# Multitasking Polynomial Homotopy Continuation in PHCpack

Jan Verschelde

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

ACA 2009 Session on High-Performance Computer Algebra
ÉTS, Montréal, Canada, 25-28 June 2009.

# Solving Polynomial Systems

On input is a polynomial system $f(\mathbf{x}) = \mathbf{0}$.

A homotopy is a family of systems:

$$h(\mathbf{x}, t) = (1 - t)g(\mathbf{x}) + t\, f(\mathbf{x}) = \mathbf{0}.$$

At $t = 1$, we have the system $f(\mathbf{x}) = \mathbf{0}$ we want to solve.
At $t = 0$, we have a good system $g(\mathbf{x}) = \mathbf{0}$:

- solutions are known or easier to solve; and
- all solutions of $g(\mathbf{x}) = \mathbf{0}$ are regular.

Tracking all solution paths is pleasingly parallel,
although not every path requires the same amount of work.

# Homotopy Continuation Methods

Types of homotopies $h$:

- $h(\mathbf{x}, t) = (1-t)g(\mathbf{x}) + t\, f(\mathbf{x}) = \mathbf{0}$, from start to target.
- $h(\mathbf{x}, t) = f(\mathbf{c}_0(1-t) + \mathbf{c}_1 t, \mathbf{x}) = \mathbf{0}$, cheater's homotopy.
- $h(\mathbf{x}, t) = f(\mathbf{c}, \mathbf{x}(t)) = \mathbf{0}$, moving basis coordinates.

Homotopies are often used in combination, or in cascades.


1. **Tien-Yien Li.** Numerical solution of polynomial systems by homotopy continuation methods. In Volume XI of *Handbook of Numerical Analysis*, pages 209–304, 2003.
2. **Andrew J. Sommese and Charles W. Wampler.** *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science.* World Scientific, 2005.

# Software Systems

Starring in alphabetical order:

- **Bertini**, first released in Fall 2006, by D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler. MPI executables available.
- **HOM4PS-2.0para** by T.Y. Li and C.H. Tsai (2009) is a parallel version of **HOM4PS-2.0** by T.L. Lee, T.Y. Li, and C.H. Tsai (2007); extends **HOM4PS** by T. Gao and T.Y. Li.
- **PHoMpara** by T. Gunji, S. Kim, K. Fujisawa, and M. Kojima (2006) is a parallel version of **PHoM** by T. Gunji, S. Kim, M. Kojima, A. Takeda, K. Fujisawa and T. Mizutani (2004).
- **POLSYS_GLP** is Algorithm 857 of ACM TOMS (2006) by H.-J. Su, J.M. McCarthy, M. Sosonkina, and L.T. Watson extends **HOMPACK90** by L.T. Watson, M. Sosonkina, R.C. Melville, A.P. Morgan, and H.F. Walker (1997) and **HOMPACK** by L.T. Watson, S.C. Billups, and A.P. Morgan (1987).

Anton Leykin is developing homotopy continuation in Macaulay2.
`http://www.math.uic.edu/~leykin/NAG4M2/index.html`.

# Parallel PHCpack

## parallel implementation of polynomial homotopy continuation methods

PHC = Polynomial Homotopy Continuation

- Version 1.0 archived as Algorithm 795 by ACM TOMS (1999)
- Pleasingly parallel implementations
  - + **Yusong Wang** of Pieri homotopies (HPSEC'04)
  - + **Anton Leykin** of monodromy factorization (HPSEC'05)
  - + **Yan Zhuang** of polyhedral homotopies (HPSEC'06)
  - + **Yun Guan** of diagonal homotopies (HPCS'08)
- Interactive Parallel Computing:
  - + **Yun Guan:** PHClab, experiments with MPITB in Octave
  - + **Kathy Piret:** bindings with Python, use of sockets

Release v2.3.42 extends **phcpy** and a preliminary **PHCwulf.py**.

# Hardware and Software

Running on a modern workstation (*not* a supercomputer):

- Hardware: Mac Pro with 2 Quad-Core Intel Xeons at 3.2 Ghz
  Total Number of Cores: 8 1.6 GHz Bus Speed
  12 MB L2 Cache per processor, 8 GB Memory
- PHCpack is written in Ada, compiled with gnu-ada compiler
  gcc version 4.3.4 20090511 for GNAT GPL 2009 (20090511)
  Target: x86_64-apple-darwin9.6.0
  Thread model: posix

Also compiled for Linux and Windows (win32 thread model).

# Starting Worker Tasks

procedure `Workers` is instantiated with a `Job` procedure, executing code based on the `id` number.

```
procedure Workers ( n : in natural ) is
   task type Worker ( id,n : natural );
   task body Worker is
   begin
     Job(id,n);
   end Worker;
   procedure Launch_Workers ( i,n : in natural ) is
     w : Worker(i,n);
   begin
     if i < n
      then Launch_Workers(i+1,n);
     end if;
   end Launch_Workers;
begin
  Launch_Workers(1,n);
end Workers;
```

# MPI versus Threads

- MPI = Message Passing Interface

  The manager/worker paradigm:

  - worker nodes perform path tracking jobs,
  - manager maintains job queue, serves workers.

  Manager must be available to serve jobs.

- Threads are lightweight processes

  Collaborative workers launched by master thread:

  - communication overhead replaced by memory sharing,
  - job queue updated in critical section using locks.

- With MPI, we worry about communication overhead.
  With threads, memory (de)allocation must be in critical sections.

# Load Balancing and Granularity Issues

We assume: # solution paths $\gg$ # cores.

Granularity Issues:

- coarse: one job = track one solution path
- fine: polynomial evaluation, linear algebra

Dynamic load balancing:

- not all jobs take the same amount of work

# An academic Benchmark: cyclic $n$-roots

The system

$$f(\mathbf{x}) = \begin{cases} f_i = \displaystyle\sum_{j=0}^{n-1} \prod_{k=1}^{i} x_{(k+j) \bmod n} = 0, \quad i = 1, 2, \ldots, n-1 \\ f_n = x_0 x_1 x_2 \cdots x_{n-1} - 1 = 0 \end{cases}$$

appeared in

- G. Björck: **Functions of modulus one on $Z_p$ whose Fourier transforms have constant modulus.** In *Proceedings of the Alfred Haar Memorial Conference, Budapest*, pages 193–197, 1985.
- J. Backelin and R. Fröberg: **How we proved that there are exactly 924 cyclic 7-roots.** In ISSAC'91 proceedings, pages 101-111, ACM, 1991.

very sparse, well suited for polyhedral methods

# First Preliminary Results

Using version 2.3.45 of PHCpack:

```
$ time phc -p -t8 < /tmp/input8
```

#worker tasks = number following the `-t`

running a cheater's homotopy on cyclic 7-roots (924 paths).

| #workers | real | user | sys | speedup |
|---|---|---|---|---|
| 1 | 15.478s | 15.457s | 0.010s | 1 |
| 2 | 7.790s | 15.483s | 0.010s | 1.987 |
| 4 | 3.926s | 15.445s | 0.011s | 3.942 |
| 8 | 1.992s | 15.424s | 0.015s | 7.770 |

Since version 2.3.46 of PHCpack:

```
$ phc -b -t8
```

blackbox solver (`phc -b`) uses multitasking

# 3 stages to solve a polynomial system $f(\mathbf{x}) = \mathbf{0}$

1. Compute the mixed volume (aka the BKK bound) of the Newton polytopes spanned by the supports $A$ of $f$ via a **regular mixed-cell configuration** $\Delta_\omega$.
2. Given $\Delta_\omega$, solve a generic system $g(\mathbf{x}) = \mathbf{0}$, using polyhedral homotopies. Every cell $C \in \Delta_\omega$ defines one homotopy

$$h_C(\mathbf{x}, s) = \sum_{\mathbf{a} \in C} c_{\mathbf{a}} \mathbf{x}^{\mathbf{a}} + \sum_{\mathbf{a} \in A \setminus C} c_{\mathbf{a}} \mathbf{x}^{\mathbf{a}} s^{\nu_{\mathbf{a}}}, \quad \nu_{\mathbf{a}} > 0,$$

   tracking as many paths as the mixed volume of the cell $C$, as $s$ goes from 0 to 1.
3. Use $(1 - t)g(\mathbf{x}) + tf(\mathbf{x}) = \mathbf{0}$ to solve $f(\mathbf{x}) = \mathbf{0}$.

Stages 2 and 3 are **computationally most intensive** $(1 \ll 2 < 3)$.

# A Static Distribution of the Workload

## used in `mpi2cell_s` with Yan Zhuang

| manager | worker 1 | worker 2 | worker 3 |
|---|---|---|---|
| Vol(cell 1) $=$ 5 | #paths(cell 1) : 5 | | |
| Vol(cell 2) $=$ 4 | #paths(cell 2) : 4 | | |
| Vol(cell 3) $=$ 4 | #paths(cell 3) : 4 | | |
| Vol(cell 4) $=$ 6 | #paths(cell 4) : 1 | #paths(cell 4) : 5 | |
| Vol(cell 5) $=$ 7 | | #paths(cell 5) : 7 | |
| Vol(cell 6) $=$ 3 | | #paths(cell 6) : 2 | #paths(cell 6) : 1 |
| Vol(cell 7) $=$ 4 | | | #paths(cell 7) : 4 |
| Vol(cell 8) $=$ 8 | | | #paths(cell 8) : 8 |
| total #paths : 41 | #paths : 14 | #paths : 14 | #paths : 13 |

Since polyhedral homotopies solve a **generic** system $g(\mathbf{x}) = \mathbf{0}$,
we **expect** every path to take the same amount of work...

# Running Polyhedral Homotopies

Running polyhedral homotopies on a random coefficient system, distributing mixed cells, for the cyclic *n*-roots problems.

Tracking MV (MV = mixed volume) many solution paths:

| | | #tasks, times in seconds | | | |
|---|---|---|---|---|---|
| n | MV | 1 | 2 | 4 | 8 |
| 7 | 924 | 12 | 6 | 3 | 2 |
| 8 | 2560 | 58 | 29 | 15 | 8 |
| 9 | 11016 | 417 | 209 | 104 | 52 |
| 10 | 35940 | 2156 | 1068 | 534 | 270 |

Comparison with MPI (`mpi2cell_d`) on cyclic 10-roots:

- `mpirun -n 9`: total wall time = 270.5 seconds.
- on same random coefficient system and same tolerances: elapsed wall clock time is 233 seconds.

# Conclusions and Future Plans

Conclusions:

- multitasking is convenient programming model
- effective speedups on modern workstation

Some future applications:

- multi-tiered parallel implementations
- investigation of granularity issues
- attention for quality up (not only speedup)
- parallel versions of Littlewood-Richardson homotopies