Solving Polynomial Systems with *phcpy*

Jasmine T. Otto[*], Angus G. Forbes[*], Jan Verschelde[†]
jtotto@ucsc.edu angus@ucsc.edu janv@uic.edu
[*] University of California Santa Cruz, [†] University of Illinois at Chicago

PHCpack is a package for **polynomial homotopy continuation** to solve polynomial systems. *ACM Transactions on Mathematical Software* has archived version 1.0 as Algorithm 795, vol. 25, no. 2, pages 251--276, 1999.

Two **blackbox solvers** in phcpy can compute:
1) the isolated solutions of a polynomial system.
2) a numerical irreducible decomposition, i.e., all *solution sets* of the system.

phcpy exposes the functionality of PHCpack to Python scripts. Computationally intensive algorithms are executed efficiently by the compiled code.

DISCUSSION

Usage of *phcpy* has been reported in research literature:

Symbolic computation – # embeddings of minimally rigid graphs (Bartzos, Emiris, Legersky, and Tsigaridas, 2018)

Pure math – Roots of Alexander polynomials (Culler and Dunfield, 2018)

Chemical engineering – Critical points of equilibrium problems (Sidky, Whitmer, and Mehta, 2016)

Numerical algebraic geometry is currently used in other fields whose models rely on nonlinear polynomial systems:

Rigid-body mechanisms – Algebraic kinematics for synthesis and control of motion, e.g. in robotics or animation. (Wampler and Sommese, 2011)

Systems biology – Model selection via analysis of steady states, e.g. in pathway analysis or disease modeling. (Gross, Davis, Ho, Bates, and Harrington, 2016)

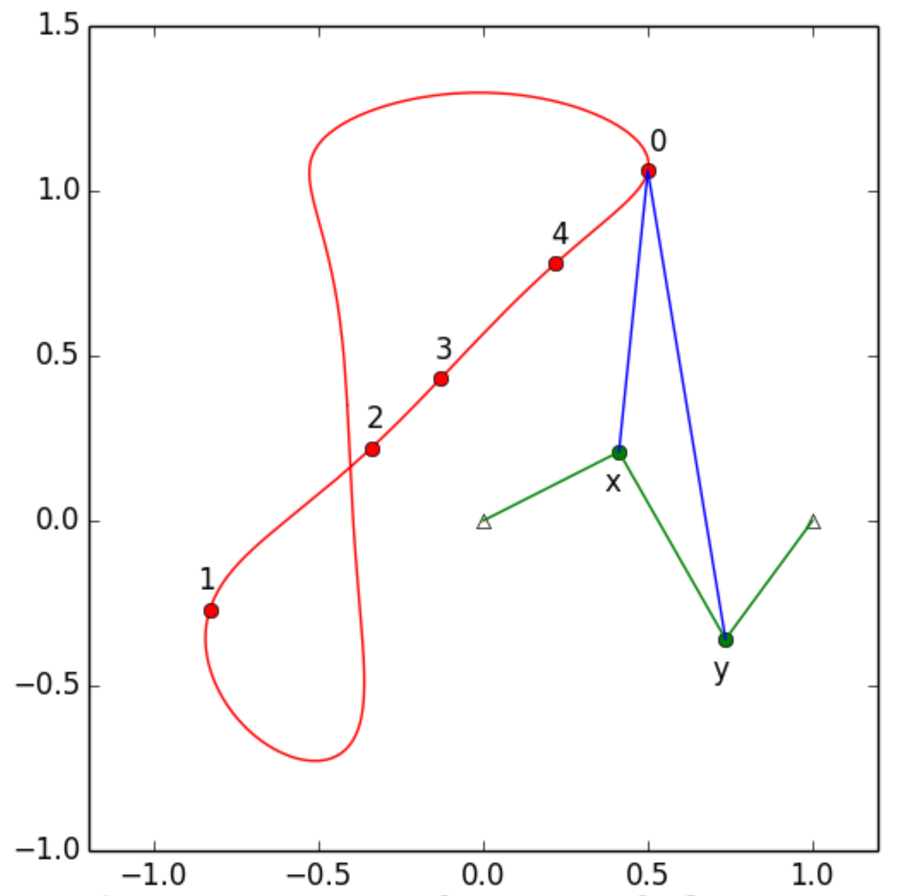
We can numerically solve polynomial systems in *phcpy*, using fast and reliable homotopy continuation methods.

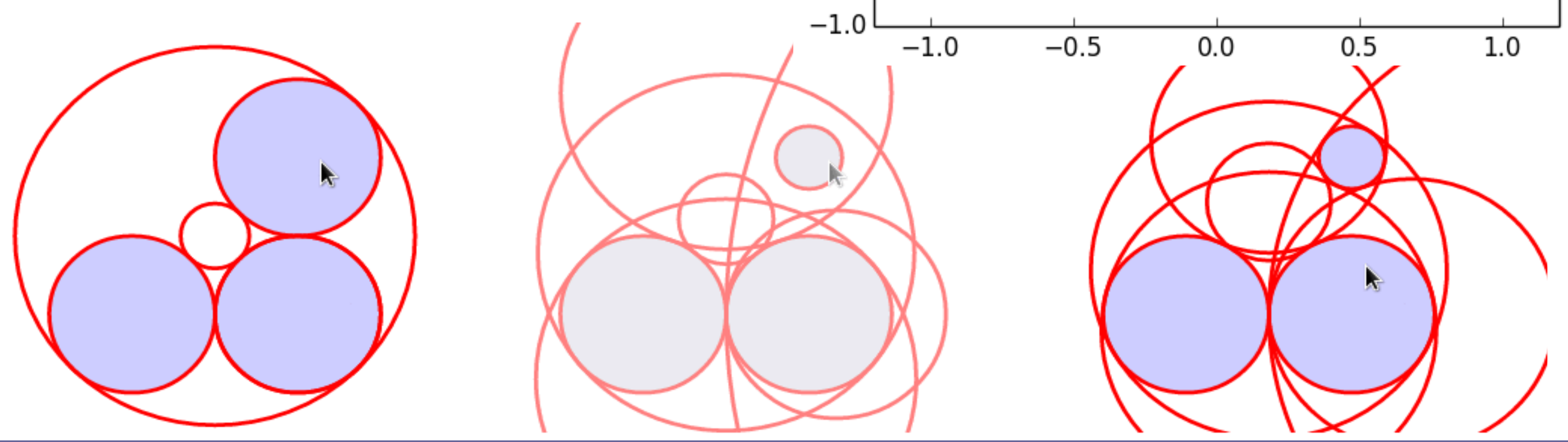
TUTORIALS math.uic.edu/~jan/phcpy_doc_html/usecases.html

(right) Given 5 precision points, design a 4-bar mechanism.

(below) Compute all circles that touch three given circles.

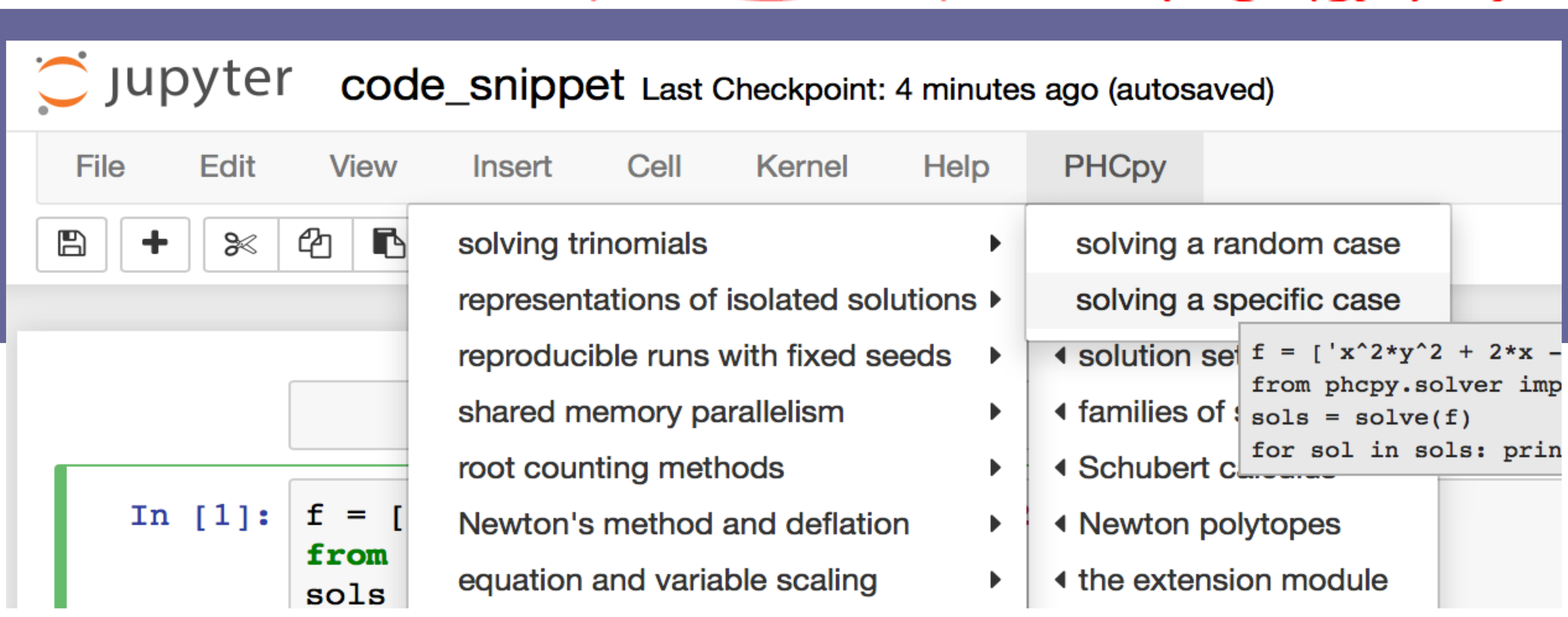
The *phcpy* documentation includes these tutorials. Further use cases are documented in our paper, conference.scipy.org/proceedings/scipy2019/pdfs/jan_verschelde.pdf





A web interface to phcpy is online at www.phcpack.org

, a notebook with Python and SageMath kernels, available for public use with a free account.



INTERACTIVE PARALLELISM

```
def qualityup(nbtasks=0, precflag='d'):
    from phcpy.families import cyclic
    from phcpy.solver import solve
    from time import perf_counter
    c7 = cyclic(7)
    tstart = perf_counter()
    s = solve(c7, verbose=False, tasks=nbtasks,\
        precision=precflag, checkin=False)
    return perf_counter() - tstart
```

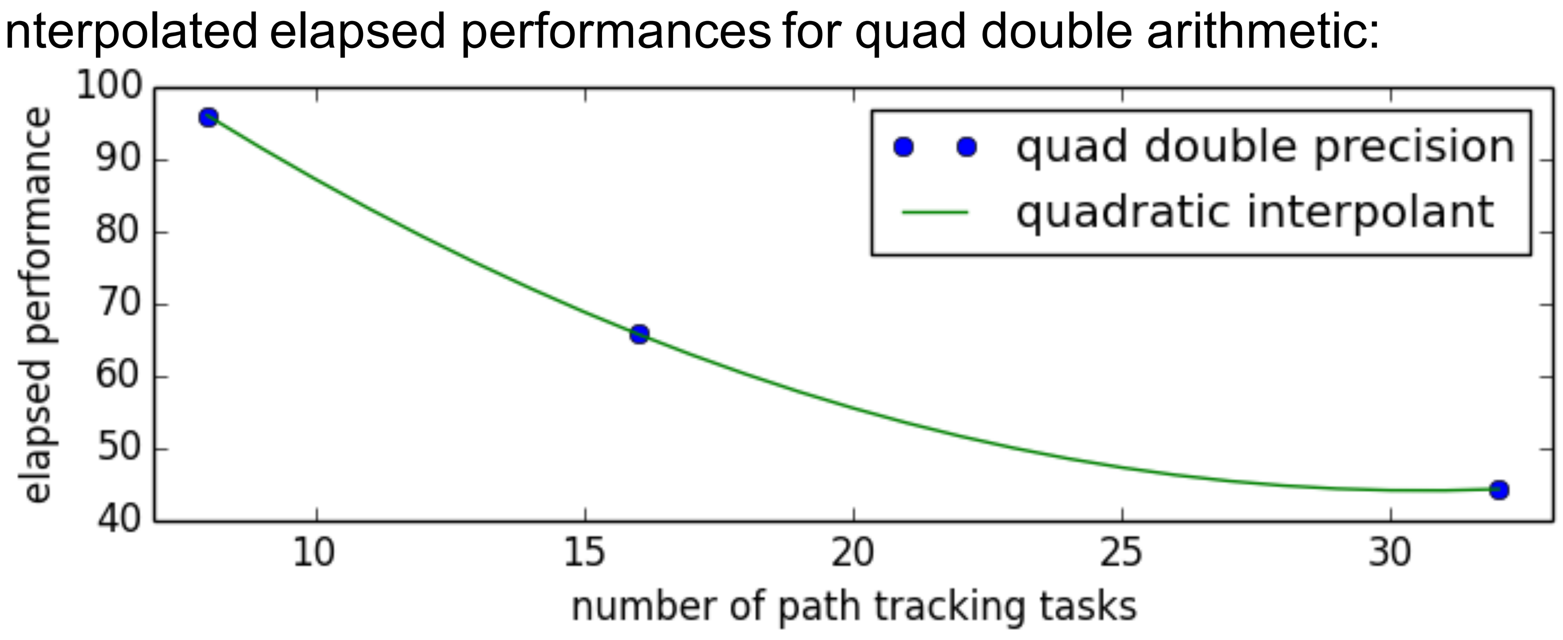
The blackbox solver's elapsed performance in double, double double, and quad double precision:

precision	d	dd	qd
Elapsed performance	5.45	**42.41**	604.91
Overhead factor	1.00	7.41	110.99

tasks	8	16	32
dd	7.56	5.07	3.88
qd	96.08	65.82	**44.35**

The overhead of extra precision is compensated by multithreading.

Interpolated elapsed performances for quad double arithmetic:



SOLUTION SETS

A numerical irreducible decomposition includes **representations** for all positive dimensional solution sets.

Consider two equations defining the *twisted cubic*:

pols = ['x*y - z;', 'x^2 – y;']

(1) A *witness set* provides generic points:

```
from phcpy.sets import embed
from phcpy.solver import solve
embp = embed(3, 1, pols)
sols = solve(embp, verbose=False)
print('#generic points :', len(sols))
```

Three points lie at the intersection of this cubic with a random plane.

(2) A *series expansion* develops from some point(s) in a coordinate hyperplane. For the twisted cubic, the series is exact after the first term.

```
from phcpy.maps import solve_binomials
maps = solve_binomials(3, pols, puretopdim=True)
for sol in maps: print(sol)
```

The solution gives (t,t^2,t^3), the parametric representation of the twisted cubic.

Baskin Engineering UC SANTA CRUZ
THE UNIVERSITY OF ILLINOIS AT CHICAGO
UIC

This material is based on work supported by the National Science Foundation under Grant 1440534.