Parallel Homotopy Algorithms to Solve Polynomial Systems*

Anton Leykin[1], Jan Verschelde[2], and Yan Zhuang[3]

Department of Mathematics, Statistics, and Computer Science
University of Illinois at Chicago, 851 South Morgan (M/C 249)
Chicago, IL 60607-7045, USA.
[1]leykin@math.uic.edu, http://www.math.uic.edu/~leykin
[2]jan@math.uic.edu, http://www.math.uic.edu/~jan
[3]yzhuan1@uic.edu, http://www2.uic.edu/~yzhuan1

Abstract. Homotopy continuation methods to compute numerical approximations to all isolated solutions of a polynomial system are known as "embarrassingly parallel", i.e.: because of their low communication overhead, these methods scale very well for a large number of processors. Because so many important problems remain unsolved mainly due to their intrinsic computational complexity, it would be embarrassing not to develop parallel implementations of polynomial homotopy continuation methods. This paper concerns the development of "parallel PHCpack", a project which started a couple of years ago in collaboration with Yusong Wang, and which currently continues with Anton Leykin (parallel irreducible decomposition) and Yan Zhuang (parallel polyhedral homotopies). We report on our efforts to make PHCpack ready to solve large polynomial systems which arise in applications.

2000 Mathematics Subject Classification. Primary 65H10. Secondary 14Q99, 68W30.

Key words and phrases. Continuation methods, high performance continuation, jumpstarting homotopies, linear-product systems, parallel computation, path following, polynomial systems, polyhedral homotopies, simplex system.

1 Motivation: we want to solve large systems

To solve a polynomial system $f(\mathbf{x}) = \mathbf{0}$, a homotopy $h(\mathbf{x}, t) = \mathbf{0}$ connects f to a start system $g(\mathbf{x}) = \mathbf{0}$ (g stands for *generic*, i.e.: all start solutions are regular), for example of the following form

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

* This material is based upon work supported by the National Science Foundation under Grant No. 0134611 and Grant No. 0410036. This work was partially supported by the National Center for Supercomputing Applications under DMS060008N and utilized the IBM pSeries 690 system copper. *Date:* 22 June 2006.

where the random complex constant γ ensures with probability one that all solution of $h(\mathbf{x}, t) = \mathbf{0}$ are regular for all $t \in [0, 1)$. Thanks to this regularity, predictor-corrector methods can track all solution paths defined by $h(\mathbf{x}(t), t) = 0$, as t moves from 0 to 1, starting at $t = 0$ at the solution of $g(\mathbf{x}) = \mathbf{0}$ and ending at $t = 1$, at approximate isolated solutions of $f(\mathbf{x}) = \mathbf{0}$.

We say that the system f is *large* if the homotopy we use to solve it requires more than 100,000 solutions to track. Although large does not always automatically imply "difficult", numerical problems are more likely to occur. This paper is concerned with three issues:

- For efficiency, it is undesirable to keep all solutions in main memory.
- Numerical stabilities may occur as dimensions and degrees grow.
- Quality control on the computed solutions must be done fast.

Recent work produced two different software systems: PHoMpara [7] (a parallel version of PHoM [8]) and POLSYS_GLP [30] (based upon HOMPACK [39]). The first software system uses polyhedral homotopies, while the second one applies linear-product start systems to solve polynomial systems. Independent of the performance of these programs relative to PHCpack, it matters that PHCpack [33] offers both types of homotopies.

The parallel implementation of the path trackers in PHCpack started in a joint work with Yusong Wang [36] and yielded a parallel version of the Pieri homotopies [10] (refined in [12], see also [18]). Parallel path tracking is discussed in [1], [4], [5], [9], [20], and [21]. Our computational experiments showed that distributing all path tracking jobs at the start performs well when all paths require the same amount of work. Otherwise, dynamic load balancing is needed to achieve an optimal performance.

The parallel PHCpack project is currently continued in collaboration with Anton Leykin [16] (see also [14]) and Yan Zhuang [37]. The work in [14] reports on the parallel implementation of methods to decompose a positive dimensional solution set, using monodromy [24] and traces [25], as needed in a numerical irreducible decomposition [23]. The techniques presented in this paper provide efficient homotopies to create *witness sets* of these positive dimensional solution sets, see [26] and [27] for introductions to numerical algebraic geometry. The development of parallel polyhedral homotopies (described in [37]) will increase the capabilities of PHCpack to deal with solution sets of larger degrees.

The parallel software was developed using personal cluster computers from RocketCalc and ported to similar Beowulf clusters like UIC's supercomputer argo. Most recently, the parallel path tracking facilities of PHCpack were installed on NCSA's IBM pSeries 690 system running AIX 5.3. The use of MPI and a description of other interfaces to PHCpack can be found in [15].

In this paper we resolve the three issues raised above. To avoid the storage of all start solutions in the main memory, we propose to *jumpstart* homotopies, either by computing the roots whenever and wherever they are needed, or by reading the start solutions from file. For the sparsest class of polynomial systems, we discovered a numerically stable solver which computes the magnitudes of the

roots separately, avoiding numerical overflow or underflow. Thirdly, for efficient quality control of the results, the programs are allowed only one linear sweep through the file which contain the solutions.

As noted before, what we call "large" does not automatically imply "difficult". The homotopies we consider in this paper are *optimal* (a notion introduced in [10]): no solution path diverges to infinity, so the overall cost of the solver is polynomial in the output size.

Acknowledgements. A first draft of this paper corresponds to a talk given by the second author at the AMS special session on Numerical Solution of Polynomial Systems held at the University of Notre Dame, 8-9 April 2006. We thank the organizers of this session, Chris Peterson and Andrew Sommese, for the opportunity to present this work. We thank the referee for valuable comments on the first draft.

2 Jumpstarting Homotopies

Homotopy continuation methods compute one solution at a time. Keeping all start solutions in main memory may decrease the overall performance, or even be impossible. Assuming a manager/worker protocol, our solution is the following:

1. The manager reads a 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 to compute the solutions whenever needed.
3. As soon as a worker reports the end of a solution path back to the manager, the solution is written to a file.

Solutions to total degree start systems can be computed very fast, faster than they can be retrieved from file. A lexicographical indexing scheme allows the manager to dictate only which node has to track which path. As all nodes know how to solve total degree start systems, they only need a number, reducing the communication overhead.

For example, a typical total degree start system may look like

$$g(x_1, x_2, x_3) = \begin{cases} x_1^4 - 1 = 0 \\ x_2^5 - 1 = 0 \\ x_3^3 - 1 = 0. \end{cases} \tag{2}$$

It has $4 \times 5 \times 3 = 60$ solutions.

We can get the 25th solution via a decomposition of 24 (start counting from 0): $24 = 1(5\times 3)+3(3)+0$. Let us verify this via lexicographic enumeration:

$$\begin{array}{c} 000\to 001\to 002\to 010\to 011\to 012\to 020\to 021\to 022\to 030\to 031\to 032\to 040\to 041\to 042 \\ 100\to 101\to 102\to 110\to 111\to 112\to 120\to 121\to 122\to \boxed{130}\to 131\to 132\to 140\to 141\to 142 \\ 200\to 201\to 202\to 210\to 211\to 212\to 220\to 221\to 222\to 230\to 231\to 232\to 240\to 241\to 242 \\ 300\to 301\to 302\to 310\to 311\to 312\to 320\to 321\to 322\to 330\to 331\to 332\to 340\to 341\to 342 \end{array} \tag{3}$$

Although examples for which the total degree homotopy is optimal are fairly rare, one interesting application appears in magnetism, posed by Shigetoshi Katsura [13], see also [3]. This applications leads to a family of systems, which scales for up to any number of equations and variables. All equations in the systems are of degree two, except for one linear equation. The largest polynomial system in this family we considered has 21 equations and a total degree of $2^{20} = 1,048,576$. Recently, the `mpi2track` function in PHCpack tracked all 1,048,576 paths using a personal cluster of fourteen CPUs all running at a clockspeed of 2.4Ghz, in a traditional manager/worker dynamic load distribution model. It took 32 hours and 44 minutes to complete the tracking, leading to an output file of 1.3Gb.

While homotopies based on the total degree are rarely efficient, the simple principle of lexicographic enumeration of the start solutions applies to linear-product start systems. These start systems first occurred in [38], using multi-homogeneous homotopies [19] and were generalized in [34]. Compared to the total degree, homotopies using linear-product start systems typically follow far fewer solution paths than the total degree.

All equations in a linear-product start system are products of linear equations, of the form of the system in (4). Every $\cdots$ in (4) corresponds to a linear polynomial with randomly chosen coefficients.

$$g(\mathbf{x}) = \begin{cases} (\cdots)\cdot(\cdots)\cdot(\cdots)\cdot(\cdots) = 0 \\ (\cdots)\cdot(\cdots)\cdot(\cdots)\cdot(\cdots)\cdot(\cdots) = 0 \\ (\cdots)\cdot(\cdots)\cdot(\cdots) = 0 \end{cases} \tag{4}$$

The random choice of the coefficients of the linear factors of the products in the linear-product start systems implies that the maximal number of isolated solutions is attained. Moreover, if every monomial in the target system $f(\mathbf{x}) = \mathbf{0}$ also occurs in the corresponding equation of the start system $g(\mathbf{x}) = \mathbf{0}$, then all isolated solutions of $f(\mathbf{x}) = \mathbf{0}$ lie at the end of some solution path defined by a homotopy using a linear-product start system $g(\mathbf{x}) = \mathbf{0}$, see [34]. Efficient implementations of this type of homotopies are described in [40] and [30].

Just like (2), the solution of the start system in (4) can be enumerated lexicographically. As the linear-product start system is stored on file in its product form, one does not need storing the start solutions on file. Moreover, any node in a parallel computer can solve for one particular solution. While the main motivation is to avoid to store the complete list of start solutions in main memory, an additional advantage is a reduced communication overhead: instead of passing the start solution vector from manager to path tracking worker, the manager simply has to pass out the label (or group of labels) to the nodes.

While there are as many candidates as the total degree, the number of start solutions (and the corresponding generalized Bézout number) is typically much less than the total degree. For efficiency – as the sequential root counting procedures in PHCpack already do – an incremental LU factorization of the coefficient matrices for each linear system leading to a start solution is an effective technique to prune the tree of all possible combinations of factors in the products of g.

3 A Numerically Stable Solver for "Simplex" Systems

Homotopies implementing Bernshteĭn's theorem [2] are described in [35]. What we now call *polyhedral homotopies* follows from the more general treatment in [11]. In [17] these methods are explained in greater detail. Bernshteĭn showed in [2] that the mixed volume of the Newton polytopes of the polynomial system bound the number of solutions (with all variables different from zero). For systems with randomly chosen coefficients, this bound is sharp. The first stage of a polyhedral homotopy method consists in the calculation of this mixed volume, see [8] and [6] for efficient programs to perform this task.

Polyhedral homotopies require in their second stage the solution of a polynomial system with random coefficients. Choosing all complex coefficients on the unit circle in the complex plane naturally leads to a well-conditioned polynomial system. Despite this good choice of the coefficients, previous versions of our software failed for some large examples used for testing the parallel polyhedral homotopies [37].

Consider for example the 12-dimensional polynomial system below in (5). It occurs as just one of the one of the 11,417 start systems generated by polyhedral homotopies to create a random coefficient start system occuring in the design of a robot (see [28], [29], [31], [32]):

$$\begin{cases} b_1 x_5 x_8 + b_2 x_6 x_9 = 0 \\ b_3 x_2^2 + b_4 = 0 \\ b_5 x_1 x_4 + b_6 x_2 x_5 = 0 \\ c_1^{(k)} x_1 x_4 x_7 x_{12} + c_2^{(k)} x_1 x_6 x_{10}^2 + c_3^{(k)} x_2 x_4 x_8 x_{10} + c_4^{(k)} x_2 x_4 x_{11}^2 \\ \quad + c_5^{(k)} x_2 x_6 x_8 x_{11} + c_6^{(k)} x_3 x_4 x_9 x_{10} + c_7^{(k)} x_4^2 x_{12}^2 + c_8^{(k)} x_3 x_6 \\ \quad + c_9^{(k)} x_4^2 + c_{10}^{(k)} x_9 = 0, \quad k = 1, 2, \ldots, 9. \end{cases} \tag{5}$$

The coefficients b_i, $i = 1, 2, \ldots, 6$, and $c_j^{(k)}$, $j = 1, 2, \ldots, 9$, $k = 1, 2, \ldots, 9$ are randomly chosen complex numbers, chosen so that $|b_i| = 1$ and $|c_j^{(k)}| = 1$. Because of this good choice of coefficients, all solutions are well conditioned. Despite the high degrees, there are only one hundred isolated solutions in $(\mathbb{C}^*)^{12}$, $\mathbb{C}^* = \mathbb{C} \setminus \{0\}$, because of the sparsity of the system: only 13 distinct monomials (after appropriate division).

We call such system a *simplex system*[1] and we can solve it fast, reducing it to *binomial system* using LU factorization on the coefficient matrix of the system. Every equation in a binomial system has exactly two monomials with nonzero coefficients. In compact form, we denote a binomial system by $\mathbf{x}^A = \mathbf{b}$ and solve if via the Hermite normal form of A, computing a unimodular matrix M ($\det(M) = \pm 1$), so that $MA = U$, with U is an upper triangular matrix and

[1] because its support corresponds to a simplex. We thank the referee for suggesting this catchy term.

$|\det(U)| = |\det(A)|$. Let $\mathbf{x} = \mathbf{z}^M$, then $\mathbf{x}^A = \mathbf{z}^{MA} = \mathbf{z}^U$, so we have reduced $\mathbf{x}^A = \mathbf{b}$ to $\mathbf{z}^U = \mathbf{b}$.

For example, for two variables we write

$$[z_1 \quad z_2] \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix} = [b_1 \quad b_2] \quad \text{for the system} \quad \begin{cases} z_1^{u_{11}} & = b_1 \\ z_1^{u_{12}} \; z_2^{u_{22}} & = b_2 \end{cases}. \tag{6}$$

Forward substitution on the triangular system shows that there are exactly $|\det(A)|$ distinct isolated solutions, and $|b_k| = 1$ implies $|z_k| = 1$, for every solution component, $k = 1, 2, \ldots, n$. So our binomial systems are numerically very well conditioned.

A simplex system is denoted by $C\mathbf{x}^A = \mathbf{b}$, where C is some coefficient matrix. The natural approach to reduce this simplex system to a binomial system is via a LU-factorization on C. Assuming $\det(C) \neq 0$, we compute a lower and an upper triangular matrix L and U so that $C = LU$ and solve two systems:

$$\begin{array}{ll} (1) & LU\mathbf{y} = \mathbf{b}, \text{ a linear system;} \\ (2) & \mathbf{x}^A = \mathbf{y}, \text{ a binomial system.} \end{array} \tag{7}$$

However, this algorithm is numerically unstable! Even if all coefficients for C and $\mathbf{b}$ are chosen to lie on the complex unit circle, varying magnitudes in the intermediate values for $\mathbf{y}$ do occur. High powers, in the range of 50 and over occur in the Hermite normal form for larger systems and magnify the imbalance between the magnitudes in $\mathbf{y}$ up to the point where numerical underflow or overflow crashes the solver.

Our new solver separates the magnitudes of the solutions from their phases. Using the following notations $\mathbf{z} = |\mathbf{z}|\mathbf{e}_\mathbf{z}$, $\mathbf{e}_\mathbf{z} = \exp(i\theta_\mathbf{z})$, $\mathbf{y} = |\mathbf{y}|\mathbf{e}_\mathbf{y}$, $\mathbf{e}_\mathbf{y} = \exp(i\theta_\mathbf{y})$, $i = \sqrt{-1}$, we rewrite the binomial system and solve

$$\mathbf{z}^U = \mathbf{y} : |\mathbf{z}|^U \mathbf{e}_\mathbf{z}^U = |\mathbf{y}|\mathbf{e}_\mathbf{y} \Leftrightarrow \begin{cases} \mathbf{e}_\mathbf{z}^U = \mathbf{e}_\mathbf{y} \\ |\mathbf{z}|^U = |\mathbf{y}| \end{cases} \tag{8}$$

The first binomial system $\mathbf{e}_z^U = \mathbf{e}_\mathbf{y}$ is well conditioned because all components of the right hand side vector have modulus one. To find the magnitudes $|\mathbf{z}|$ we solve $|\mathbf{z}|^U = |\mathbf{y}|$, using a logarithmic scale, i.e.: $U\log(|\mathbf{z}|) = \log(|\mathbf{y}|)$. Even as the magnitude of the values $\mathbf{y}$ may be extreme, $\log(|\mathbf{y}|)$ will be modest in size.

Our new numerically stable solver to solve a simplex system $C\mathbf{x}^A = \mathbf{b}$ executes the following steps:

1. The LU factorization of C yields $\mathbf{x}^A = \mathbf{y}$, where $C\mathbf{y} = \mathbf{b}$.
2. Use the Hermite normal form of A, $MA = U$, $\det(M) = \pm 1$, to solve the 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 the upper triangular linear system $U\log(|\mathbf{z}|) = \log(|\mathbf{y}|)$.
4. Compute the 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, causing floating point overflow or underflow, we deal with $|\mathbf{z}|$ at a logarithmic scale and never raise small or large numbers to high powers. Only at the very end do we calculate $|\mathbf{x}| = 10^{\log(|\mathbf{x}|)}$ and $\mathbf{x} = |\mathbf{x}|\mathbf{e}_{\mathbf{x}}$.

For more on the parallel implementation of polyhedral homotopy methods in PHCpack, we refer to [37].

4 Scanning Solution Files into Frequency Tables

During runtime, we often want to monitor the progress of a large path tracking job, and 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. For each solution at the end of path, Newton's method reports three floating point numbers:

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. These frequency tables used to judge the quality of solution lists are a first step to employ so-called endgames, eventually with some reruns of the paths at tighter tolerances.

Recall the application posed by Shigetoshi Katsura [13] we mentioned in Section 2, which led to a homotopy of 2^{20} paths, leaving a file of 1.3Gb to process. Reading all solutions from file into main memory takes about 4 minutes and occupies more than 400Mb. While most modern workstations are well equipped with a large internal memory, to determine whether all solutions are distinct (no path crossing has happened) we do not need to occupy that much memory. The data compression of 400Mb into about 42Mb is by randomly projecting the solution vectors (of length 21) to the plane. The creation of a quadtree [22] (using as many levels till each leaf holds no more than 1,024 points) takes about 7 seconds and occupies about 58Mb. Sorting the leaves of the quadtree to determine path crossings takes less than a second.

Notice that the time to read all solutions from disk (4 minutes) dominates the time to create the quadtree (7 seconds).

As the quality analysis of solution lists can already be done while the lists are still incomplete, remedial action or more computationally demanding endgames (we refer to [27, Chapter 10] for an overview) will lead to extra jobs to be distributed among the worker nodes.

5 Towards High Performance Continuation ...

The polynomial systems we typically consider have a number n of variables which is relatively modest, averaging around 8 or 10. The nonlinearity results in a large number of solution paths, which we denote by R for the root count used in the homotopy. In this paper we considered R in the range of 100,000 and higher. Because $R \gg n$, several issues must be addressed to improve the performance of parallel homotopies. In particular, we avoid storing all start solutions in main memory by jumpstarting the homotopies. We discovered a numerical instability in the polyhedral homotopies which was not treated before and emphasized the need for fast quality control of large solution lists.

The "parallel PHCpack" effort has led to good speedups of running times on existing benchmark systems, essentially leaving the basic path tracking facilities and homotopy constructors intact, calling the routines in PHCpack in conjunction with message passing primitives. To solve large polynomial systems, an internal reorganization of PHCpack is needed, in an effort to turn Polynomial Homotopy Continuation into High Performance Continuation.

References

1. D.C.S. Allison, A. Chakraborty, and L.T. Watson. Granularity issues for solving polynomial systems via globally convergent algorithms on a hypercube. *J. of Supercomputing*, 3:5–20, 1989.
2. D.N. Bernshteĭn. The number of roots of a system of equations. *Functional Anal. Appl.*, 9(3):183–185, 1975. Translated from *Funktsional. Anal. i Prilozhen.*, 9(3):1–4,1975.
3. W. Boege, R. Gebauer, and H. Kredel. Some examples for solving systems of algebraic equations by calculating Groebner bases. *J. Symbolic Computation*, 2:83–98, 1986.
4. A. Chakraborty, D.C.S. Allison, C.J. Ribbens, and L.T. Watson. Note on unit tangent vector computation for homotopy curve tracking on a hypercube. *Parallel Computing*, 17(12):1385–1395, 1991.
5. A. Chakraborty, D.C.S. Allison, C.J. Ribbens, and L.T. Watson. The parallel complexity of embedding algorithms for the solution of systems of nonlinear equations. *IEEE Transactions on Parallel and Distributed Systems*, 4(4):458–465, 1993.
6. T. Gao, T.Y. Li, and M. Wu. Algorithm 846: MixedVol: A software package for mixed volume computation. *ACM Trans. Math. Softw.*, 31(4):555–560, 2005.
7. T. Gunji, S. Kim, K. Fujisawa, and M. Kojima. PHoMpara – parallel implementation of the Polyhedral Homotopy continuation Method for polynomial systems. *Computing*, 77(4):387–411, 2006.
8. T. Gunji, S. Kim, M. Kojima, A. Takeda, K. Fujisawa, and T. Mizutani. PHoM – a polyhedral homotopy continuation method for polynomial systems. *Computing*, 73(4):55–77, 2004.
9. S. Harimoto and L.T. Watson. The granularity of homotopy algorithms for polynomial systems of equations. In G. Rodrigue, editor, *Parallel processing for scientific computing*, pages 115–120. SIAM, 1989.
10. B. Huber, F. Sottile, and B. Sturmfels. Numerical Schubert calculus. *J. Symbolic Computation*, 26(6):767–788, 1998.

11. B. Huber and B. Sturmfels. A polyhedral method for solving sparse polynomial systems. *Math. Comp.*, 64(212):1541–1555, 1995.
12. B. Huber and J. Verschelde. Pieri homotopies for problems in enumerative geometry applied to pole placement in linear systems control. *SIAM J. Control Optim.*, 38(4):1265–1287, 2000.
13. S. Katsura. Users posing problems to PoSSO. In the PoSSO Newsletter, no. 2, July 1994, edited by L. Gonzelez-Vega and T. Recio.
14. A. Leykin and J. Verschelde. Decomposing solution sets of polynomial systems: a new parallel monodromy breakup algorithm. Accepted for publication in *The International Journal of Computational Science and Engineering*.
15. A. Leykin and J. Verschelde. Interfacing with the numerical homotopy algorithms in phcpack. Proceedings of ICMS'06, this volume.
16. A. Leykin and J. Verschelde. Factoring solution sets of polynomial systems in parallel. In Tor Skeie and Chu-Sing Yang, editors, *Proceedings of the 2005 International Conference on Parallel Processing Workshops. 14-17 June 2005. Oslo, Norway. High Performance Scientific and Engineering Computing*, pages 173–180. IEEE Computer Society, 2005.
17. T.Y. Li. Numerical solution of polynomial systems by homotopy continuation methods. In F. Cucker, editor, *Handbook of Numerical Analysis. Volume XI. Special Volume: Foundations of Computational Mathematics*, pages 209–304. North-Holland, 2003.
18. T.Y. Li, X. Wang, and M. Wu. Numerical schubert calculus by the pieri homotopy algorithm. *SIAM J. Numer. Anal.*, 40(2):578–600, 2002.
19. A. Morgan and A. Sommese. A homotopy for solving general polynomial systems that respects m-homogeneous structures. *Appl. Math. Comput.*, 24(2):101–113, 1987.
20. A.P. Morgan and L.T. Watson. A globally convergent parallel algorithm for zeros of polynomial systems. *Nonlinear Analysis*, 13(11):1339–1350, 1989.
21. W. Pelz and L.T. Watson. Message length effects for solving polynomial systems on a hypercube. *Parallel Computing*, 10(2):161–176, 1989.
22. H. Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16(2), 1984.
23. A.J. Sommese, J. Verschelde, and C.W. Wampler. Numerical decomposition of the solution sets of polynomial systems into irreducible components. *SIAM J. Numer. Anal.*, 38(6):2022–2046, 2001.
24. A.J. Sommese, J. Verschelde, and C.W. Wampler. Using monodromy to decompose solution sets of polynomial systems into irreducible components. In C. Ciliberto, F. Hirzebruch, R. Miranda, and M. Teicher, editors, *Application of Algebraic Geometry to Coding Theory, Physics and Computation*, pages 297–315. Kluwer Academic Publishers, 2001. Proceedings of a NATO Conference, February 25 - March 1, 2001, Eilat, Israel.
25. A.J. Sommese, J. Verschelde, and C.W. Wampler. Symmetric functions applied to decomposing solution sets of polynomial systems. *SIAM J. Numer. Anal.*, 40(6):2026–2046, 2002.
26. A.J. Sommese, J. Verschelde, and C.W. Wampler. Introduction to numerical algebraic geometry. In A. Dickenstein and I.Z. Emiris, editors, *Solving Polynomial Equations. Foundations, Algorithms and Applications*, volume 14 of *Algorithms and Computation in Mathematics*, pages 301–337. Springer–Verlag, 2005.
27. A.J. Sommese and C.W. Wampler. *The Numerical solution of systems of polynomials arising in engineering and science*. World Scientific, 2005.

28. H.-J. Su. *Computer-Aided Constrained Robot Design Using Mechanism Synthesis Theory*. PhD thesis, University of California, Irvine, 2004.
29. H.-J. Su and J.M. McCarthy. Kinematic synthesis of RPS serial chains. In the Proceedings of the ASME Design Engineering Technical Conferences (CDROM), Chicago, IL, Sep 2-6, 2003.
30. 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.*
31. H.-J. Su, J.M. McCarthy, and L.T. Watson. Generalized linear product homotopy algorithms and the computation of reachable surfaces. *ASME Journal of Information and Computer Sciences in Engineering*, 4(3):226–234, 2004.
32. H.-J. Su, C.W. Wampler, and J.M. McCarthy. Geometric design of cylindric PRS serial chains. *ASME Journal of Mechanical Design*, 126(2):269–277, 2004.
33. J. Verschelde. Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Trans. Math. Softw.*, 25(2):251–276, 1999. Software available at `http://www.math.uic.edu/~jan`.
34. J. Verschelde and R. Cools. Symbolic homotopy construction. *Applicable Algebra in Engineering, Communication and Computing*, 4(3):169–183, 1993.
35. J. Verschelde, P. Verlinden, and R. Cools. Homotopies exploiting Newton polytopes for solving sparse polynomial systems. *SIAM J. Numer. Anal.*, 31(3):915–930, 1994.
36. J. Verschelde and Y. Wang. Computing feedback laws for linear systems with a parallel Pieri homotopy. In Y. Yang, editor, *Proceedings of the 2004 International Conference on Parallel Processing Workshops, 15-18 August 2004, Montreal, Quebec, Canada. High Performance Scientific and Engineering Computing*, pages 222–229. IEEE Computer Society, 2004.
37. J. Verschelde and Y. Zhuang. Parallel implementation of the polyhedral homotopy method. To appear in the proceedings of *The 8th Workshop on High Performance Scientific and Engineering Computing (HPSEC-06)*, Columbus, Ohio, USA, August 18, 2006.
38. C.W. Wampler, A.P. Morgan, and A.J. Sommese. Numerical continuation methods for solving polynomial systems arising in kinematics. *ASME J. of Mechanical Design*, 112(1):59–68, 1990.
39. L.T. Watson, S.C. Billups, and A.P. Morgan. Algorithm 652: HOMPACK: a suite of codes for globally convergent homotopy algorithms. *ACM Trans. Math. Softw.*, 13(3):281–310, 1987.
40. S.M. Wise, A.J. Sommese, and L.T. Watson. Algorithm 801: POLSYS_PLP: a partitioned linear product homotopy code for solving polynomial systems of equations. *ACM Trans. Math. Softw.*, 26(1):176–200, 2000.