Algorithm 801: POLSYS_PLP: A Partitioned Linear Product Homotopy Code for Solving Polynomial Systems of Equations

STEVEN M. WISE
Virginia Polytechnic Institute and State University
ANDREW J. SOMMESE
University of Notre Dame
and
LAYNE T. WATSON
Virginia Polytechnic Institute and State University

Globally convergent, probability-one homotopy methods have proven to be very effective for finding all the isolated solutions to polynomial systems of equations. After many years of development, homotopy path trackers based on probability-one homotopy methods are reliable and fast. Now, theoretical advances reducing the number of homotopy paths that must be tracked, and in the handling of singular solutions, have made probability-one homotopy methods even more practical. POLSYS_PLP consists of Fortran 90 modules for finding all isolated solutions of a complex coefficient polynomial system of equations. The package is intended to be used in conjunction with HOMPACK90 (Algorithm 777), and makes extensive use of Fortran 90 derived data types to support a partitioned linear product (PLP) polynomial system structure. PLP structure is a generalization of m-homogeneous structure, whereby each component of the system can have a different m-homogeneous structure. The code requires a PLP structure as input, and although finding the optimal PLP structure is a difficult combinatorial problem, generally physical or engineering intuition about a problem yields a very good PLP structure. POLSYS_PLP employs a sophisticated power series end game for handling singular solutions, and provides support for problem definition both at a high level and via hand-crafted code. Different PLP structures and their corresponding Bezout

This work was supported in part by Air Force Office of Scientific Research grant F49620-96-1-010, National Science Foundation grant DMS-962596, and National Aeronautics and Space Administration grant NAG-2-118.
Authors' addresses: S. M. Wise, Department of Mathematics, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061-0123; A. J. Sommese, Department of Mathematics, University of Notre Dame, Notre Dame, IN 46556-5683; email: sommese.1@nd.edu; L. T. Watson, Departments of Computer Science and Mathematics, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061-0106; email: ltw@cs.vt.edu.

numbers can be systematically explored before committing to root finding.

Categories and Subject Descriptors: D.3.2 [**Programming Languages**]: Language Classifications—*Fortran 90*; G.1.5 [**Numerical Analysis**]: Roots of Nonlinear Equations—*Continuation (homotopy) methods*; *Polynomials, methods for*; *Systems of equations*; G.4 [**Mathematics of Computing**]: Mathematical Software

General Terms: Algorithms

Additional Key Words and Phrases: Chow-Yorke algorithm, curve tracking, fixed point, globally convergent, homotopy methods, m-homogeneous, partitioned linear product, probability-one, zero

1. INTRODUCTION

Polynomial systems of equations arise in many applications: robotics, computer vision, kinematics, chemical kinetics, truss design, geometric modeling, and many others (see Morgan [1987] and Verschelde [1996]). In applications where all the solutions, or a significant number of solutions, must be found, or when locally convergent methods fail, globally convergent, probability-one homotopy methods are preferred. Homotopy methods for polynomial systems were first proposed by Garcia and Zangwill [1977] and Drexler [1979]. While the method in Garcia and Zangwill [1977] was easily demonstrated by topological techniques and the start system was easily solved, the homotopy produced many more paths than the total degree of the system. Drexler used the powerful results of algebraic geometry to prove his method. Two years later, using differential geometry, Chow et al. [1979] improved on the results of Garcia and Zangwill with a general homotopy which produced the same number of paths as the number of solutions (provided there are a finite number of them), counting multiplicities and solutions at infinity. The start system of this homotopy was difficult to solve. Morgan [1983] solved this problem with a much simpler start system, which had trivially obtained roots, and could be used in a general homotopy. The suggestion by Wright [1985] and Morgan [1986a; 1986b] to track the homotopy zero curves in complex projective space, rather than in Euclidean space, was another fundamental breakthrough—in complex projective space certain paths would no longer diverge to infinity (have infinite arc length), and paths in general were made shorter. Other notable publications, which appear around the end of the first decade of research, are by Meintjes and Morgan [1985], Tsai and Morgan [1985], and Watson et al. [1987].

In roughly the past decade, since the development of robust, efficient homotopy path tracking algorithms, work has shifted toward lowering the number of paths that must be tracked. In essence, all the methods try to construct a start system for the homotopy map that better models the structure of the given polynomial system, the target system for the homotopy map. Early work includes m-homogeneous theory by Morgan and Sommese [1987a]. In m-homogeneous theory the powerful connection of probability-one homotopy methods for polynomials with the field of algebraic geometry is reestablished (see Drexler [1979]) with the generalization of the classical theorem of Bezout. Generalizations of m-homogeneous theory appeared in Verschelde and Haegemans [1993] with the GBQ method, and in Verschelde and Cools [1993] with set-structure analysis. The methods in both Verschelde and Haegemans [1993] and Verschelde and Cools [1993] are derived by modifying slightly the main theorem in Morgan and Sommese [1987a], but are nonetheless important. The most recent definitive theoretical work is that of Morgan et al. [1995]. Though the theorems of Morgan et al. [1995] are very powerful, used in their full generality, they suggest more an approach for exploiting structure than an algorithm. The method used here in POLSYS_PLP for constructing the start system is essentially the same as that in Verschelde and Haegemans [1993], but is based on the results of Morgan et al. [1995].

Attention has also been paid to the problem of calculating singular solutions of polynomial systems using homotopy methods. Approaches have been proposed based on Newton's method (see for example Griewank [1985]), and based on complex analysis as in the work of Morgan et al. [1991; 1992a; 1992b]. The most useful approach of those mentioned is found in Morgan et al. [1992b], where the foundation of a reasonable end game is laid. Other work has come from Sosonkina et al. [1996], and their approach, which is used in the polynomial system routine POLSYS1H of HOMPACK90 [Watson et al. 1997], is moderately successful on low, odd order singularities. To accurately compute a singular solution of order 30, say, requires a very sophisticated end game like that in Morgan et al. [1992b], which POLSYS_PLP incorporates.

Publically available codes for solving polynomial systems of equations using globally convergent, probability-one homotopy methods do exist: HOMPACK [Watson et al. 1987], written in Fortran 77, and HOMPACK90 [Watson et al. 1997], written in Fortran 90, both have polynomial system solvers. CONSOL in the book by Morgan [1987] is also written in Fortran 77. However, neither HOMPACK90 nor CONSOL has a sophisticated start system that can lower the number of homotopy paths that must be tracked below the total degree. The package PHCPACK by Verschelde [1997], written in Ada, allows a great variety of choices for the start system, and uses an end game like the one proposed in Morgan et al. [1992b]. PHCPACK, based on BKK theory, has a distinctly combinatorial flavor, and tends to be rather slow on large-scale production problems.

Polynomial structure is a complicated subject, attacked variously by the combinatorial BKK theory [Verschelde and Cools 1993] and algebraic geometry [Morgan et al. 1995]. A design choice of POLSYS_PLP is to strike a balance between the most general structural descriptions (yielding minimal numbers of paths to track, but extremely difficult algorithmically) and no structure at all (where the total degree number of paths must be tracked, algorithmically trivial). The trade-off is moot, because a search for structure may very well cost more than simply tracking the paths a fancier structure would have eliminated. Further, for many industrial problems, an m-homogeneous structure is perfectly adequate, and often even optimal. The structure supported by POLSYS_PLP is called *partitioned linear product*, which in generality lies between m-homogeneous and the arbitrary set-structure supported by PHCPACK.

An excellent source on homotopy methods in general is Allgower and Georg [1990], and Blum et al. [1998] contains many references oriented toward the complexity aspects of polynomial root finding.

2. POLYNOMIAL SYSTEMS OF EQUATIONS

Let $F(z) = 0$ be a polynomial system of n equations in n unknowns. In symbols,

$$F_i(z) = \sum_{j=1}^{n_i}\left[c_{ij}\prod_{k=1}^{n} z_k^{d_{ijk}}\right] = 0, \quad i = 1, \ldots, n, \tag{1}$$

where the c_{ij} are complex (and usually assumed to be different from zero), and the d_{ijk} are nonnegative integers. The *degree* of $F_i(z)$ is

$$d_i = \max_{1 \le j \le n_i} \sum_{k=1}^{n} d_{ijk},$$

and the *total degree* of the system (1) is

$$d = \prod_{i=1}^{n} d_i.$$

Define $F'(w)$ to be the homogenization of $F(z)$:

$$F_i'(w) = w_{n+1}^{d_i} F_i(w_1/w_{n+1}, \ldots, w_n/w_{n+1}), \quad i = 1, \ldots, n. \tag{2}$$

Note that, if $F'(w^0) = 0$, then $F'(\alpha w^0) = 0$ for any complex scalar α. Therefore, "solutions" of $F'(w) = 0$ are (complex) lines through the origin in $\mathbf{C}^{n+1}$. The set of all lines through the origin in $\mathbf{C}^{n+1}$ is called complex projective n-space, denoted $\mathbf{P}^n$, and is a compact n-dimensional complex manifold. (Note that we are using complex dimension: $\mathbf{P}^n$ is $2n$-dimensional as a real manifold.) The solutions of $F'(w) = 0$ in $\mathbf{P}^n$ are identified with the solutions and solutions at infinity of $F(z) = 0$ as follows. If $L \in \mathbf{P}^n$ is a solution to $F'(w) = 0$ with $w = (w_1, w_2, \ldots, w_{n+1}) \in L$ and $w_{n+1} \neq 0$, then $z = (w_1/w_{n+1}, w_2/w_{n+1}, \ldots, w_n/w_{n+1}) \in \mathbf{C}^n$ is a solution to $F(z) = 0$. On the other hand, if $z \in \mathbf{C}^n$ is a solution to $F(z) = 0$, then the line through $w = (z, 1)$ is a solution to $F'(w) = 0$ with $w_{n+1} = 1 \neq 0$.

The standard definition of *solutions to $F(z) = 0$ at infinity* is simply *solutions to $F'(w) = 0$ (in $\mathbf{P}^n$) generated by w with $w_{n+1} = 0$.*

A solution $\hat{w} \in \mathbf{P}^n$ is called *geometrically isolated* if there exists an open ball $B \subset \mathbf{P}^n$, with $\hat{w} \in B$ and no other solutions in B. If no such ball exists, then the solution $\hat{w}$ is said to exist on a *positive dimensional solution set*. Suppose $\hat{w}$ is a geometrically isolated solution to (2), and suppose B is a ball that contains it. For almost all perturbations of the coefficients of the polynomial, the perturbed polynomial has only nonsingular solutions. For all such sufficiently small perturbations of the coefficients, there exists a finite number m of solutions inside B to the perturbed system of equations. This number m is the *multiplicity* of the solution $\hat{w}$ to (2). A solution $\hat{z} \in \mathbf{C}^n$ to (1) is *singular* if the Jacobian matrix at $\hat{z}$, $D_zF(\hat{z})$, is singular, and *nonsingular* otherwise. Singular solutions at infinity are defined analogously in terms of coordinate patches [Morgan 1987]. A solution has multiplicity greater than one precisely when it is singular [Morgan 1987].

Now that the solution set has been described, the following beautiful result can be stated [van der Waerden 1953]:

BEZOUT'S THEOREM. *There are no more than d isolated solutions to $F'(w)$ in $\mathbf{P}^n$. If $F'(w) = 0$ has only a finite number of solutions in $\mathbf{P}^n$, it has exactly d solutions, counting multiplicity.*

In practical problems, finite, nonsingular, geometrically isolated solutions are of great importance and interest; they are also the easiest to deal with in the homotopy setting. However, since it is not possible *a priori* to separate the nonsingular solutions from the singular solutions and solutions at infinity, homotopy algorithms are forced to deal with the latter. Solutions that are singular or at infinity can cause serious numerical difficulties and inefficiency—these two types of solutions are discussed in later sections. More importantly, the problem of handling these "bad" solutions pales in comparison to the potentially huge number of solutions (and homotopy zero curves that must be tracked). The total degree d, called the Bezout number or more precisely the one-homogeneous Bezout number [Morgan and Sommese 1987a], can be overwhelming even for tame-looking problems. For example, 20 cubic equations would have $d = 3^{20} \approx 3.5 \times 10^9$. Consequently, recent research has looked for methods that shrink (in a rigorous sense) the number of solutions that must be computed, while still retaining all the finite isolated solutions. *Reduction*, which seeks to lower the dimension of the system, is one approach that will work, but is not discussed here (see Morgan [1987, Chapter 7]). Sophisticated mathematical approaches, generally speaking, seek to "factor out" a significant portion of the nonphysical solutions (typically, including many solutions at infinity and multiplicities). For many important practical problems this is possible, since often systems that arise from physical models have symmetries and redundancies (which spawn solutions at infinity and multiple solutions), yet only a small number (compared to d) of finite, nonsingular solutions [Morgan and Sommese 1987b; Verschelde 1996; Verschelde and Cools 1993]. The next section discusses one such approach to reducing the number of homotopy zero curves that must be tracked: the partitioned linear product (PLP) homotopy.

3. HOMOTOPIES FOR POLYNOMIAL SYSTEMS

Define a homotopy map $\rho : [0, 1) \times \mathbf{C}^n \to \mathbf{C}^n$ by

$$\rho(\lambda, z) = (1 - \lambda)G(z) + \lambda F(z). \tag{3}$$

$\lambda \in [0, 1)$ is the *homotopy parameter*; $G(z) = 0$ is the *start system*; and $F(z) = 0$ is the *target system*. The goal is to find a start system with the same structure as the target system, while possessing the property that $G(z) = 0$ is easily solved. In this section a start system with a *partitioned linear product* (PLP) structure will be constructed.

Let $P = (P_1, P_2, \dots, P_n)$ be an n-tuple of partitions P_i of the set $\{z_1, z_2, \dots, z_n\}$. That is, for $i = 1, 2, \dots, n$, $P_i = \{S_{i1}, S_{i2}, \dots, S_{im_i}\}$, where S_{ij} has cardinality $n_{ij} \neq 0$, $\cup_{j=1}^{m_i} S_{ij} = \{z_1, z_2, \dots, z_n\}$, and $S_{ij_1} \cap S_{ij_2} = \emptyset$ for $j_1 \neq j_2$. For clarity, P is called the *system partition*, and the P_i are the *component partitions*. For $i = 1, 2, \dots, n$ and $j = 1, 2, \dots, m_i$ define d_{ij} to be the degree of the component F_i in only the variables of the set S_{ij}, that is, considering the variables of $\{z_1, z_2, \dots, z_n\} \setminus S_{ij}$ as constants. Thus if $F_2(z_1, z_2, z_3) = z_2^2 + z_3 z_2^3 - z_1$, $S_{21} = \{z_3\}$, and $S_{22} = \{z_1, z_2\}$, then $d_{21} = 1$, $d_{22} = 3$. It is convenient, though only for the definition of the start system, to rename the variables component-by-component. Let $S_{ij} = \{z_{ij1}, z_{ij2}, \dots, z_{ijn_{ij}}\}$. With all this said, the start system is represented mathematically by $G_i(z) = \Pi_{j=1}^{m_i} G_{ij}$, where

$$G_{ij} = \begin{cases} \left(\sum_{k=1}^{n_{ij}} c_{ijk} z_{ijk} \right)^{d_{ij}} - 1, & \text{if } d_{ij} > 0; \\ 1, & \text{if } d_{ij} = 0, \end{cases} \qquad i = 1, 2, \dots, n, \tag{4}$$

where the numbers $c_{ijk} \in \mathbf{C}_0 = \mathbf{C} \setminus \{0\}$ are chosen at random. The structure defined by the system partition P and manifested in (4) is called the *partitioned linear product* structure. The degree of $G_i(z)$ is

$$\deg(G_i) = \sum_{j=1}^{m_i} d_{ij}.$$

Note that $d_i \leq \deg(G_i)$ always holds—this fact will be important later, when the projective transformation of the homotopy map is defined.

This start system is modeled after the one in Wampler [1994], and, like its model, is desirable because it is computationally efficient and its solutions, all of which are obtained by the solution of a complex linear system, are nonsingular. To be precise, the linear subsystems into which $G(z) = 0$ decomposes, whether solvable or unsolvable, can be uniquely characterized by two lexicographic vectors. The first, $\Phi = (\Phi_1, \Phi_2, \ldots, \Phi_n)$,is called the *factor lexicographic vector*, and the second, $\Delta = (\Delta_1, \Delta_2, \ldots, \Delta_n)$, is called the *degree lexicographic vector*, where $(1, 1, \ldots, 1) \leq \Phi \leq (m_1, m_2, \ldots, m_n)$, and where, given Φ and all $d_{j\Phi_j} \neq 0$, $(0, 0, \ldots, 0) \leq \Delta \leq (d_{1\Phi_1} - 1, d_{2\Phi_2} - 1, \ldots, d_{n\Phi_n} - 1)$. For example, suppose that the lexicographic pair (Φ, Δ) with all $d_{j\Phi_j} \neq 0$ is given. Then the linear system this pair uniquely represents is

$$A_\Phi z = \begin{pmatrix} \sum_{k=1}^{n_{1\Phi_1}} c_{1\Phi_1 k} z_{1\Phi_1 k} \\ \sum_{k=1}^{n_{2\Phi_2}} c_{2\Phi_2 k} z_{2\Phi_2 k} \\ \vdots \\ \sum_{k=1}^{n_{n\Phi_n}} c_{n\Phi_n k} z_{n\Phi_n k} \end{pmatrix} = \begin{pmatrix} e^{i(\Delta_1/d_{1\Phi_1})} \\ e^{i(\Delta_2/d_{2\Phi_2})} \\ \vdots \\ e^{i(\Delta_n/d_{n\Phi_n})} \end{pmatrix} \equiv b_\Delta, \tag{5}$$

where the z_{ijk} are as defined above. Either A_Φ is generically nonsingular, that is, nonsingular for almost all choices of the c_{ijk} from $\mathbf{C}_0$, or structurally singular. If A_Φ is structurally singular, then it contributes no solutions to $G(z) = 0$ and may be ignored. If A_Φ is generically nonsingular, then $A_\Phi z = b_\Delta$ has a unique solution for each Δ such that $(0, 0, \ldots, 0) \leq \Delta \leq (d_{1\Phi_1} - 1, d_{2\Phi_2} - 1, \ldots, d_{n\Phi_n} - 1)$. If some $d_{j\Phi_j} = 0$, then the factor $G_{j\Phi_j} = 1$ cannot be zero and A_Φ need not even be considered.

In order to count the number of solutions of $G(z) = 0$, it must be determined for each Φ whether or not A_Φ is generically nonsingular. If A_Φ is nonsingular then $\Pi_{i=1}^{n} d_{i\Phi_i}$ is added to the "root count." The final root count is the total number of solutions B_{PLP} to $G(z) = 0$ and is called the PLP Bezout number. There is a combinatorial formula for determining whether or not A_Φ is generically invertible [Verschelde and Cools 1993]. For large problems, however, this rule is expensive. POLSYS_PLP uses numerical linear algebra with random real matrices to determine the generic invertibility of A_Φ. The issues of how to choose the c_{ijk} for such a method, and the likelihood of a generically invertible A_Φ being numerically singular, are discussed in detail in Section 8. The numerical algorithm, based on updated Householder reflections, used by POLSYS_PLP for calculating B_{PLP} follows:

Algorithm.

```
**begin** $index(1 : n) := 0$; $B_{PLP} := 0$;
**for** $\Phi(1 : n) := (1, \ldots, 1)$ **step** lexicographically **until** $(m_1, \ldots, m_n)$ **do**
```

```
      Suppose that index and $\Phi$ agree in the first $k-1$ components and disagree
      in the $k$th component.
      for $j := k$ step 1 until $n$ do
        if $d_{j\Phi_j} = 0$ then
          $\Phi(j+1:n) := (m_{j+1}, \dots, m_n)$ ; exit do
        endif
        Form the $j$th row of $A_\Phi$.
        $A_\Phi^T(1:n, 1:j-1)$ has been triangularized from Householder reflec-
        tions that are saved. Apply the saved Householder reflections in order
        (from the 1st to the $(j-1)$st) to $A_\Phi^T(1:n, j)$.
        Calculate and save the Householder reflection for $A_\Phi^T(j:n, j)$ and apply
        it to $A_\Phi^T(1:n, j)$, thus continuing the triangulation of $A_\Phi^T$.
        if $A_\Phi^T(j,j) \approx 0$ then
          Declare $A_\Phi$ structurally singular.
          $\Phi(j+1:n) := (m_{j+1}, \dots, m_n)$ ; exit do
        endif
        if $j = n$ then
          Declare $A_\Phi$ generically nonsingular.
          $B_{PLP} := B_{PLP} + \Pi_{i=1}^n d_{i\Phi_i}$
        endif
      enddo
      index$(1:n) := \Phi(1:n)$
    enddo
```

The importance of the number B_{PLP} derives from the next two theorems.

THEOREM 3.1 [MORGAN ET AL. 1995]. *Let $f : \mathbf{C}^n \to \mathbf{C}^n$ be a system of polynomials and $U \subset \mathbf{C}^n$ be (Zariski) open. Define $N(f, U)$ to be the number of nonsingular solutions to $f = 0$ that are in U. Assume that there are positive integers $r_1, \dots, r_n$ and $m_1, \dots, m_n$ and finitely generated complex vector spaces V_{ij} of polynomials for $i = 1, \dots, n$ and $j = 1, \dots, m_i$, such that*

$$f_i = \sum_{k=1}^{r_i} \prod_{j=1}^{m_i} p_{ijk}, \tag{6}$$

where $p_{ijk} \in V_{ij}$ for $i = 1, \dots, n$, $j = 1, \dots, m_i$ and $k = 1, \dots, r_i$. Let a system g be defined by $g_i = \Pi_{j=1}^{m_i} g_{ij}$, with each g_{ij} a generic choice from V_{ij}. Then

$$N(f, U) \le N(g, U), \tag{7}$$

and (7) is equality if, for each i with $1 \le i \le n$, there is a positive integer k_i such that the $p_{ijk_i} \in V_{ij}$ are generic for $j = 1, \dots, m_i$. Also, $g(z) = 0$ is a suitable start system for the polynomial homotopy $h(t, z) = (1-t)f(z) + tg(z)$ to find all nonsingular solutions to $f(z) = 0$.

Remark 3.1 The reader will have noted that the homotopy of Theorem 3.1 is different from the one given in (3), but this difference is only a cosmetic change of variables $t = 1 - \lambda$. Moreover, the last line of Theorem 3.1 will be made precise in Section 4.

Remark 3.2 The subsystems of $g = 0$ are the systems $\hat{g} = (g_{1j_1}, \ldots, g_{nj_n}) = 0$ with $1 \le j_i \le m_i$ and $i = 1, \ldots, n$ (see Remark 1.1 from Morgan et al. [1995]). In practice, one chooses g so that the subsystems $\hat{g} = 0$ are easy to solve, e.g., so that solving $\hat{g} = 0$ reduces to solving a linear system.

Let $\{e_{ijl} \mid l = 1, \ldots, \ell_{ij}\}$ denote a basis of the vector space V_{ij}. For $i = 1, \ldots, n$ and $j = 1, \ldots, m_i$ define

$$B_{ij} = \{z \mid e_{ijl}(z) = 0 \text{ for } l = 1, \ldots, \ell_{ij}\}.$$

Following Morgan et al. [1995], the *bases have no pairwise intersection with* U if for any choice of j' and j'' with $1 \le j' < j'' \le m_i$,

$$B_{ij'} \cap B_{ij''} \cap U = \emptyset.$$

The next theorem relates the nonsingular solutions of $g(z) = 0$ with those of its subsystems.

THEOREM 3.2 [MORGAN ET AL. 1995]. *Let g and V_{ij} be as in Theorem 3.1. Then*

(1) *$z_0 \in U$ is a nonsingular solution to $g(z) = 0$ if and only if z_0 is a solution to exactly one subsystem of $g(z) = 0$ and it is a nonsingular solution to this subsystem.*

(2) *Assume that the bases for the V_{ij} have no pairwise intersection with U. Then, if $z_0 \in U$ is a nonsingular solution to some subsystem of $g(z) = 0$, it is a solution to exactly one subsystem of $g(z) = 0$.*

Remark 3.3 It follows from Theorem 3.2 that

$$N(g, U) \le \sum_{\substack{1 \le j_1 \le m_1 \\ 1 \le j_2 \le m_2 \\ \vdots \\ 1 \le j_n \le m_n}} N((g_{1j_1}, g_{2j_2}, \ldots, g_{nj_n}), U), \tag{8}$$

with equality if the bases have no pairwise intersection with U. (This is Remark 2.2 from Morgan et al. [1995].)

Suppose that P is a system partition as before. The vector spaces of polynomials V_{ij} will be constructed using P: consider both S_{ij} and d_{ij} for $j = 1, 2, \ldots, m_i$ and $i = 1, 2, \ldots, n$, and define V_{ij} to be the complex vector space of polynomials generated by the monomials in the variables from S_{ij} up to degree d_{ij} and the constant 1. For example, suppose that

$f(z) = 0$ is a polynomial system in five variables for which $P_2 = \{S_{21}, S_{22}\}$, $S_{21} = \{z_2, z_5, z_1\}$, $S_{22} = \{z_3, z_4\}$, $d_{21} = 2$, and $d_{22} = 3$. Then

$$V_{21} = \mathbf{C}\langle z_2^2, z_5^2, z_1^2, z_2z_5, z_2z_1, z_5z_1, z_2, z_5, z_1, 1\rangle,$$

$$V_{22} = \mathbf{C}\langle z_3^3, z_4^3, z_3^2z_4, z_3z_4^2, z_3^2, z_4^2, z_3z_4, z_3, z_4, 1\rangle.$$

Choosing the V_{ij}, albeit implicitly, from P, with $U = \mathbf{C}^n$, ensures that the bases have no pairwise intersection with U. Since $G_{ij} \in V_{ij}$ is generic,

$$N(F, \mathbf{C}^n) \leq N(G, \mathbf{C}^n) = \sum_{\substack{1\leq j_1\leq m_1 \\ 1\leq j_2\leq m_2 \\ \vdots \\ 1\leq j_n\leq m_n}} N((G_{1j_1}, G_{2j_2}, \ldots, G_{nj_n}), \mathbf{C}^n) = B_{PLP}. \tag{9}$$

This entire discussion would be moot if B_{PLP} were not in many cases smaller than the total degree d. In fact, for many practical problems for well-chosen V_{ij}, B_{PLP} is much smaller than the total degree d. The computational implications for the homotopy map (3) with the start system $G(z) = 0$ are clear—only B_{PLP} homotopy zero curves must be tracked.

Since the start system of Theorem 3.1 can result in a lower number of paths to be tracked, while guaranteeing that paths will reach all nonsingular solutions of $f(z) = 0$, the number $N(g, \mathbf{C}^n)$ is commonly referred to as a generalized Bezout number. The PLP method just explained is but one way of arriving at such a number. m-homogeneous theory and the m-homogeneous Bezout number correspond to the special case when each component partition is the same. Morgan et al. [1995] show how the m-homogeneous Bezout number is easily derived from Theorem 3.1. The name partitioned linear product (PLP) is essentially a description of the structure of the start system $G(z) = 0$. A generalization of PLP is the linear product decomposition (LPD), where the start system reduces to a product of linear systems, but need not correspond to a system partition ($S_{ij_1} \cap S_{ij_2} \neq \emptyset$ possibly). LPD, which is exactly equivalent to set-structure analysis [Verschelde and Cools 1993], generalizes PLP because it allows for groupings of variables more general than system partitions. The method of greatest generality is the general product decomposition (GPD). GPD is any method that utilizes Theorem 3.1 in more generality than LPD, so the start system does not reduce to a product of linear systems. There is thus a hierarchy of methods, based on start system complexity:

—one-homogeneous,

—m-homogeneous,

—partitioned linear product,

—linear product decomposition,

—general product decomposition.

As with the the name "m-homogeneous Bezout number," the values of $N(f, \mathbf{C}^n)$ using the the PLP method are called PLP Bezout numbers, and so on for the other methods. Finding the lowest possible PLP Bezout number is a challenging problem. There is no way, short of an exhaustive search through all possible system partitions, of knowing which P will give the lowest value of B_{PLP}. It is possible to construct a heuristic algorithm for picking P. Verschelde [1996] describes one such algorithm for obtaining an m-homogeneous partition, but as pointed out in Verschelde [1996], the heuristic does not always work. Even if B_{PLP} is not minimized, it may still be small enough so that the path tracking is computationally tractable.

4. THE PROBABILITY-ONE ASPECT

Suppose that P is a system partition for (1) corresponding to the PLP Bezout number B_{PLP}. The following theorem demonstrates the probability-one aspect of the homotopy method in POLSYS_PLP.

THEOREM 4.1 *For almost all choices of c_{ijk} in the start system defined by (4), $\rho^{-1}(0)$ consists of B_{PLP} smooth curves emanating from $\{0\} \times \mathbf{C}^n$, which either diverge to infinity as λ approaches 1 or converge to solutions of $F(z) = 0$. Each nonsingular solution of $F(z) = 0$ will have a curve converging to it.*

Theorem 4.1 is essentially a restatement of the last line of Theorem 3.1, but deserves emphasis; its proof can be found in Section A.5 in the appendix of Morgan et al. [1995]. A noteworthy observation is that since the homotopy map ρ is complex analytic, the homotopy parameter λ is monotonically increasing as a function of arc length along the homotopy zero curves starting at $\lambda = 0$ [Morgan 1987]. Thus, the homotopy zero curves never have turning points with respect to λ.

Though B_{PLP} may be much smaller than the total degree, the possibility of tracking paths of (3) that diverge to infinity still exists. These paths pose significant computational challenges, since time is wasted on divergent paths, and large magnitude solutions may not be found if a path is terminated prematurely. Tracking paths in complex projective space, which was originally proposed in Morgan [1986a; 1986b], eliminates these concerns. With a suitable "projective transformation" no paths diverge to infinity as λ approaches 1. Moreover, though not guaranteed, paths tend be shorter in projective space.

Constructing the projective transformation is straightforward [Morgan 1986a; 1986b; Watson et al. 1987]. As with the homogenization of $F(z)$, define the homogenization of $\rho(\lambda, z)$ to be

$$\rho_i'(\lambda, w) = w_{n+1}^{\deg(G_i)} \rho_i\left(\lambda, \frac{w_1}{w_{n+1}}, \dots, \frac{w_n}{w_{n+1}}\right), \quad i = 1, \dots, n.$$

Define the linear function

$$u(w_1, \ldots, w_{n+1}) = \xi_1 w_1 + \xi_2 w_2 + \ldots + \xi_{n+1} w_{n+1},$$

where the numbers $\xi_i \in \mathbf{C}_0$ are chosen at random. The *projective transformation* of $\rho(\lambda, z)$ is

$$\rho''(\lambda, w) = \begin{pmatrix} \rho'_1(\lambda, w) \\ \rho'_2(\lambda, w) \\ \vdots \\ \rho'_n(\lambda, w) \\ u(w)-1 \end{pmatrix}.$$

That the projective transformation can be applied to the homotopy map ρ, without changing the essence of Theorem 4.1, follows from Remark 1.4 of Morgan et al. [1995]. The precise statement follows.

THEOREM 4.2 *For almost all choices of the c_{ijk} in the start system defined by (4) and almost all choices of the ξ in the linear function $u(w)$, $(\rho'')^{-1}(0)$ consists of B_{PLP} smooth curves emanating from $\{0\} \times \mathbf{C}^{n+1}$, which converge to solutions of $F'(w) = 0$. Each nonsingular solution of $F'(w) = 0$ will have a curve converging to it.*

Henceforth, Theorem 4.2 will tacitly be the operative theorem, and references to ρ tacitly assume that the computer implementation actually works with ρ''.

5. HOMOTOPY PATH TRACKING

Theorem 4.1 says that in order to reach the nonsingular solutions of $F(z) = 0$, "smooth" (nonintersecting, nonbifurcating) paths in $\rho^{-1}(0)$ must be tracked. There are fast, reliable ways of doing this numerically. Three different path tracking algorithms (ordinary differential equation based, normal flow, and augmented Jacobian matrix) are described in Watson et al. [1987] and Watson et al. [1997]. Simple linear-predictor, Newton-corrector methods are described in Morgan [1987] and Verschelde [1997]. There is compelling evidence favoring higher-order methods and the normal flow algorithm over simpler schemes [Lundberg and Poore 1991; Morgan et al. 1989; Watson et al. 1987; 1997]. POLSYS_PLP uses the sophisticated homotopy zero curve tracking routine STEPNX from HOMPACK90.

The normal flow algorithm has essentially three phases: prediction, correction, and step size estimation. Once a curve in $\rho^{-1}(0)$ has been tracked to a point for which $\lambda > 1 - \epsilon$, where $0 < \epsilon \ll 1$, the algorithm enters an "end game," discussed in the next section. For $\lambda \in [0, 1 - \epsilon]$, based on Theorem 4.1, it can be assumed that the path being tracked is smooth and that the Jacobian matrix $D\rho(\lambda, z)$ has full rank.

Let $\gamma(s)$ denote a path under consideration, where s is the arc length of the path, and $s = 0$ at $\lambda = 0$. For the prediction phase, assume that several points $P^{(1)} = (\lambda(s_1), z(s_1))$, $P^{(2)} = (\lambda(s_2), z(s_2))$ on γ with corresponding tangent vectors $(d\lambda/ds(s_1), dz/ds(s_1))$, $(d\lambda/ds(s_2), dz/ds(s_2))$ have been found, and h is an estimate of the optimal step (in arc length) to take along γ. The prediction of the next point on γ is

$$Z^{(0)} = p(s_2 + h), \tag{10}$$

where $p(s)$ is the Hermite cubic interpolating $(\lambda(s), z(s))$ at s_1 and s_2. Precisely,

$$\begin{aligned} p(s_1) &= (\lambda(s_1), z(s_1)), \quad & dp/ds(s_1) &= (d\lambda/ds(s_1), dz/ds(s_1)), \\ p(s_2) &= (\lambda(s_2), z(s_2)), \quad & dp/ds(s_2) &= (d\lambda/ds(s_2), dz/ds(s_2)), \end{aligned}$$

and each component of $p(s)$ is a polynomial in s of degree less than or equal to 3.

Starting at the predicted point $Z^{(0)}$, the corrector iteration mathematically is

$$Z^{(k+1)} = Z^{(k)} - [D\rho(Z^{(k)})]^{\dagger}\rho(Z^{(k)}), \quad k = 0, 1, \ldots, \tag{11}$$

where $[D\rho(Z^{(k)})]^{\dagger}$ is the Moore-Penrose pseudoinverse of the $n \times (n + 1)$ Jacobian matrix $D\rho$. Computationally the corrector step $\Delta Z = Z^{(k+1)} - Z^{(k)}$ is the unique minimum 2-norm solution of the equation

$$[D\rho]\Delta Z = -\rho. \tag{12}$$

Small perturbations of the c_{ijk} in $G(z)$ produce small changes in the trajectory γ. Geometrically, the iterates given by (11) return to the zero curve γ along the flow normal to the Davidenko flow (the family of trajectories γ for varying c_{ijk}), hence the name "normal flow algorithm." Robust and accurate numerical linear algebra procedures for solving (12) and for computing the kernel of $[D\rho]$ (tangent vectors required for (10)) are described in detail in Watson et al. [1987].

When the iteration (11) converges, the final iterate $Z^{(k+1)}$ is accepted as the next point on γ, and the tangent vector to the integral curve through $Z^{(k)}$ is used for the tangent—this saves a Jacobian matrix evaluation and factorization at $Z^{(k+1)}$. The next phase, step size estimation, attempts to balance progress along γ with the effort expended on the iteration (11), and is a sophisticated blend of mathematics, computational experience, and mathematical software principles. Complete details are given in Watson et al. [1997].

6. THE END GAME

The projective transformation eliminates diverging paths, but in doing so may give paths leading to highly singular solutions at infinity. When the curve being tracked converges to a multiple solution or a positive dimensional solution set of $F(z) = 0$ at $\lambda = 1$, necessarily rank $D\rho(\lambda, z) < n$, which affects both numerical stability and the rate of convergence of the corrector iteration (11). Newton-type algorithms may do very well with nonsingular solutions, but incur a significant expense at singular solutions, an order of magnitude worse than at nonsingular solutions. This section describes one way that reasonably accurate estimates of a singular solution may be obtained at a fairly low computational cost (compared to Newton-type algorithms). The algorithm proposed here is based on that in Morgan et al. [1992b], but differs in several important aspects.

Define $h : D_0 \times D \to \mathbf{C}^n$ by $h(t, z) = (1 - t)f(z) + tg(z)$, where $f(z)$ and $g(z)$ are polynomial systems, with $D_0 \times D \subset \mathbf{C} \times \mathbf{C}^n$ open and $D_0 \supset [0, 1]$. Assume

(1) $z^*, \bar{z} \in D$ with $h(0, z^*) = f(z^*) = 0$ and $h(1, \bar{z}) = g(\bar{z}) = 0$;

(2) $h^{-1}(0)$ contains a connected complex curve $K \subset D_0 \times D$ containing z^* and $\bar{z}$, so that there is a smooth path $z(t)$ with $t \in [0, 1]$ and $z([0, 1]) \subset K$ such that $z(1) = \bar{z}$, $z(0) = z^*$, and $Dh(t, z(t))$ has rank n for $t \in (0, 1]$.

THEOREM 6.1 [MORGAN ET AL. 1992B]. *There is a $\delta > 0$, a smallest positive integer c, and a power series*

$$Z(\sigma) = \sum_{k=0}^{\infty} a_k \sigma^k$$

convergent for $|\sigma| < \delta$ such that

$$Z(\sigma) = z(\sigma^c) \tag{13}$$

for $\sigma \in [0, \delta)$.

Remark 6.1.1 The integer c is called the *cycle number* of the curve $z(t)$ and is defined in Morgan et al. [1991]. If z^* is a geometrically isolated solution to $f(z) = 0$ that has multiplicity m, then $c \leq m$.

Remark 6.1.2 As in Theorem 3.1, the change of variables $\lambda = 1 - t$ recasts the theorem into the notation of this article.

Remark 6.1.3 The following definition is useful [Morgan et al. 1992b]: σ is said to be in the *operating range* if $|\sigma|$ is small enough so that Theorem 6.1 holds and large enough so that the numerical process described below is not overwhelmed by ill conditioning. In many cases this annulus is large enough to work in. It is, however, possible that it will be empty. The size of the operating range annulus depends on the machine precision.

Remark 6.1.4 It is important to note that even though $z(\sigma^c)$ from Theorem 6.1 is defined only for real values of σ, $Z(\sigma)$ is defined and analytic for all $\sigma \in \mathbf{C}$, $|\sigma| < \delta$. Thus, as pointed out in Morgan et al. [1992b], the analyticity of h and Z give

$$h(\sigma^c, Z(\sigma)) = 0$$

for all σ such that $|\sigma| < \delta$. This fact suggests that samples of the homotopy zero path $Z(\sigma)$ can be taken at both positive and negative real values of σ, or even at complex σ in a neighborhood of $\sigma = 0$. Morgan et al. [1992b] use the latter for an end game based on complex contour integration.

The following example from Morgan et al. [1995] nicely illustrates the theorem: let $f(z) = z^m$, so that the solution $z^* = 0$ has multiplicity m. Let

$$h(t, z) = t(z^m - 1) + (1 - t)z^m.$$

Then the zero paths of the homotopy map h are given by (here $i = \sqrt{-1}$)

$$z(t) = e^{i(j/m)} \sqrt[m]{t},$$

for $j = 0, 1, 2, \ldots, m - 1$. Here $c = m$, $a_0 = 0$, $a_1 = e^{i(j/m)}$, and $\sigma^m = t$, i.e.,

$$Z(\sigma) = e^{i(j/m)}\sigma.$$

The notation $Z(\sigma)$ tacitly assumes that the change of variables $t = \sigma^c$ has taken place. Practical computation with $Z(\sigma)$ is complicated, because neither the value of c nor the size and location of the operating range will be immediately apparent.

The strategy in POLSYS_PLP is to track a zero curve γ of ρ (using the normal flow algorithm) to $\lambda > 1 - \epsilon$, where $0 < \epsilon \ll 1$, and then enter the end game. Since neither the value of c nor the location of the operating range is known, both must be determined. Once they are found, samples of $Z(\sigma)$ using (13) can be taken at real positive and negative values of σ. Then an interpolant to $Z(\sigma)$ can be used to approximate $z^* = Z(0)$. Morgan et al. [1992b] interpolate the even function $A(\sigma) = (Z(\sigma) + Z(-\sigma))/2$, and then approximate the root by estimating $z^* = A(0) = Z(0)$; this requires more Jacobian matrix evaluations than using just $Z(\sigma)$ (because Z must be evaluated at *exactly* σ and $-\sigma$), and typically only reduces the error slightly. The algorithm, inspired by one in Morgan et al. [1992b], follows:

Algorithm.

```
Given $\hat{c}_{max}$, $tol_1$, $tol_2$, $big \gg 1$, a point $(\lambda_1, z(\lambda_1))$ on $\gamma$ with $\lambda_1 \le 1 - \epsilon$, and a
point $(\lambda_0, z(\lambda_0))$ on $\gamma$ with $1 - \epsilon < \lambda_0 < 1$.
begin $hold := big$;
main loop: do
  do
    The points $P_\lambda^{(i)} = (\lambda_i, z(\lambda_i))$ for $i = 1, 0$ and the corresponding deriva-
    tives $dP_\lambda^{(i)} = dz/d\lambda(\lambda_i)$, where $\lambda_1 < \lambda_0$, have been found.
    $P_\lambda^{(2)} := P_\lambda^{(1)}$; $dP_\lambda^{(2)} := dP_\lambda^{(1)}$; $P_\lambda^{(1)} := P_\lambda^{(0)}$; $dP_\lambda^{(1)} := dP_\lambda^{(0)}$; $\lambda_2 := \lambda_1$; $\lambda_1 := \lambda_0$;
    Using the curve tracker, get one more point $P_\lambda^{(0)}$ and derivative $dP_\lambda^{(0)}$ at
    $\lambda_0$, where $\lambda_1 < \lambda_0 \approx \lambda_1 + 0.75(1 - \lambda_1) < 1$.
    if $D_z\rho(P_\lambda^{(0)})$ numerically singular then
      Flag convergence failure.
      exit main loop
    endif
    for $\hat{c} := 1$ step 1 until $\hat{c}_{max}$ do
      Change variables from $\lambda$ to $\sigma$ via
      $P_\sigma^{(i)} = (\sigma_i, Z(\sigma_i)) = ((1 - \lambda_i)^{1/\hat{c}}, z(\lambda_i))$ and
      $dP_\sigma^{(i)} = dZ/d\sigma(\sigma_i) = -\hat{c}\sigma^{\hat{c}-1}dz/d\lambda(\lambda_i)$.
      Construct a fourth-order Hermite interpolant (in $\sigma$) using $P_\sigma^{(i)}$ and
      $dP_\sigma^{(i)}$ for $i = 2, 1$, and a sixth-order Hermite interpolant using $P_\sigma^{(i)}$
      and $dP_\sigma^{(i)}$ for $i = 2, 1, 0$, obtaining the respective approximations $z^*$
      and $z^{**}$ at $\sigma = 0$.
      $test(\hat{c}) := \|z^* - z^{**}\|_\infty/(1 + \|z^{**}\|_\infty)$;
    enddo
    $err_1 :=$ minval($test(1 : \hat{c}_{max})$);
    $c :=$ minloc($test(1 : \hat{c}_{max})$);
    if $err_1 \le tol_1 * 10^{c/2}$ then exit
  enddo
  Convert path samples to variable $\sigma$, presuming that $c$ is the correct cycle
  number, and that the samples are taken from the operating range.
  for $j := 1$ step 1 until 3 do
    Use the samples (of $Z(\sigma)$) $P_\sigma^{(i)}$ and $dP_\sigma^{(i)}$ for $i = 2, \ldots, 1 - j$ to con-
    struct a $2(2 + j)$th-order Hermite interpolant. Use this interpolant to
    approximate $Z(\sigma)$ at $\sigma = -\sigma_{j-1}$.
    Apply the corrector iteration (11) yielding $\sigma_{-j}$, $P_\sigma^{(-j)}$, and $dP_\sigma^{(-j)}$ (after
    appropriate changes of variables).
    if $D_z\rho(P_\lambda^{(-j)})$ numerically singular then
      Flag convergence failure.
      exit main loop
    endif
  enddo
  Construct a 12th-order interpolant $H_{12}(\sigma)$ using $P_\sigma^{(i)}$ and $dP_\sigma^{(i)}$ for $i = 2,
  \ldots, -3$, construct a 10th-order interpolant $H_{10}(\sigma)$ using $P_\sigma^{(i)}$ and $dP_\sigma^{(i)}$ for
  $i = 2, \ldots, -2$, construct an 8th-order interpolant $H_8(\sigma)$ using $P_\sigma^{(i)}$ and
  $dP_\sigma^{(i)}$ for $i = 1, \ldots, -2$, construct a 6th-order interpolant $H_6(\sigma)$ using $P_\sigma^{(i)}$
  and $dP_\sigma^{(i)}$ for $i = 1, \ldots, -1$, and obtain an approximation $z^* = H_{12}(0)$ of
  the power series at $\sigma = 0$.
```

```
    $gm := \sqrt{(tol_1 * 10^{c/2}) * (tol_2 * 10^{c-1})}$;
    $err_2 := \|H_{10}(0) - H_{12}(0)\|_\infty / (1 + \|z^*\|_\infty)$ ;
    if $\left( \frac{\|H_8(0) - H_6(0)\|_\infty}{1 + \|H_{12}(0)\|_\infty} \le tol_1 * 10^{c/2}$ and $\frac{\|H_{10}(0) - H_8(0)\|_\infty}{1 + \|H_{12}(0)\|_\infty} \le gm$
       and $\frac{\|H_{12}(0) - H_{10}(0)\|_\infty}{1 + \|H_{12}(0)\|_\infty} \le tol_2 * 10^{c-1} \right)$ then
       exit main loop
    else
       if $err_2 \le 1.01 * hold$ then
          $hold := err_2$
       else
          Flag convergence failure and exit main loop if second occurrence
       endif
    endif
 enddo
```

The above pseudocode captures the spirit of the algorithm—continue iterating as long as progress is being made—but not the precise details. Difficulty portending failure is measured in several ways: failure of corrector iteration to converge, predicted λ value inconsistent with parity of c, change in predicted c after sampling $Z(\sigma)$ for $\sigma < 0$, increase in err_2 after main loop iteration. These failures are counted, and any two consecutive failures (not separated by a successful, logically consistent iteration) cause the whole algorithm to abort. POLSYS_PLP uses the parameter values $\hat{c}_{max} = 8$, $\epsilon = 0.97$, $tol_1 = 10^{-6}$, and $tol_2 =$ FINALTOL, an input parameter indicating the final accuracy desired. For any given machine precision there are theoretical limitations, which Morgan et al. [1992b] discuss, on the maximum cycle number for which the algorithm is still useful. POLSYS_PLP with 64-bit IEEE Standard 754 arithmetic has had success computing solutions with cycle numbers up to 6 and multiplicity 30. As Morgan et al. [1992b] demonstrate, the true cycle number c will be evident provided $\hat{c}_{max} \ge c$. ϵ and tol_1 are chosen based solely on computational experience. ϵ should not be so small that the path tracker encounters numerical instability from the singularity, and it should not be so large that the end game requires a large number of iterations before reaching the operating range. tol_1 must be small enough that the correct cycle number will be chosen—time would be wasted by computing points $P_\sigma^{(j)}$ for $\sigma < 0$ with an incorrect cycle number prediction—but not so stringent that the process leaves the operating range because $err_1 < tol_1 * 10^{c/2}$ is never satisfied. tol_2 is the desired accuracy of the solution. For solutions with large cycle numbers, say for $c > 3$, $err_2 < tol_2 * 10^{c-1}$ may not be achievable. In such cases, the algorithm simply returns with the last best estimate of z^*, which is often still very reasonable. Section 8 discusses the numerical performance of the end game in POLSYS_PLP.

7. ORGANIZATION AND USAGE

The package POLSYS_PLP consists of two Fortran 90 modules (`GLOBAL_PLP`, `POLSYS`). `GLOBAL_PLP` contains the Fortran 90 derived data types that define the target system, the start system, and the system partition. As its name suggests, `GLOBAL_PLP` provides data globally to the routines in POLSYS_PLP. The module `POLSYS` contains three subroutines: `POLSYS_PLP`, `BEZOUT_PLP`, and `SINGSYS_PLP`. `POLSYS_PLP` finds the root count (the Bezout number B_{PLP} for a given system partition P) and the roots of a polynomial system; `BEZOUT_PLP` finds only the root count; `SINGSYS_PLP` checks the singularity of a given start subsystem, and is of interest only to expert users. The package uses the HOMPACK90 modules `REAL_PRECISION`, `HOMPACK90_GLOBAL`, and `HOMOTOPY` [Watson et al. 1997], the HOMPACK90 subroutine `STEPNX`, and numerous LAPACK and BLAS subroutines [Anderson et al. 1995]. The physical organization of POLSYS_PLP into files is described in a README file that comes with the distribution.

Arguments to `POLSYS_PLP` include an input tracking tolerance `TRACKTOL`, an input final solution error tolerance `FINALTOL`, an input singularity tolerance `SINGTOL` for the root counting algorithm, input parameters for curve tracking, various output solution statistics, and four Fortran 90 optional arguments: `NUMRR`, `RECALL`, `NO_SCALING`, and `USER_F_DF`. The integer `NUMRR` specifies the number of iterations times 1000 that the path tracker is allowed; the default value is 1. The logical variable `RECALL` should be included if, after the first call, `POLSYS_PLP` is being called again to retrack a selected set of curves. The presence of the logical variable `NO_SCALING` (regardless of value) causes `POLSYS_PLP` *not* to scale the target polynomial system. The logical optional argument `USER_F_DF` specifies that the user is supplying hand-crafted code for function and Jacobian matrix evaluation—this option is recommended if efficiency is a concern, or if the original formulation of the system is other than a linear combination of monomials.

POLSYS_PLP takes full advantage of Fortran 90 features. For example, all real and complex type declarations use the KIND specification; derived data types are used for storage flexibility and simplicity; array sections, automatic arrays, and allocatable arrays are fully utilized; interface blocks are used consistently; where appropriate, modules, rather than subroutine argument lists, are used for data association; low-level linear algebra is done with Fortran 90 syntax rather than with BLAS routines; internal subroutines are used extensively with most arguments available via host association. POLSYS_PLP is easy to use, with a short argument list, and the target system $F(z)$ defined with a simple tableau format (unless the optional argument `USER_F_DF` is present). The calling program requires the statement

```
USE POLSYS
```

The typical use of POLSYS_PLP is either to call `BEZOUT_PLP` to obtain the root count B_{PLP} of a polynomial system of equations for a specified system partition P, or to call POLSYS_PLP to obtain all the roots of the polynomial (and the root count as a by-product). It is advisable to explore several system partitions with BEZOUT_PLP before committing to one and calling POLSYS_PLP. A sample main program MAIN_TEMPLATE demonstrates how to use POLSYS_PLP as just described. MAIN_TEMPLATE uses NAMELIST input for the target system and partition definitions, and allows the user to solve multiple polynomial systems in a single run.

The template TARGET_SYSTEM_USER (an external subroutine) is also provided. This subroutine would contain hand-crafted code for function and Jacobian matrix evaluation if the optional argument USER_F_DF to POLSYS_PLP were used.

The system partition must be defined by the user in the module GLOBAL_PLP. Heuristics, as in PHCPACK, exist for estimating an optimal system partition (PLP structure), but are no substitute for physical insight into the problem at hand. In practice, polynomial systems typically arise as sums of products with physical variables naturally grouped. Matching the PLP structure to the problem's "physical" structure usually yields a nearly optimal Bezout number B_{PLP}. Intuitively, the idea is to get all the degrees d_{ij} as low as possible (see the example below). For real problems, an m-homogeneous partition almost always suffices, and for the remainder a PLP structure is adequate. Of course LPD and GPD Bezout numbers can be lower than B_{PLP}, but no class of applications has yet emerged for which B_{LPD} or B_{GPD} are significantly lower than B_{PLP}.

7.1 An Example

The *Boon* problem [Verschelde 1997] from the field of neurophysiology demonstrates the value of exploiting structure and the applicability of POLSYS_PLP. Define $F : \mathbf{C}^6 \to \mathbf{C}^6$ by

$$F(z) = \begin{pmatrix} z_1^2 + z_3^2 - 1.0 \\ z_2^2 + z_4^2 - 1.0 \\ z_5 z_3^3 + z_6 z_4^3 - 1.2 \\ z_5 z_1^3 + z_6 z_2^3 - 1.2 \\ z_5 z_3^2 z_1 + z_6 z_4^2 z_2 - 0.7 \\ z_5 z_3 z_1^2 + z_6 z_4 z_2^2 - 0.7 \end{pmatrix}.$$

Thus the total degree of F is $d = 1024$. Consider the following system partition for F:

$$P = \Big\{\big\{\{1, 3\}, \{2, 4, 5, 6\}\big\}, \big\{\{1, 3, 5, 6\}, \{2, 4\}\big\}, \big\{\{1, 2\}, \{3, 4\}, \{5, 6\}\big\}^4\Big\},$$

which would be input for POLSYS_PLP along with the coefficients and degrees of the variables for each term in each component of $F(z)$. The subroutine POLSYS_PLP then computes the degree structure

$$d_{11} = 2, d_{12} = 0;$$

$$d_{21} = 0, d_{22} = 2;$$

$$d_{31} = 0, d_{32} = 3, d_{33} = 1;$$

$$d_{41} = 3, d_{42} = 0, d_{43} = 1;$$

$$d_{51} = 1, d_{52} = 2, d_{53} = 1;$$

$$d_{61} = 2, d_{62} = 1, d_{63} = 1.$$

These degrees are used with P to compute the start system $G(z) = 0$ defined in (4), which has exactly 216 nonsingular solutions (the PLP Bezout number B_{PLP}). Then $F(z)$ has at most 216 isolated solutions, found by tracking 216 homotopy zero curves starting at the roots of G. For comparison, the best m-homogeneous Bezout number, derived from the partition

$$\Big\{\{1, 2\}, \{3, 4\}, \{5, 6\}\Big\},$$

is 344.

8. PERFORMANCE

This section discusses implementation issues and numerical performance of the root counting algorithm of Section 3, and of the end game algorithm in Section 6. Root counting is done in the subroutines `BEZOUT_PLP` and `SINGSYS_PLP`, the former calling the latter. The end game is housed in an internal subroutine `ROOT_PLP`.

8.1 Root Counting

The danger in implementing a root counting algorithm with floating-point arithmetic, rather than as an exact integer computation, is that a generically nonsingular A_Φ may be classified singular, and vice versa. Structurally singular A_Φ will always be ill conditioned, so the issue reduces to the likelihood that a generically invertible A_Φ will also be ill conditioned, and hence misclassified. Different ways for choosing the start system coefficients c_{ijk} have been tested by accumulating statistics on cond A_Φ.

Observe that the literature on random dense matrices is not directly applicable here, because the matrices A_Φ are typically very sparse and very structured (which of course is the whole point of the PLP structure). Thus, rather than generating random sparse matrices, a better test is to take a few nontrivial PLP structures, generate large numbers of A_Φ for those structures, and observe the distribution of cond A_Φ. Results for three representative target polynomial systems $F^{(1)}$, $F^{(2)}$, $F^{(3)}$ are reported here;

two of these are *cyclic4* and *cyclic8* from Björk and Fröberg [1991]. Relevant data for these three systems follow. For $F^{(1)}$: $n = 8$,

$$P = \Big\{\big\{\{1, 2\}, \{3, 4\}, \{5, 6\}, \{7, 8\}\big\}^{8}\Big\};$$

for $F^{(2)}$ (*cyclic4*): $n = 4$,

$$P = \Big\{\big\{\{1, 2, 3, 4\}\big\}, \big\{\{1, 3\}, \{2, 4\}\big\}, \big\{\{1\}, \{2\}, \{3\}, \{4\}\big\}^{2}\Big\};$$

for $F^{(3)}$ (*cyclic8*): $n = 8$,

$$P = \Big\{\big\{\{1, 2, 3, 4, 5, 6, 7, 8\}\big\}, \big\{\{1, 3, 5, 7\}, \{2, 4, 6, 8\}\big\},$$

$$\big\{\{1, 5\}, \{2, 6\}, \{3, 7\}, \{4, 8\}\big\}^{2}, \big\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{8\}\big\}^{4}\Big\}.$$

Random c_{ijk} were chosen from the unit square in $\mathbf{C}$ and $[-1, -1/2] \cup [1/2,1] \subset \mathbf{R}$. Using only real c_{ijk} yielded the same qualitative results as complex c_{ijk}, and of course is considerably cheaper. From a PLU factorization of each A_Φ, $\mu = \min_{1 \le i \le n} |U_{ii}|$ was computed. Let $r = \lceil \log_{10} \mu \rceil$ for $\mu \neq 0$ and $r = -100$ for $\mu = 0$. $r = -7$ corresponds to μ being the square root of machine precision for 64-bit IEEE arithmetic. If A_Φ was declared singular for $r < -7$, then A_Φ was correctly identified as either structurally singular or generically nonsingular in every single instance.

Figure 1 shows histograms for r for $F^{(1)}$, $F^{(2)}$, $F^{(3)}$. The point made emphatically by Figure 1 is that the invertible and singular A_Φ are clearly separated, with only a handful of matrices even approaching the cutoff value. Though not guaranteed, it is extremely unlikely that POLSYS_PLP will misclassify a matrix A_Φ.

For consistency with HOMPACK90, and because for poorly scaled systems accuracy is important, POLSYS_PLP uses QR factorizations for both the root count algorithm (`BEZOUT_PLP`) and the solution of the start subsystems to get the start points for the homotopy map (internal subroutine `START_POINTS_PLP`). Since both of these calculations proceed in lexicographic order, the QR factorization for one lexicographic vector can be efficiently updated for the next lexicographic vector. Here updating means simply replacing the tail of a sequence of Householder reflections with different reflections. Using QR rather than PLU to classify A_Φ can only improve the classification algorithm. The pseudocode

$$\textbf{if } A_\Phi^T(j, j) \approx 0 \textbf{ then}$$

in the root counting algorithm is actually the test

```
IF (ABS(A_PHI(J, J)) <= SINGTOL) THEN
```

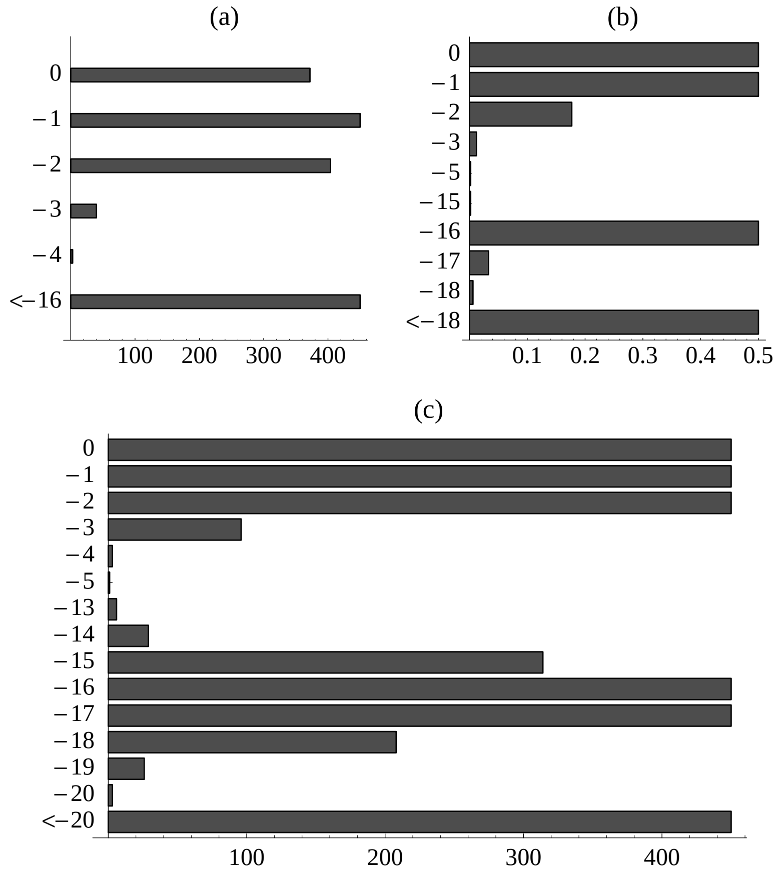

Fig. 1. Histograms of r frequencies. (a) $F^{(1)}$, 4^8 matrices; (b) $F^{(2)}$, 32 matrices (frequencies averaged over 1000 runs); (c) $F^{(3)}$, 131,072 matrices.

where SINGTOL is an argument to SINGSYS_PLP, and is set to the square root of machine precision by default.

8.2 End Game

This section discusses the performance of the internal subroutine ROOT_PLP, the end game algorithm of Section 6. Three problems having singular solutions are considered: PB601 [Morgan and Watson 1989] is very hard, and PB801 and PB803 [Morgan et al. 1992b] are of moderate difficulty. Performance of the end game subroutine ROOTNX of HOMPACK90 [Watson et al. 1997] is compared to that of ROOT_PLP (cf. Table I). For problems PB601, PB801, and PB803, TRACKTOL was 10^{-4}, 10^{-4}, 10^{-3} respectively, and FINALTOL was 10^{-14}, 10^{-12}, 10^{-12} respectively. Scaling was enabled for all problems, and the polynomials were evaluated from their coefficient tableaus rather than hand-written code. Recall for the ensuing discussion that there is no way to know *a priori* which curves will lead to nonsingular solutions.

Table I. Path Tracking Statistics

	Path Characteristics	Average Number of Function Evaluations			ROOTNX
		Tracking	ROOT_PLP	ROOTNX	Failures
PB601	42 paths, $\hat{c} = 6$	88	154	102	26
	3 paths, $\hat{c} = 3$	39	98	36	0
	15 paths, $\hat{c} = 1$	68	93	50	0
PB801	113 paths, $\hat{c} = 2$	117	48	2191	82
	15 paths, $\hat{c} = 1$	85	47	19	0
PB803	32 paths, $\hat{c} = 2$	114	35	59	32
	64 paths, $\hat{c} = 1$	121	22	14	16

There are three possibilities: use ROOTNX always, use a hybrid scheme applying ROOTNX to nonsingular solutions and ROOT_PLP to singular solutions, or use ROOT_PLP always. ROOTNX performs better at nonsingular solutions than ROOT_PLP. Used alone, ROOTNX can be much more expensive overall than ROOT_PLP, its good performance at nonsingular solutions overshadowed by its extremely poor performance at singular solutions. ROOT_PLP handles singular solutions much better than ROOTNX. The cost of ROOTNX can be much higher than ROOT_PLP at singular solutions (see PB801 in Table I). Moreover, whereas ROOTNX could rarely find even a poor approximation of a singular root, ROOT_PLP obtained good approximations, even for the multiplicity 30 solution of PB601. ROOTNX failed completely on many paths (even on some paths for which $\hat{c} = 1$), but ROOT_PLP never failed. Data for PB803 and PB801 show that the overall cost of using ROOT_PLP alone can be far less than the cost of using ROOTNX alone. Thus using ROOTNX alone (as POLSYS1H of HOMPACK90 does) is not viable.

Early versions of POLSYS_PLP used a hybrid end game: if a curve appeared to be going to a singular solution, ROOT_PLP was chosen; otherwise ROOTNX was used, with reversion to ROOT_PLP if ROOTNX failed. Various criteria such as condition number estimates and step size reduction were tried. The hybrid schemes proved unsuccessful, since curves were frequently misdiagnosed: ROOTNX would be called but would then fail because the curve was actually going to a singular solution, and ROOT_PLP was sometimes called when a curve was converging to a nonsingular solution. Essentially, any failure of ROOTNX is more costly than using ROOT_PLP alone on that root. Despite the appeal of hybrid methods, the evidence seems to support using ROOT_PLP only on all roots.

ACKNOWLEDGMENTS

The authors are deeply indebted to A. P. Morgan and C. W. Wampler for fundamental theoretical contributions upon which this work is based, and for graciously providing insight and guidance for the development of industrial-strength software.

REFERENCES

Allgower, E. L. and Georg, K. 1990. *Numerical Continuation Methods*. Springer-Verlag, Berlin, Germany.

Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., DuCroz, J., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S., and Sorensen, D. 1995. *LAPACK Users' Guide*. 2nd ed. SIAM, Philadelphia, PA.

Björck, G. and Fröberg, R. 1991. A faster way to count the solutions of inhomogeneous systems of algebraic equations, with applications to cyclic n-roots. *J. Symb. Comput. 12*, 3 (Sept. 1991), 329–336.

Blum, L., Cucker, F., Shub, M., and Smale, S. 1998. *Complexity and Real Computation*. Springer-Verlag, Secaucus, NJ.

Chow, S. N., Mallet-Paret, J., and Yorke, J. A. 1979. A homotopy method for locating all zeros of a system of polynomials. In *Functional Differential Equations and Approximation of Fixed Points*, H. O. Peitgen and H. O. Walther, Eds. Springer Lecture Notes in Mathematics, vol. 730. Springer-Verlag, New York, NY, 228–237.

Drexler, F. J. 1979. Eine methode zur berechung sämtlicher lösunger von polynomgleichungessystemen. *Numer. Math. 29*, 1, 45–58.

Garcia, C. B. and Zangwill, W. I. 1977. Global continuation methods for finding all solutions to polynomial systems of equations in N variables. Center for Math Studies in Business and Economics Rep. 7755. University of Chicago, Chicago, IL.

Griewank, A. 1985. On solving nonlinear equations with simple singularities or nearly singular solutions. *SIAM Rev. 27*, 4, 537–563.

Lundberg, B. N. and Poore, A. B. 1991. Variable order Adams-Bashforth predictors with an error-stepsize control for continuation methods. *SIAM J. Sci. Stat. Comput. 12*, 3 (May 1991), 695–723.

Meintjes, K. and Morgan, A. P 1987. A methodology for solving chemical equilibrium systems. *Appl. Math. Comput. 22*, 4 (June 1987), 333–361.

Morgan, A. P. 1983. A method for computing all solutions to systems of polynomial equations. *ACM Trans. Math. Softw. 9*, 1 (Mar.), 1–17.

Morgan, A. P. 1986a. A transformation to avoid solutions at infinity for polynomial systems. *Appl. Math. Comput. 18*, 1, 77–86.

Morgan, A. P. 1986b. A homotopy for solving polynomial systems. *Appl. Math. Comput. 18*, 1, 87–92.

Morgan, A. P. 1987. *Solving Polynomial Systems Using Continuation for Engineering and Scientific Problems*. Prentice-Hall, Inc., Upper Saddle River, NJ.

Morgan, A. and Sommese, A. 1987a. A homotopy for solving general polynomial systems that respects m-homogenous structures. *Appl. Math. Comput. 24*, 2 (Nov. 1987), 101–113.

Morgan, A. and Sommese, A. 1987b. Computing all solutions to polynomial systems using homotopy continuation. *Appl. Math. Comput. 24*, 2 (Nov. 1987), 115–138.

Morgan, A. P. and Watson, L. T. 1989. A globally convergent parallel algorithm for zeros of polynomial systems. *Non-Linear Anal. 13*, 11 (Nov. 1989), 1339–1352.

Morgan, A. P., Sommese, A. J., and Wampler, C. W. 1991. Computing singular solutions to nonlinear analytic systems. *Numer. Math. 58*, 669–684.

Morgan, A. P., Sommese, A. J., and Wampler, C. W. 1992a. Computing singular solutions to polynomial systems. *Adv. Appl. Math. 13*, 3 (Sept. 1992), 305–327.

Morgan, A. P., Sommese, A. J., and Wampler, C. W. 1992b. A power series method for computing singular solutions to nonlinear analytic systems. *Numer. Math. 63*, 391–409.

Morgan, A. P., Sommese, A. J., and Wampler, C. W. 1995. A product-decomposition bound for Bezout numbers. *SIAM J. Numer. Anal. 32*, 4 (Aug. 1995), 1308–1325.

Morgan, A. P., Sommese, A. J., and Watson, L. T. 1989. Finding all isolated solutions to polynomial systems using HOMPACK. *ACM Trans. Math. Softw. 15*, 2 (June 1989), 93–122.

Sosonkina, M., Watson, L. T., and Stewart, D. E. 1996. Note on the end game in homotopy zero curve tracking. *ACM Trans. Math. Softw. 22*, 3 (Sept.), 281–287.

TSAI, L.-W. AND MORGAN, A. P. 1985. Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods. *ASME J. Mech. Trans. Automat. Des. 107*, 48–57.

VAN DER WAERDEN, B. L. 1953. *Modern Algebra*. Frederick Ungar, New York, NY. Volumes 1 and 2.

VERSCHELDE, J. 1996. Homotopy continuation methods for solving polynomial systems. Ph.D. Dissertation. Department of Computer Science, Katholieke Universiteit Leuven, Leuven, Belgium.

VERSCHELDE, J. 1997. PHCPACK: A general-purpose solver for polynomial systems by homotopy continuation. Preprint.

VERSCHELDE, J. AND COOLS, R. 1993. Symbolic homotopy construction. *Appl. Alg. Eng. Commun. Comput. 4*, 169–183.

VERSCHELDE, J. AND HAEGEMANS, A. 1993. The *GBQ*-algorithm for constructing start systems of homotopies for polynomial systems. *SIAM J. Numer. Anal. 30*, 2 (Apr. 1993), 583–594.

WAMPLER, C. W. 1994. An efficient start system for multi-homogeneous polynomial continuation. *Numer. Math. 66*, 4, 517–523.

WATSON, L. T., BILLUPS, S. C., AND MORGAN, A. P. 1987. ALGORITHM 652: HOMPACK: A suite of codes for globally convergent homotopy algorithms. *ACM Trans. Math. Softw. 13*, 3 (Sept. 1987), 281–310.

WATSON, L. T., SOSONKINA, M., MELVILLE, R. C., MORGAN, A. P., AND WALKER, H. F. 1997. Algorithm 777: HOMPACK90: A suite of Fortran 90 codes for globally convergent homotopy algorithms. *ACM Trans. Math. Softw. 23*, 4, 514–549.

WRIGHT, A. H. 1985. Finding all solutions to a system of polynomial equations. *Math. Comput. 44* (Jan. 1985), 125–133.

Received: August 1998; revised: November 1999 and December 1999; accepted: December 1999