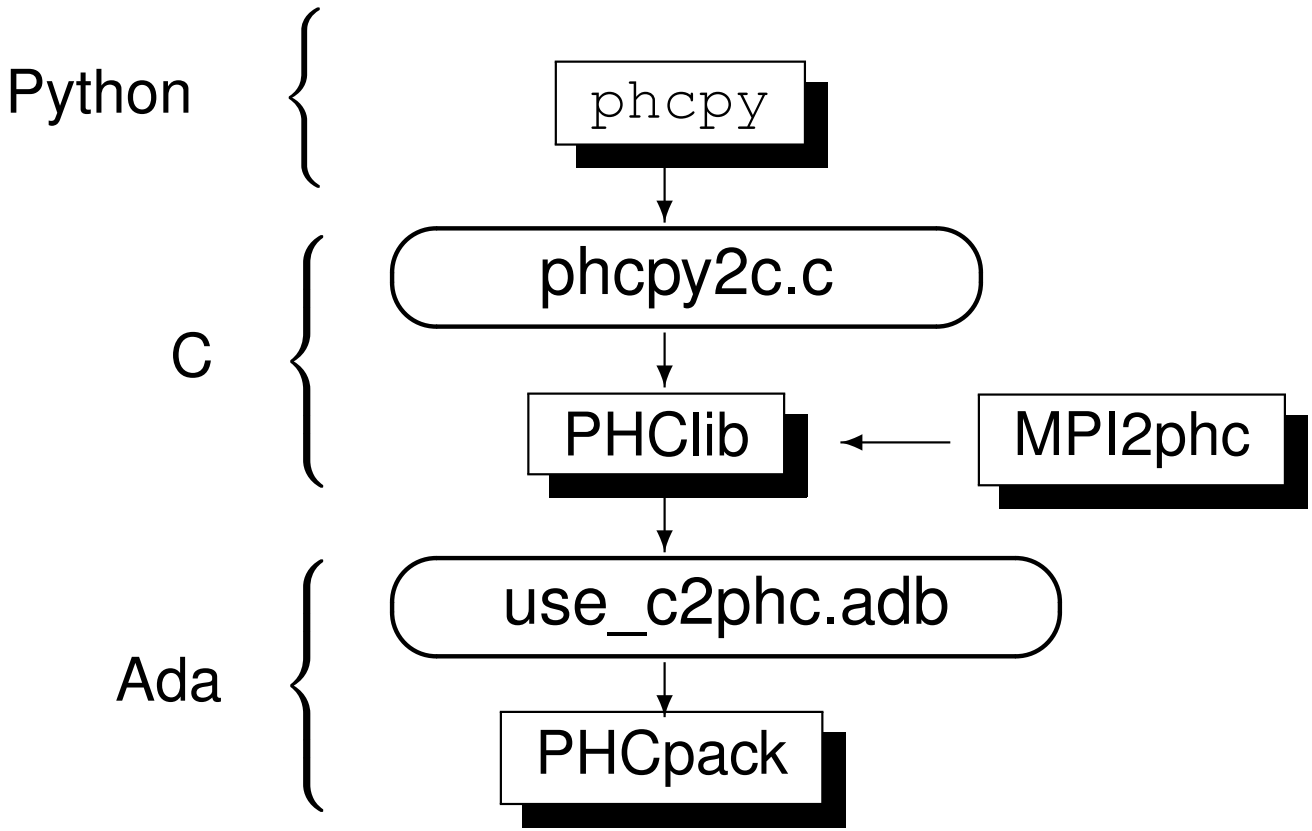
milestones in the project history of phcpy

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.
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.
10. Presented at Python devroom at FOSDEM 2019.
11. In SciPy proceedings 2019, with Angus Forbes and Jasmine Otto.

Solving Polynomial Systems with PHCpack and phcpy

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
4. Short Review
   - research software

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.

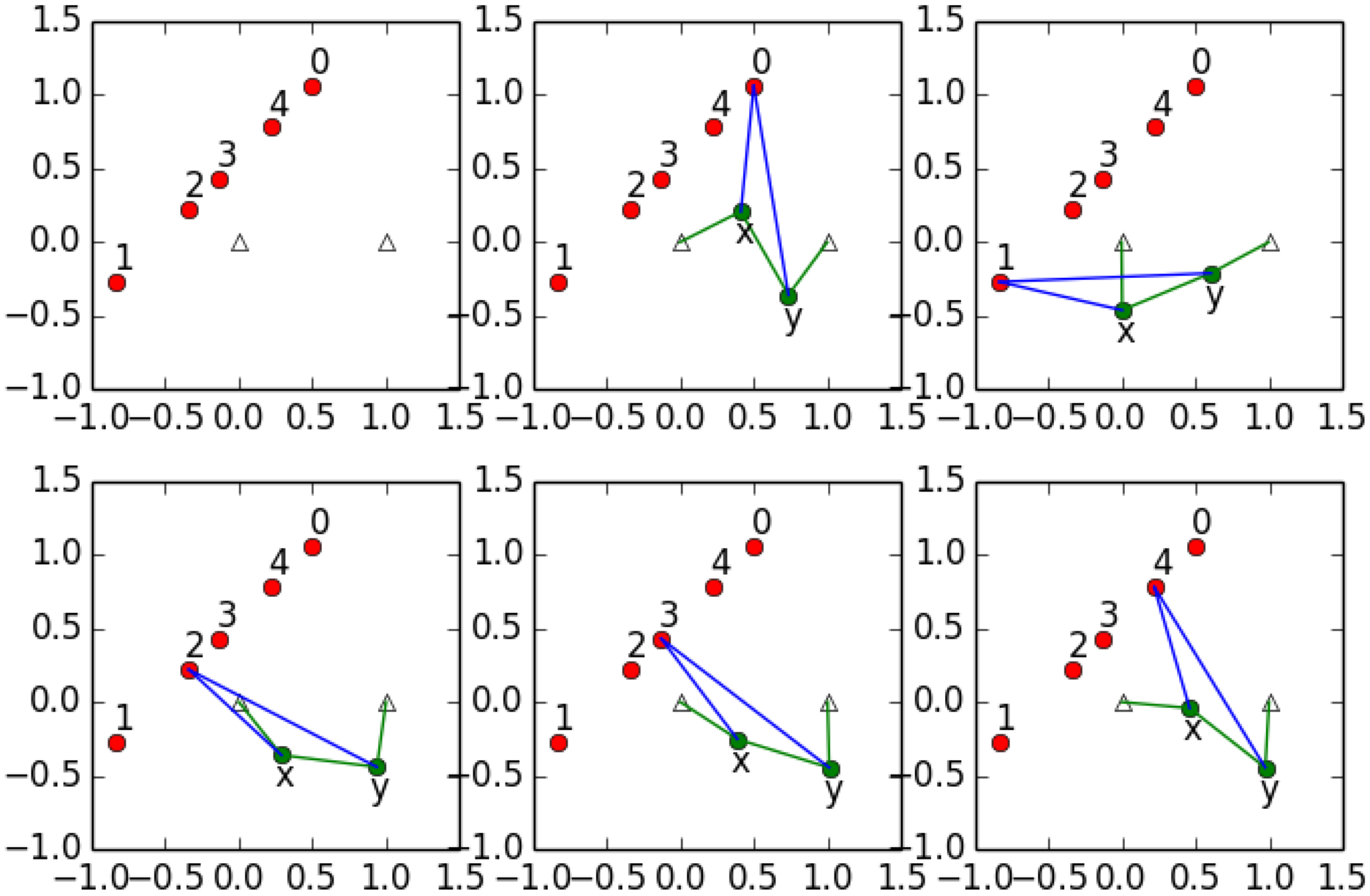
One can start writing the user's manual using the code of the test functions in each Python module.

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

Solving Polynomial Systems with PHCpack and phcpy

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
4. Short Review
   - research software

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`.

polynomial homotopy continuation

$\mathbf{f}(\mathbf{x}) = \mathbf{0}$ 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 $\mathbf{h}(\mathbf{x}, t) = \gamma(1-t)\mathbf{g}(\mathbf{x}) + t\,\mathbf{f}(\mathbf{x}) = \mathbf{0}$, $t \in [0,1]$, $\gamma \in \mathbb{C}$,
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.

Newton's method is the most computationally intensive stage:

1. Evaluation and differentiation of all polynomials in the system.
2. Solve a linear system for the update to the approximate solution.

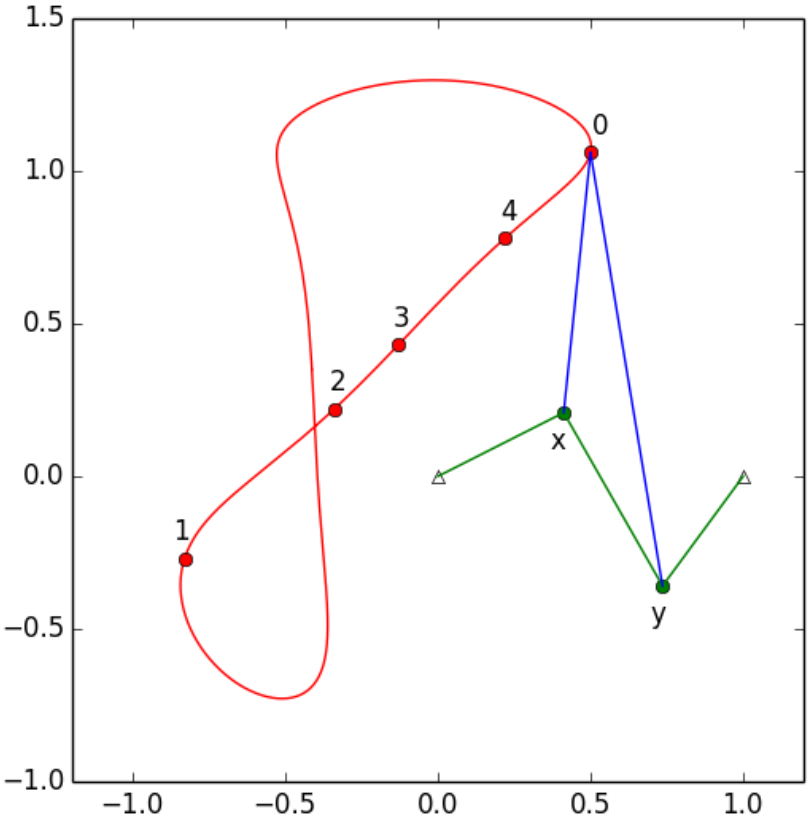
Bootstrapping to solve a start system $\mathbf{g}(\mathbf{x}) = \mathbf{0}$:

- Random coefficients of $\mathbf{g}$ imply that all solutions are regular.
- Polyhedral homotopies deform $\mathbf{g}$ to 2-nomial systems.

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.

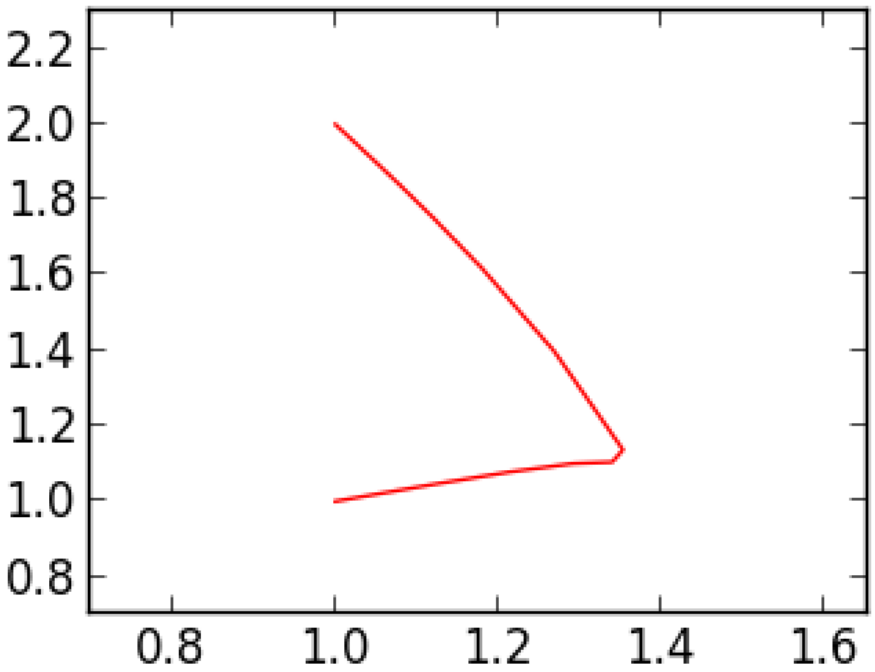
Solving Polynomial Systems with PHCpack and phcpy

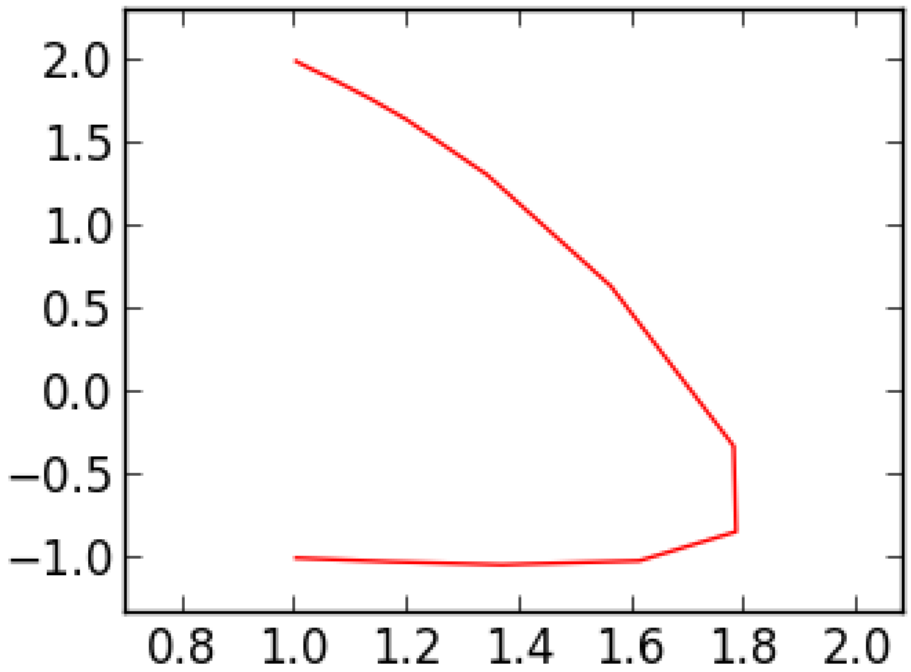
1. Introduction
   - PHCpack and phcpy
   - project history of phcpy
   - design, implementation, documentation
2. Solving Polynomial Systems
   - an example from mechanical design
   - step-by-step solution path tracking
3. Cloud Computing
   - a first web interface: CGI scripting
   - a second web interface: JupyterHub
   - conclusions and future developments
4. Short Review
   - research software

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.



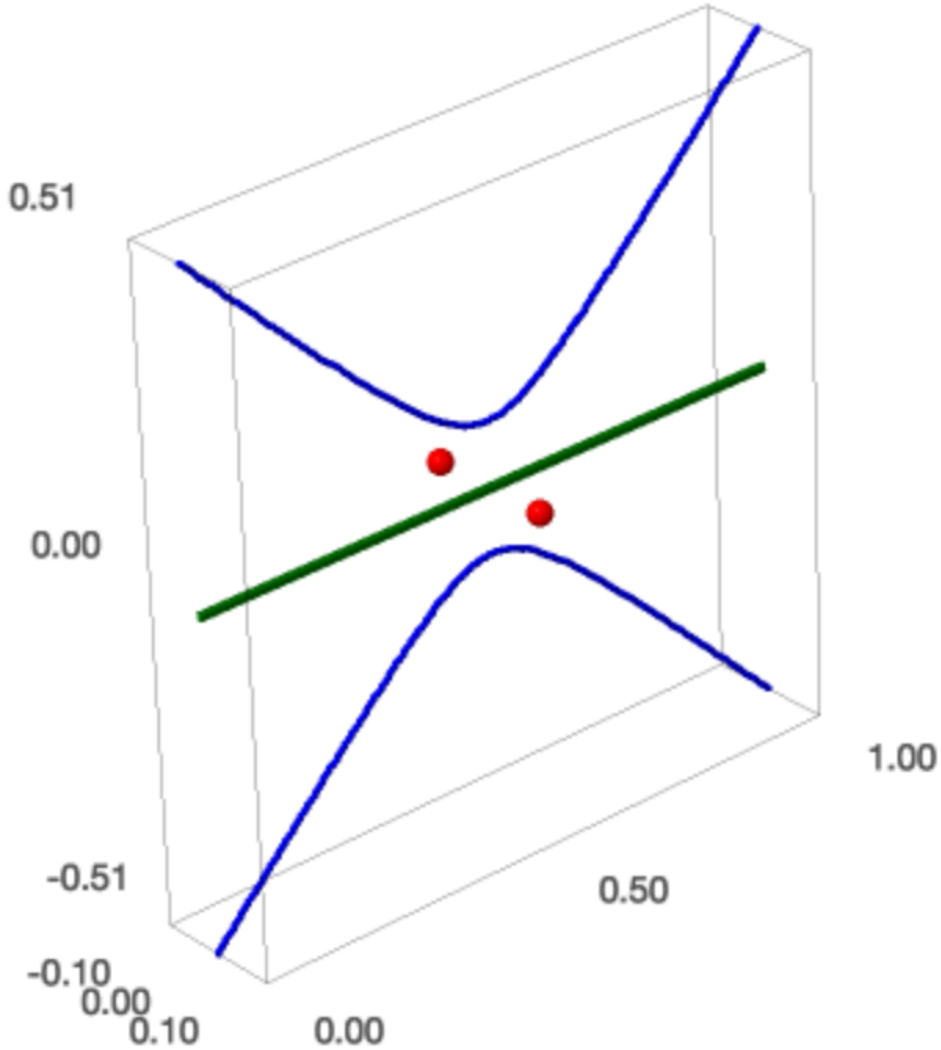


And now we have a research question: why do paths have this shape?

nearby singularities for complex values of t

Consider $x^2 - (t - 1/2)^2 - p^2 = 0$, for $p > 0$,

$$x(t) = \pm\sqrt{4p^2 + 4t^2 - 4t + 1} = 0 \Rightarrow t = 1/2 \pm p\sqrt{-1}$$



detecting nearby singularities

Applying the ratio theorem of Fabry, we can detect singular points based on the coefficients of the Taylor series.

Theorem (the ratio theorem, Fabry 1896)

If for the series $x(t) = c_0 + c_1 t + c_2 t^2 + \cdots + c_n t^n + c_{n+1} t^{n+1} + \cdots$,
we have $\lim_{n \to \infty} c_n / c_{n+1} = z$, *then*

- z *is a singular point of the series, and*
- *it lies on the boundary of the circle of convergence of the series.*

Then the radius of this circle is less than $|z|$.

L. Leau in 1899 comments on the proof (Bieberbach, 1955, page 51): "*d'habiles calculs malheureusement assez complexes.*"

A proof is in the book
The Taylor Series, an introduction to the theory of functions of a complex variable, by Paul Dienes, Dover Publications, 1957.

the ratio theorem of Fabry and Padé approximants

Consider $n = 3$, $x(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4$.

$$[3/1]_x = \frac{a_0 + a_1 t + a_2 t^2 + a_3 t^3}{1 + b_1 t}$$

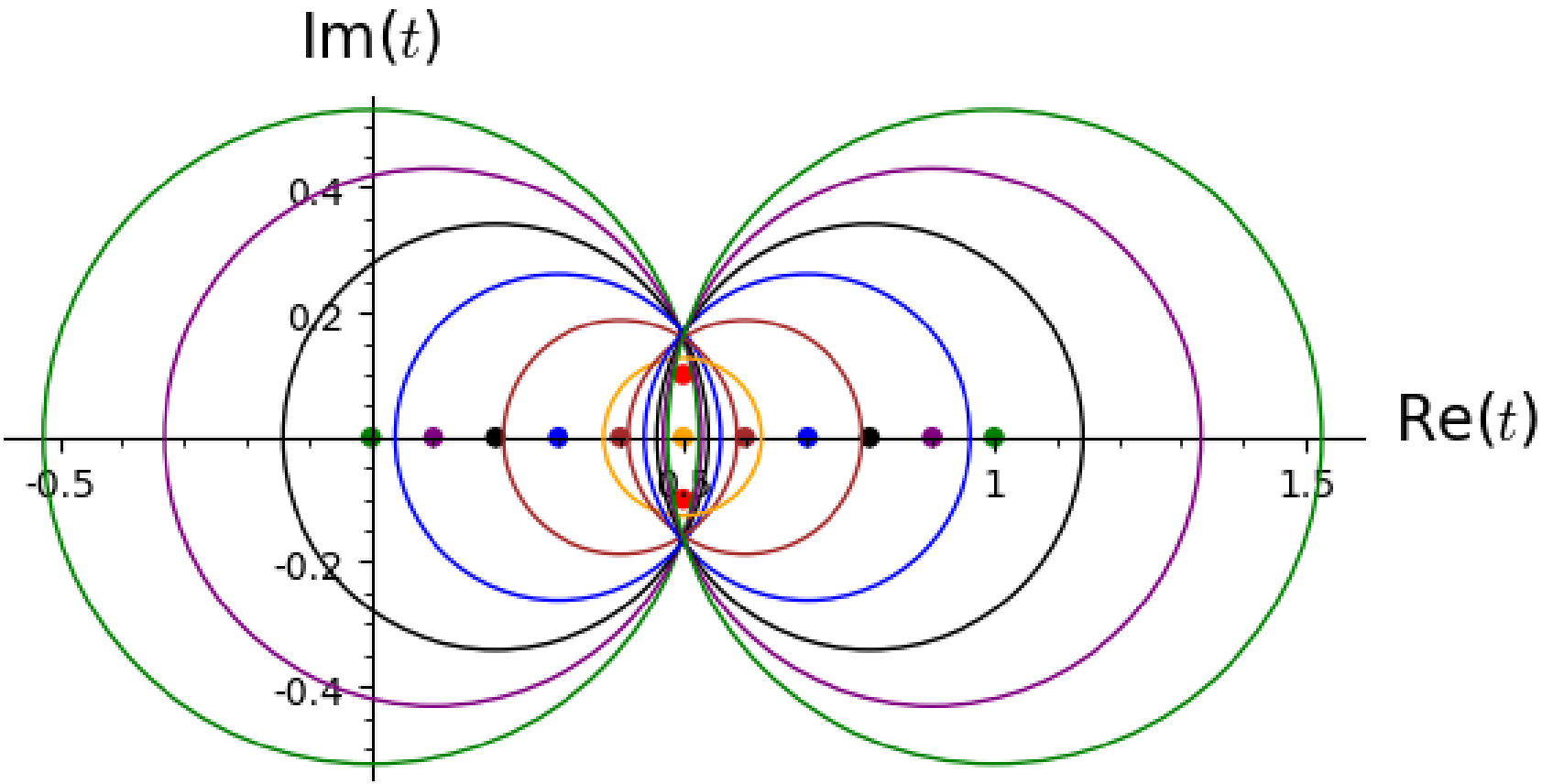
$$\begin{aligned}
(c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4)(1 + b_1 t) &= a_0 + a_1 t + a_2 t^2 + a_3 t^3 \\
c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 & \\
+ b_1 c_0 t + b_1 c_1 t^2 + b_1 c_2 t^3 + b_1 c_3 t^4 &= a_0 + a_1 t + a_2 t^2 + a_3 t^3
\end{aligned}$$

We solve for b_1 in the term for t^4: $c_4 + b_1 c_3 = 0 \Rightarrow b_1 = -c_4/c_3$.

The denominator of $[3/1]_x$ is $1 - c_4/c_3 t$. The pole of $[3/1]_x$ is c_3/c_4.

approaching and leaving a nearby singularity

Circles centered at $t = 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0$ have as radius the computed distance to the closest pole.



The singular points at $0.5 \pm 0.1I$ are represented by the red dots.

Solving Polynomial Systems with PHCpack and phcpy

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
4. Short Review
   - research software

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. `MySLQdb` 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.

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

$$\begin{cases} x^2 + xy^2 - 3 = 0 \\ 2x^2y + 5 = 0 \end{cases} \quad \text{and} \quad \begin{cases} 3 + 2ab^2 = 0 \\ b^2 - 5 + 2a^2b = 0 \end{cases}$$

are isomorphic to each other. There support sets are

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

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 graph isomorphism problem

Definition

The *graph isomorphism problem* asks whether for two undirected graphs F, G there is a bijection ϕ 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).

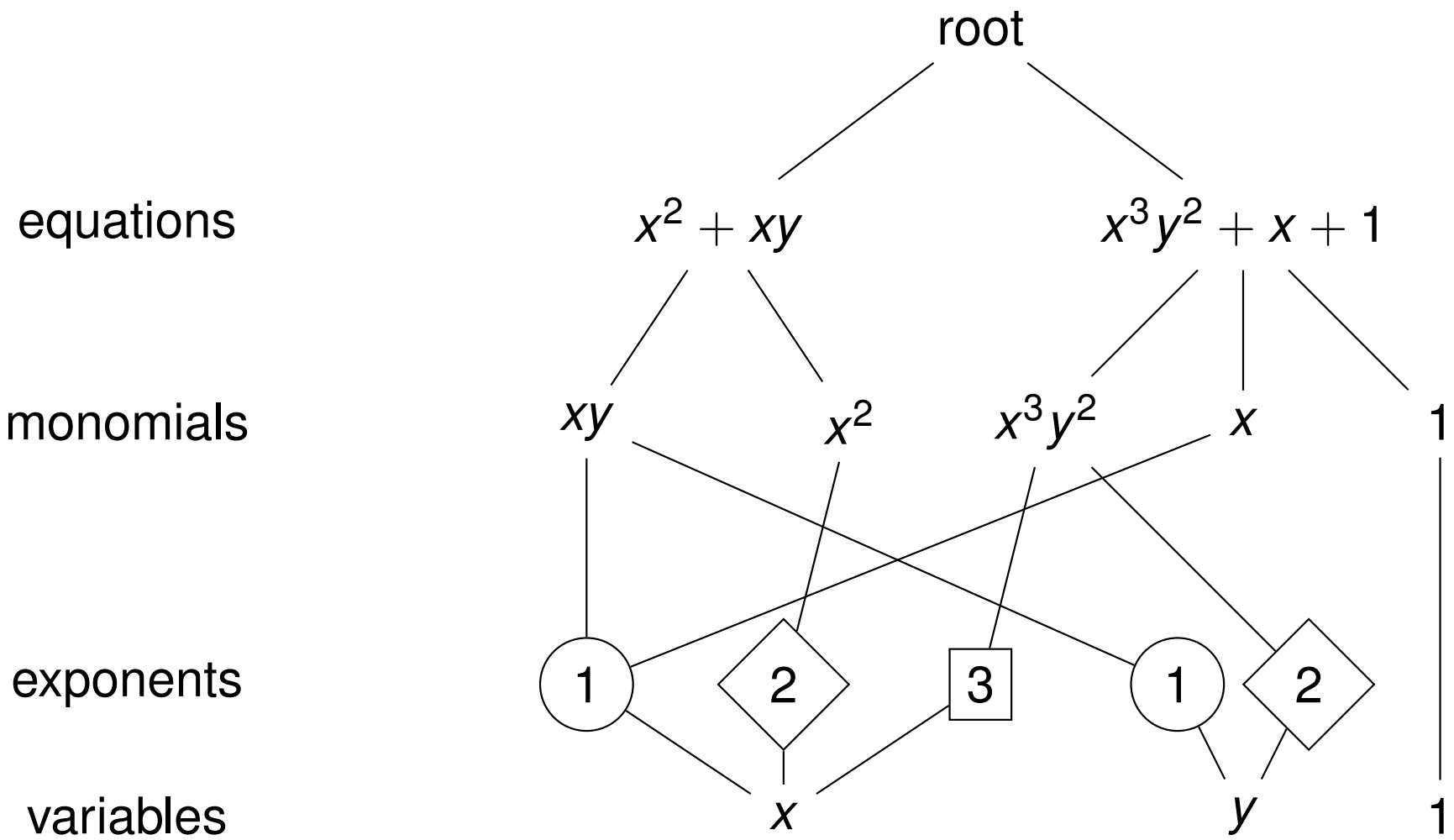
Proposition

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**.

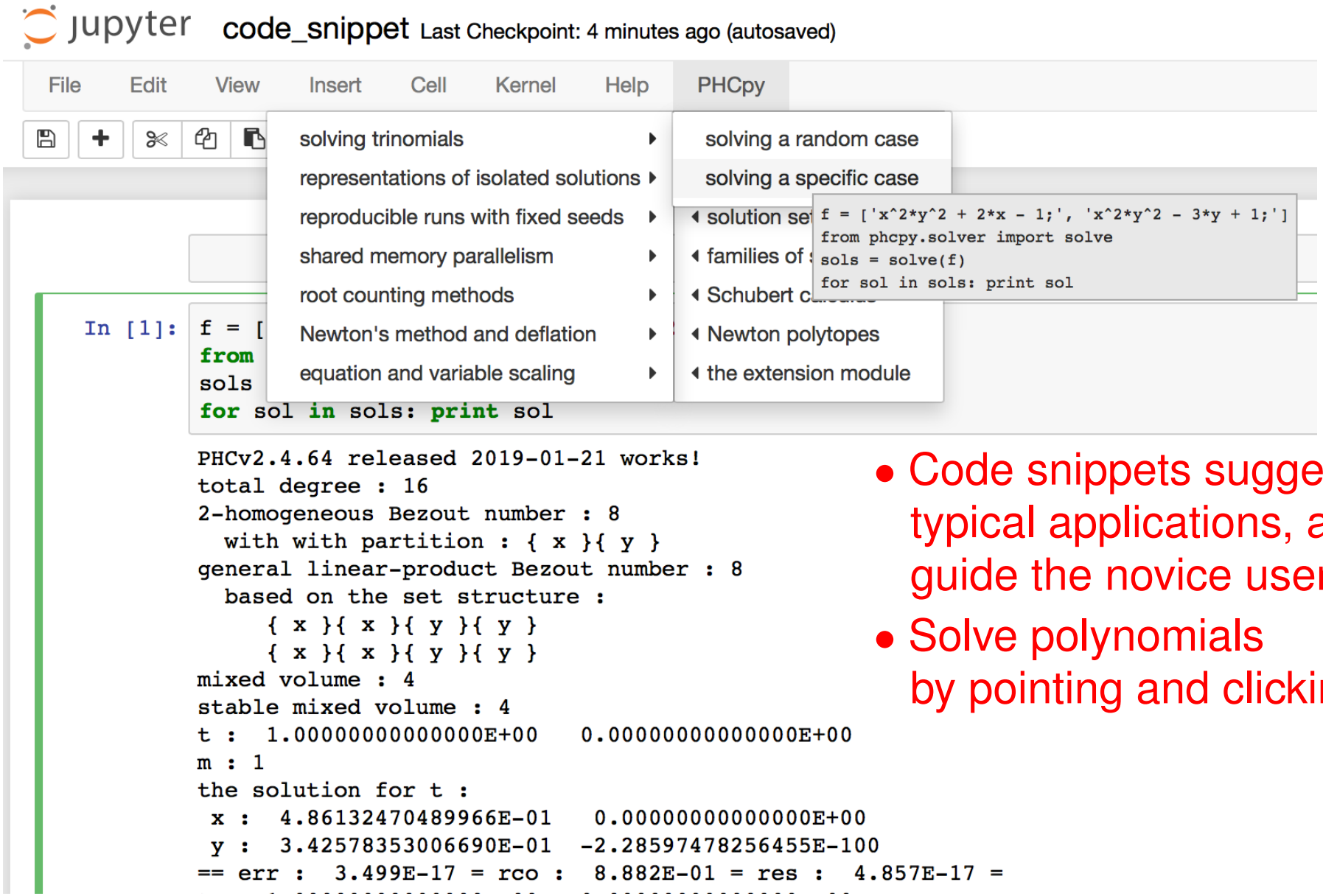
B.D. McKay and A. Piperno. **Practical graph isomorphism, II.** *Journal of Symbolic Computation*, 60:94–112, 2014.

canonical graph labelings with nauty



Solving Polynomial Systems with PHCpack and phcpy

phcpy in a SageMath kernel of a Jupyter notebook



- 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.

Solving Polynomial Systems with PHCpack and phcpy

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
4. Short Review
   - research software

conclusions and future developments

The current version of phcpy is 1.1.1.

- 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:

- Interface with Julia.
- Interactive pipelined parallel computations.
- Acceleration with Graphics Processing Units.

Solving Polynomial Systems with PHCpack and phcpy

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
4. Short Review
   - research software

research software

The description of MCS 507: The design, analysis, and use of mathematical, statistical, and scientific software.

- Python and Julia have computational ecosystems.
- SageMath and Oscar bundle software systems.
- Most software systems run in the cloud.

As a topic for research:

1. Software allows a practical study of algorithms.
2. New algorithms are best developed with software.