Parallel Homotopy Algorithms to Solve Polynomial Systems

Jan Verschelde

Department of Math, Stat & CS
University of Illinois at Chicago
Chicago, IL 60607-7045, USA

eMail: `jan@math.uic.edu`

URL: `http://www.math.uic.edu/~jan`

Joint work with Anton Leykin and Yan Zhuang.

AMS session on Numerical Solving of Polynomial Systems

University of Notre Dame, 8-9 April 2006.

Motivation: solve large systems

Currently, large means > 100,000 solution paths are needed to solve $f(\mathbf{x}) = \mathbf{0}$ using start system $g(\mathbf{x}) = \mathbf{0}$, defined by a typical homotopy $h(\mathbf{x}, t) = \mathbf{0}$:

$$h(\mathbf{x}, t) = \gamma(1 - t)g(\mathbf{x}) + tf(\mathbf{x}) = \mathbf{0}, \quad \gamma \in \mathbb{C},$$

as t goes from 0 to 1.

- efficiency: undesirable to keep all solutions in main memory;
- numerical instabilities may occur as dimensions grow;
- quality control on the computed solutions.

Although “large” does not always automatically imply “difficult”, size matters.

Other Parallel Homotopy Solvers

T. Gunji, S. Kim, K. Fujisawa, and M. Kojima:
PHoMpara – parallel implementation of the Polyhedral Homotopy continuation Method for polynomial systems. Research report b-419, Tokyo Institute of Technology, 2005.

H.-J. Su, J.M. McCarthy, M. Sosonkina, and L.T. Watson:
Algorithm 8xx: **POLSYS_GLP**: A parallel general linear product homotopy code for solving polynomial systems of equations. To appear in *ACM Trans. Math. Softw.*

parallel PHCpack

initiated with Yusong Wang

- static and dynamic load balancing for cheater's homotopy and coefficient-parameter polynomial continuation;
- Pieri homotopies are well suitable despite their tree structure;
- main program in C calls MPI and phc: "mpi2phc";
- good speedup for existing benchmark problems.

Distributing all path tracking jobs at the start performs well
when all paths require same amount of work,
otherwise **dynamic load balancing** is needed for best performance.

MPI = Message Passing Interface, a collection of library routines to exchange data between nodes of a multicomputer.

current progress in PHCpack

with Anton Leykin: parallel monodromy breakup

(HPSEC'05, MAGIC'05, to appear in IJCSE)

with Yan Zhuang: parallel polyhedral homotopies

(see talk later in this session)

An ambitious Swap of Letters:

PHC = Polynomial Homotopy Continuation

HPC = High Performance Computing

towards High Performance *Continuation*

New Features in PHCpack (v2.3.07)

+ efficiency: jumpstarting homotopies

+ reliability: numerically stable fewnomial solver

+ quality control: scanning solution lists into frequency tables

internal: library between MPI main program and Ada core

Jumpstarting Homotopies

Problem: huge #paths (e.g.: $> 100{,}000$),

undesirable to store all start solutions in main memory.

Solution: (assume manager/worker protocol)

1. The manager reads start solution from file "just in time" whenever a worker needs another path tracking job.
2. For total degree and linear-product start systems, it is simple the compute the solutions whenever needed.
3. As soon as worker reports the end of a solution path back to the manager, the solution is written to file.

Indexing Start Solutions

The start system $\left\{ \begin{array}{l} x_1^4 - 1 = 0 \\ x_2^5 - 1 = 0 \\ x_3^3 - 1 = 0 \end{array} \right.$ has $4 \times 5 \times 3 = 60$ solutions.

Get 25th solution via decomposition: $24 = 1(5 \times 3) + 3(3) + 0$.

Verify via lexicographic enumeration:

$$000 \rightarrow 001 \rightarrow 002 \rightarrow 010 \rightarrow 011 \rightarrow 012 \rightarrow 020 \rightarrow 021 \rightarrow 022 \rightarrow 030 \rightarrow 031 \rightarrow 032 \rightarrow 040 \rightarrow 041 \rightarrow 042$$

$$100 \rightarrow 101 \rightarrow 102 \rightarrow 110 \rightarrow 111 \rightarrow 112 \rightarrow 120 \rightarrow 121 \rightarrow 122 \rightarrow \boxed{130} \rightarrow 131 \rightarrow 132 \rightarrow 140 \rightarrow 141 \rightarrow 142$$

$$200 \rightarrow 201 \rightarrow 202 \rightarrow 210 \rightarrow 211 \rightarrow 212 \rightarrow 220 \rightarrow 221 \rightarrow 222 \rightarrow 230 \rightarrow 231 \rightarrow 232 \rightarrow 240 \rightarrow 241 \rightarrow 242$$

$$300 \rightarrow 301 \rightarrow 302 \rightarrow 310 \rightarrow 311 \rightarrow 312 \rightarrow 320 \rightarrow 321 \rightarrow 322 \rightarrow 330 \rightarrow 331 \rightarrow 332 \rightarrow 340 \rightarrow 341 \rightarrow 342$$

Using Linear-Product Start Systems Efficiently

- Store start systems in their linear-product product form, e.g.:

$$g(\mathbf{x}) = \left\{ \begin{array}{c} (\cdots)\cdot(\cdots)\cdot(\cdots)\cdot(\cdots) = 0 \\ (\cdots)\cdot(\cdots)\cdot(\cdots)\cdot(\cdots)\cdot(\cdots) = 0 \\ (\cdots)\cdot(\cdots)\cdot(\cdots) = 0 \end{array} \right.$$

- Lexicographic enumeration of start solutions,

  $\rightarrow$ as many candidates as the total degree.

- Eventually store results of incremental LU factorization.

  $\rightarrow$ prune in the tree of combinations.

A well conditioned polynomial system

just one of the 11,417 start systems generated by polyhedral homotopies

12 equations, 13 distinct monomials (after division):

$$\left\{ \begin{array}{c} b_1 x_5 x_8 + b_2 x_6 x_9 = 0 \\ b_3 x_2^2 + b_4 = 0 \\ b_5 x_1 x_4 + b_6 x_2 x_5 = 0 \\ c_1^{(k)} x_1 x_4 x_7 x_{12} + c_2^{(k)} x_1 x_6 x_{10}^2 + c_3^{(k)} x_2 x_4 x_8 x_{10} + c_4^{(k)} x_2 x_4 x_{11}^2 \\ + \, c_5^{(k)} x_2 x_6 x_8 x_{11} + c_6^{(k)} x_3 x_4 x_9 x_{10} + c_7^{(k)} x_4^2 x_{12}^2 + c_8^{(k)} x_3 x_6 \\ + \, c_9^{(k)} x_4^2 + c_{10}^{(k)} x_9 = 0, \quad k = 1, 2, \ldots, 9 \end{array} \right.$$

Random coefficients chosen on the complex unit circle.

Despite the high degrees, only 100 well conditioned solutions.

Solve a “binomial” system $\mathbf{x}^A = \mathbf{b}$ via Hermite

Hermite normal form of A: $MA = U$, $\det(M) = \pm 1$,
U is upper triangular, $|\det(U)| = |\det(A)| = \#$solutions.

Let $\mathbf{x} = \mathbf{z}^M$, then $\mathbf{x}^A = \mathbf{z}^{MA} = \mathbf{z}^U$, so solve $\mathbf{z}^U = \mathbf{b}$.

$n = 2$:

$$[z_1 \quad z_2] \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix} = [b_1 \quad b_2].$$

$$\left\{ \begin{array}{rcl} z_1^{u_{11}} & = & b_1 \\ z_1^{u_{12}} \ z_2^{u_{22}} & = & b_2 \end{array} \right.$$

$|b_k| = 1 \Rightarrow |z_i| = 1$

numerically well conditioned

Reduce a "fewnomial" system $C\mathbf{x}^A = \mathbf{b}$ via LU

$$
\begin{array}{ccccc}
C = LU & & (1) & LU\mathbf{y} = \mathbf{b} & \text{linear system} \\
 & \Rightarrow & & & \\
\text{assume } \det(C) \neq 0 & & (2) & \mathbf{x}^A = \mathbf{y} & \text{binomial system}
\end{array}
$$

This is a numerically unstable algorithm!

Randomly chosen coefficients for C and $\mathbf{b}$ on complex unit circle,

but still, varying magnitudes in $\mathbf{y}$ do occur.

High powers, e.g.: 50, magnify the imbalance

$\rightarrow$ numerical underflow or overflow in binomial solver.

Separate Magnitudes from Angles

Solve $\mathbf{x}^A = \mathbf{y}$ via Hermite: $MA = U \Rightarrow \mathbf{x} = \mathbf{z}^M : \mathbf{z}^U = \mathbf{y}$.

$\mathbf{z} = |\mathbf{z}|\mathbf{e}_\mathbf{z}$, $\mathbf{e}_\mathbf{z} = \exp(i\theta_\mathbf{z})$, $\mathbf{y} = |\mathbf{y}|\mathbf{e}_\mathbf{y}$, $\mathbf{e}_\mathbf{y} = \exp(i\theta_\mathbf{y})$, $i = \sqrt{-1}$.

Solve $\mathbf{z}^U = \mathbf{y}$: $|\mathbf{z}|^U \mathbf{e}_\mathbf{z}^U = |\mathbf{y}|\mathbf{e}_\mathbf{y} \Leftrightarrow \begin{cases} \mathbf{e}_\mathbf{z}^U = \mathbf{e}_\mathbf{y} & \text{well conditioned} \\ |\mathbf{z}|^U = |\mathbf{y}| & \text{find magnitudes} \end{cases}$

To solve $|\mathbf{z}|^U = |\mathbf{y}|$, consider: $U \log(|\mathbf{z}|) = \log(|\mathbf{y}|)$.

Even as the magnitude of the values $\mathbf{y}$ may be extreme, $\log(|\mathbf{y}|)$ will be modest in size.

a numerically stable fewnomial solver

We solve $C\mathbf{x}^A = \mathbf{b}$ by

1. LU factorization of $C \rightarrow \mathbf{x}^A = \mathbf{y}$, where $C\mathbf{y} = \mathbf{b}$.
2. Use Hermite normal form of A: $MA = U$, $\det(M) = \pm 1$, to solve binomial system $\mathbf{e}_\mathbf{z}^U = \mathbf{e}_\mathbf{y}$, $\mathbf{z} = |\mathbf{z}|\mathbf{e}_\mathbf{z}$, $\mathbf{y} = |\mathbf{y}|\mathbf{e}_\mathbf{y}$.
3. Solve upper triangular linear system $U \log(|\mathbf{z}|) = \log(|\mathbf{y}|)$.
4. Compute magnitude of $\mathbf{x} = \mathbf{z}^M$ via $\log(|\mathbf{x}|) = M \log(|\mathbf{z}|)$.
5. As $|\mathbf{e}_\mathbf{z}| = 1$, let $\mathbf{e}_\mathbf{x} = \mathbf{e}_\mathbf{z}^M$.

Even as $\mathbf{z}$ may be extreme, we deal with $|\mathbf{z}|$ at a logarithmic scale and never raise small or large number to high powers.

Only at the very end do we calculate $|\mathbf{x}| = 10^{\log(|\mathbf{x}|)}$ and $\mathbf{x} = |\mathbf{x}|\mathbf{e}_\mathbf{x}$.

Quality Control: Scanning Solution Files

During runtime, we want to

1. monitor progress of a large path tracking job;
2. get an impression about the “quality” of the solutions which have been already computed;

but again, we do not want store all solutions in main memory.

Scanning Solution Files into Frequency Tables

Newton's method reports for each solution:

1. the magnitude of the last update to the solution vector;
2. an estimate for the inverse condition number of the Jacobian matrix at the solution;
3. the magnitude of the residual.

These three numbers determine the quality of a solution.

To determine the overall quality of the list of solutions,
the program builds frequency tables, e.g.: counting #solutions with condition number between 10^{k-1} and 10^k, for some range of k.

$\rightarrow$ can be done with incomplete solution lists

Conclusions

Three issues to improve performance of parallel homotopies

- Avoid storing all solutions in main memory.
- Numerical stability matters even more.
- Fast quality control of large solution lists.

Computational results:

see talk of Yan Zhuang later in the session.