Numerical Factorization of Multivariate Complex Polynomials

Andrew J. Sommese* Jan Verschelde† Charles W. Wampler‡

7 May 2003

Abstract

One can consider the problem of factoring multivariate complex polynomials as a special case of the decomposition of a pure dimensional solution set of a polynomial system into irreducible components. The importance and nature of this problem however justify a special treatment. We exploit the reduction to the univariate root finding problem as a way to sample the polynomial more efficiently, certify the decomposition with linear traces, and apply interpolation techniques to construct the irreducible factors. With a random combination of differentials we lower multiplicities and reduce to the regular case. Estimates on the location of the zeroes of the derivative of polynomials provide bounds on the required precision. We apply our software to study the singularities of Stewart-Gough platforms.

2000 Mathematics Subject Classification. Primary 13P05, 14Q99; Secondary 65H10, 68W30.

Key words and phrases. Approximate factorization, divided differences, generic points, homotopy continuation, irreducible decomposition, Newton interpolation, numerical algebraic geometry, monodromy, multiple roots, polynomial, Stewart-Gough platform, symbolic-numeric computation, traces, witness points.

1 Introduction

We consider a polynomial f with complex coefficients in several variables. We wish to write f as a product of irreducible polynomials:

$$f(\mathbf{x}) = \prod_{i=1}^{N} q_i(\mathbf{x})^{\mu_i}, \quad \mathbf{x} = (x_1, x_2, \ldots, x_n), \quad \sum_{i=1}^{N} \mu_i \deg(q_i) = \deg(f). \tag{1}$$

Note that every irreducible factor q_i occurs with multiplicity μ_i.

*Department of Mathematics, University of Notre Dame, Notre Dame, IN 46556-4618, USA. *E-mail:* sommese@nd.edu. *URL:* http://www.nd.edu/˜sommese.

†Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago, 851 South Morgan (M/C 249), Chicago, IL 60607-7045, USA. *E-mail:* jan@math.uic.edu or jan.verschelde@na-net.ornl.gov. *URL:* http://www.math.uic.edu/˜jan.

‡General Motors Research & Development, Mail Code 480-106-359, 30500 Mound Road, Warren, MI 48090-9055, USA. *E-mail:* Charles.W.Wampler@gm.com.

The problem of factoring multivariate polynomials occurs frequently in computer algebra. Especially the case when the coefficients of f are known only approximately is important for applications and is stated as a challenge problem to symbolic computation in [7]. Recent papers on this problem are [1, 2], [4, 5], [6], [16], [17]. These papers propose algorithms in hybrid symbolic-numeric computation [3]. We find our way of working very much related to the method of computing the approximate gcd of two polynomials using their zeros, as defined in [13].

Our approach is to specialize the tools we built for computing an irreducible decomposition of the solution set of a polynomial system to this special case. These tools were developed to implement the research program *Numerical Algebraic Geometry*, outlined in [27]. In [19] we gave algorithms to decompose solution sets of polynomial systems into irreducible components of various degrees and dimensions, applying an embedding and sequence of homotopies [18] to find generic points efficiently. The homotopy test presented in [20] to determine whether a given point belongs to an irreducible component led to the use of monodromy [21] for which linear traces [22] provide an efficient verification and interpolation method. Applications of our software [24] to design problems in mechanical engineering are described in [25]. A tutorial to our recent developments can be found in [26].

In [22] we gave an algorithm for using monodromy to decompose the zero set of a polynomial system into irreducible components. The main difficulty in the use of monodromy occurs when we track points on irreducible components of multiplicity at least two. In [23] we presented a method for tracking these paths, but it necessitates special care and usually requires higher precision arithmetic. In this article we specialize the algorithm of [22] to the case of a single polynomial $f(\mathbf{x})$ on $\mathbb{C}^n$. The main contribution of this paper is to show how to marry symbolic and numerical methods to deal with the difficulties posed by multiplicity at least two components.

A naive idea to lower the multiplicity is to differentiate. For example, if we had $f(\mathbf{x}) = (x_1^2 - x_2)^3(x_1^2 + x_2^2 + x_2^3)$, we could differentiate twice with respect to x_1 and we would obtain a polynomial where $x_1^2 - x_2$ is a factor of multiplicity one. Of course, there are a few obvious problems with this approach.

For simplicity, first assume that the polynomial is on $\mathbb{C}$. Data for polynomials arising in engineering and science is sometimes noisy. Also zeroes of polynomials of multiplicity more than one are hard to compute exactly. Thus even if the actual polynomial has a factor with multiplicity $\mu \geq 2$, we must expect that if we solve the restriction of the polynomial to a general line, we will find not a single zero of multiplicity μ, but a cluster of μ zeroes. Assume a polynomial has a zero z of multiplicity μ. Differentiating $\mu - 1$ times will yield a polynomial having a nonsingular factor $x - z$. Because of slightly perturbed coefficients or roundoff errors, we may have a nearby polynomial $f(x)$ containing a factor $\prod_{i=1}^{\mu}(x - z_i)$ with z_i near z. In contrast to the case of an exact multiple root, it does not follow that $f^{(\mu-1)}(x)$ has a single zero near z. Here a remarkable result of Marden and Walsh [8] (formulated as Corollary 3.3 below) gives mild numerical conditions guaranteeing the numerical stability of the symbolic operation of differentiation. It guarantees that if we have a cluster of μ roots in a disk D of radius r, and no root outside of D is a distance less than R from the center of the disk D, then under mild conditions on the size of R/r, $f^{(\mu-1)}(x)$ has one root in D, and a lower bound is given for the distance of any root of $f^{(\mu-1)}(z)$ outside of D to the center of D.

For polynomials of several variables, i.e., $\mathbf{x} \in \mathbb{C}^n$, we choose a general line and find the roots of the restriction of $f(\mathbf{x})$ to a line $\mathbf{x}(t) = \mathbf{x}_0 + t\mathbf{v}$ where $\mathbf{x}_0, \mathbf{v} \in \mathbb{C}^n$ are random vectors. For roots of multiplicity one, we can use the monodromy technique of [21], or if the degree of $f(\mathbf{x})$ is low, use the trace theorem of [22] to justify the partial sum approach of [16]. To deal with the clusters of μ roots we can compute the $(\mu-1)$-st derivative of $f(\mathbf{x})$ in the direction $\mathbf{v}$, i.e., with $\mathbf{v}=(v_1,\ldots,v_n)$, we compute

$$g(\mathbf{x}) := \left(v_1\frac{\partial}{\partial x_1}+\cdots+v_1\frac{\partial}{\partial x_1}\right)^{\mu-1} f(\mathbf{x}), \tag{2}$$

and apply the techniques to the multiplicity one roots of $g(\mathbf{x})$ corresponding to the clusters. Since

$$g(\mathbf{x}_0+t\mathbf{v}) = \left(\frac{d}{dt}\right)^{\mu-1} f(\mathbf{x}_0+t\mathbf{v}), \tag{3}$$

we can use Corollary 3.3 to check that $g(\mathbf{x}_0+t\mathbf{v})$ has multiplicity one roots corresponding to the multiplicity μ clusters. Now as we vary the line, we can use $g(\mathbf{x})$ to track the appropriate roots. As a numerical safety check we can check that the continuations of these roots on the line intersection with $g^{-1}(0)$ have small residual when we evaluate $f(\mathbf{x})$ at them.

In this paper we first outline the algorithms using pseudocode. Then we justify our use of differentiation to remove multiplicities and examine the implications of the results of Marden and Walsh for the behavior of root clusters under differentiation. After discussing some numerical aspects of our implementation, we apply our software to a problem from mechanical engineering concerning the singularities of Stewart-Gough platforms.

Acknowledgments. We gratefully acknowledge the support of this work by Volkswagen-Stiftung (RiP-program at Oberwolfach). The first author thanks the Duncan Chair of the University of Notre Dame and National Science Foundation. This material is based upon work supported by the National Science Foundation under Grant No. 0105653. The second author thanks the Department of Mathematics of the University of Illinois at Chicago and National Science Foundation. This material is based upon work supported by the National Science Foundation under Grant No. 0105739 and Grant No. 0134611. The third author thanks General Motors Research and Development for their support.

2 Algorithms

Given a pure k-dimensional affine algebraic set Z, we use the term "witness point set" to designate the intersections of Z with a generic linear space L_{N-k} of complementary dimension $N-k$. (See, for example, [19].) L_{N-k} can be defined by k linear equations on $\mathbb{C}^N$, each having random, complex coefficients, or equivalently, it can be given in parametric form as $\mathbf{x}=\mathbf{x}_0+\sum_{i=1}^{N-k}\mathbf{v}_i t_i$, where $\mathbf{x}_0,\mathbf{v}_i\in\mathbb{C}^N$ are random and complex. In the case at hand, Z is a hypersurface given by a single polynomial equation $f(\mathbf{x})=0$, so we intersect it with a one-dimensional linear space, L_1.

In the initial stage we compute a set of witness points on the hypersurface defined by $f(\mathbf{x})=0$ and store clustered points according to the size of the cluster. More precisely, if $d=\deg(f)$, **WitnessGenerate** computes d witness points. and partitions the set of witness points into $W=\{W_1,W_2,\ldots,W_m\}$, where each W_i is a set of clusters of size i.

Algorithm 2.1 $W=$ **WitnessGenerate**$(f,\mathbf{x}_0,\mathbf{v})$

Input : $f(\mathbf{x})$ polynomial in n variables with complex coefficients;
$\mathbf{x}_0$ and $\mathbf{v}$ represent a random line $\mathbf{x}(t)=\mathbf{x}_0+t\mathbf{v}$.

Output : $W=\{W_1,W_2,\ldots,W_m\}$, $m=\max_{i=1}^{d}\mu_i$, for all $X\in W_i$: $\#X=\mu_i$.

The method to solve a polynomial in one variable is invoked once in **WitnessGenerate**. The algorithm **RegularFactor** assumes the witness points all have multiplicity one. It is invoked repeatedly in the main factorization algorithm.

Algorithm 2.2 $P = \mathbf{RegularFactor}(f, \mathbf{x}_0, \mathbf{v}, W_1)$

Input : $f(\mathbf{x})$ polynomial in n variables with complex coefficients;
$\mathbf{x}_0$ and $\mathbf{v}$ represent a random line $\mathbf{x}(t) = \mathbf{x}_0 + t\mathbf{v}$;
W_1 set of t-values for which $f(\mathbf{x}(t)) = 0$, with multiplicity one.

Output : $P = \{p_1, p_2, \ldots, p_k\}$, k irreducible factors of f with $\displaystyle\sum_{i=1}^{k} \deg(p_i) = \#W_1$.

We have two different implementations of **RegularFactor**:

1. Using the monodromy grouping algorithm [21], certified with linear traces [22].
2. Applying linear traces to enumerated factors [4, 5], [16].

Both methods apply path-following techniques. In each step of the path tracker we slightly move the random line, predict the location of the solutions and feed the predicted solutions to Newton's method for correction.

The main difference between the two implementations lies in the number of computed samples. With monodromy we compute witness point sets on many random lines, and take the witness points connected by paths as belonging to the same irreducible factor. Linear traces are then applied to certify the factorization predicted by the monodromy. In the enumeration method, we also use linear traces, but plainly enumerate all possible factorizations. For 3 witness points, the enumeration method runs as follows:

```
Is 1 a factor?
 |- Yes: Is 2 a factor?
 |    |- Yes: 1,2,3 is the factorization
 |    |- No: 1,23 is the factorization
 |- No: Is 12 a factor?
      |- Yes: 12,3 is the factorization
      |- No: is 13 a factor?
          |- Yes: 13,2 is the factorization
          |- No: 123 is the factorization
```

Each test is answered by the computation of a linear trace and comparing the value at the linear trace with the sum directly computed from the samples. The largest number of tests in this algorithm occurs when the factor witnessed by W_1 is irreducible, and equals $2^{w-1} - 1$, where $w = \#W_1$.

After the grouping of the witness points along the irreducible factors, we can apply interpolation techniques to find symbolic expressions for the polynomials. In our implementation we postponed all interpolation to the end, because this stage is most time consuming and sometimes also not really necessary.

After **WitnessGenerate** and **RegularFactor** we have all irreducible factors of multiplicity one. To build the higher multiplicity factors, we propose to take random combinations of all partial derivatives. Each differentiation cuts the multiplicity by one and the number of solutions in the cluster drops accordingly. To process W_i, f is differentiated $i-1$ times yielding $g := D^{(i-1)}f$. The routine **RefineCluster** takes the center of each clustered set of W_i as an initial approximation for root refinement with g. Thereafter we can apply **RegularFactor** on g and the reduced set W_i as before.

Algorithm 2.3 $Q = \mathbf{Factor}(f)$

Input : $f(\mathbf{x})$ polynomial in n variables with complex coefficients.
Output : $Q = \{ \ (q_i, \mu_i) \mid i = 1, 2, \ldots, N \ \}$, irreducible factors q_i with multiplicities μ_i.

$\mathbf{x}_0 := \mathbf{Random}(n, \mathbb{C})$;	[*choose n random numbers in $\mathbb{C}$*]
$\mathbf{v} := \mathbf{Random}(n, \mathbb{C})$;	[$\mathbf{v}$ *is direction of line* $\mathbf{x}(t) = \mathbf{x}_0 + t\mathbf{v}$]
$W := \mathbf{WitnessGenerate}(f, \mathbf{x}_0, \mathbf{v})$;	[*find witness points*]
$Q := \emptyset$;	[*Q will collect all factors*]
$P := \mathbf{RegularFactor}(f, \mathbf{x}_0, \mathbf{v}, W_1)$;	[*find regular factors*]
for all $p_i \in P$ do	
$Q := Q \cup (p_i, 1)$;	[*collect multiplicity one factors*]
end for;	
$D := \displaystyle\sum_{i=1}^{n} v_i \frac{\partial}{\partial x_i}$;	[*differential along direction* $\mathbf{v}$]
$g := f$;	[*g is $(\mu-1)$-th differential of f, $\mu = 1$*]
for $\mu = 2, 3, \ldots, \#W$ do	[*construct multiplicity μ factors*]
$g := Dg$;	[*differentiate so that $g = D^{(\mu-1)}f$*]
$W_i := \mathbf{RefineCluster}(g, \mathbf{x}_0, \mathbf{v}, W_i)$;	[*refine center of clusters*]
$P := \mathbf{RegularFactor}(g, \mathbf{x}_0, \mathbf{v}, W_i)$;	[*find regular factors*]
for all $p_i \in P$ do	
$Q := Q \cup (p_i, \mu)$;	[*collect multiplicity μ factors*]
end for;	
end for;	
return Q.	

With the witness points grouped according to the factors, we can apply interpolation methods, e.g. using traces [20], to obtain symbolic representations of the factors.

3 How clusters of zeroes spread out under differentiation

At the innermost level of our routine, we have reduced the problem to following a multiple root of a complex polynomial in one variable. Recall that if $h(z)$ has a root of multiplicity μ at a point z_0, then its derivative $h'(z)$ has a root of multiplicity $\mu - 1$ at z_0, and so $h^{(\mu-1)}(z)$ has a nonsingular root at z_0. While a nonsingular root is much easier to compute accurately than a multiple root, we must be concerned about the numerical stability of the differentiation. This problem manifests itself in the way that the roots away from z_0 move under differentiation. Ignoring the roots exactly at z_0, it may well happen that $h'(z)$ has at least one root closer to z_0 than any root of $h(z)$, and after $\mu - 1$ derivatives, some root may come so close to z_0 as to be numerically indistinguishable from z_0 itself. For a simple example showing the problem, consider $h(z) = z^\mu (z-1)^{d-\mu}$. The root $z = 0$ occurs with multiplicity μ, with the $d - \mu$ remaining roots at $z = 1$. The derivative $h'(z)$ has $z = 0$ as a root of multiplicity $\mu - 1$, but it also has a root $\frac{\mu}{d}$, which for large d can be near zero. If we will need to differentiate $\mu - 1$ times, this can lead to a serious problem.

The problem is further exacerbated if, due to numerical roundoff in its representation, we begin with $\widehat{h}(z)$, nearby to $h(z)$, having a cluster of μ roots near z_0. After differentiation, we would like to have an $\widehat{h}'(z)$ with a cluster of $\mu - 1$ roots near z_0 and all other roots far away, but as seen from the above example, this may not always be the case. The following result gives some bounds on the behavior of the roots under differentiation and helps in guiding the choice of how many digits of precision we should use in implementation of our algorithm. The result follows from a very special case of a beautiful, but somewhat intricate, classical result of Marden and Walsh [8, Theorem 21.1] about the geometry of the zeroes of the derivative of a polynomial. For the convenience of the reader, we include a self-contained proof of the result we need.

As a preliminary step, we derive a simple result on sums of complex numbers.

Lemma 3.1 *Let $u_1, \ldots, u_\mu$ denote $\mu > 0$ complex numbers satisfying $|u_i - \rho| < r$ where ρ and r are real numbers satisfying $\rho > r > 0$. Then*

$$\frac{\mu}{\rho + r} < \left| \sum_{i=1}^{\mu} \frac{1}{u_i} \right| .$$

Proof. Write each u_i in polar form, $r_i e^{\sqrt{-1}\theta_i}$. Since $\rho > r$, the real parts of each $\frac{1}{u_i}$ are positive, and we have from the triangle inequality that

$$\left| \sum_{i=1}^{\mu} \frac{1}{u_i} \right| \geq \sum_{i=1}^{\mu} \operatorname{Re}\left(\frac{1}{u_i} \right) = \sum_{i=1}^{\mu} \frac{1}{r_i} \cos(-\theta_i).$$

Note that for a fixed r_i, the smallest value of the positive number $\frac{1}{r_i}\cos(-\theta_i)$ for $|u_i - \rho| < r$ occurs at the boundary $|u_i - \rho| = r$. It is a simple calculus problem that the minimum of the real part of $\frac{1}{u}$ for u satisfying $|u - \rho| = r$, occurs when $u = \rho + r$. □

Theorem 3.2 (Marden and Walsh) *Let $h(z)$ be a degree d polynomial of one complex variable. Assume that $h(z)$ has μ zeroes in the disk $\Delta_r(z_0) := \{z \in \mathbb{C} \big| |z - z_0| \leq r\}$ and $d - \mu$ zeroes in the region $\{z \in \mathbb{C} \big| |z - z_0| \geq R\}$, where $R > r$. Then, if $\mu(R + r) > 2dr$, it follows that $h'(z)$ has $\mu - 1$ roots in $\Delta_r(z_0)$ and all the remaining $d - \mu$ roots in the region $\left\{ z \in \mathbb{C} \,\middle|\, |z - z_0| \geq \frac{\mu}{d}(R + r) - r \right\}$.*

Proof. By translation we can assume without loss of generality that $z_0 = 0$. In this case we abbreviate $\Delta_r(0)$ to Δ_r. Without loss of generality we assume that $h(z)$ is monic, i.e., that the highest order term of $h(z)$ is z^d.

Note that to prove the theorem, it is enough to prove the following assertion.

Claim 3.2.1 *Given a real number ρ satisfying $R > \rho > r$, it follows that if*

$$\rho < \frac{\mu}{d}(R + r) - r, \tag{4}$$

then h' has exactly $\mu - 1$ zeroes in the disk Δ_ρ.

The assumptions of Theorem 3.2 and Claim 3.2.1 imply that we may write $h(z) = p(z)q(z)$, where $p(z)$ is a monic polynomial of degree μ, which has the same μ zeroes with multiplicities, that $h(z)$ has in Δ_r, and $q(z)$ is a monic polynomial with all roots at least distance $R > \rho > r$ from the origin. The polynomial $p'(z)q(z)$ has the same zeroes in Δ_r as $p'(z)$. Therefore, it suffices to check, that for $\rho > r$ satisfying equation (4), $p'(z)q(z)$ and $h'(z)$ have the same number of zeroes counting multiplicities in Δ_ρ. By Rouché's Theorem, e.g., [8, page 2], we know that $p'(z)q(z)$ and $h'(z) = p'(z)q(z) + p(z)q'(z)$ have the same number of zeroes in Δ_ρ if

$$|p(z)q'(z)| < |p'(z)q(z)| \tag{5}$$

for z satisfying $|z| = \rho$. Therefore, to prove Claim 3.2.1, it suffices to show that equation (4) implies equation (5). Since $h(z)$ has no zeroes satisfying $|z| = \rho$, it suffices to show that equation (4) implies

$$\left|\frac{q'(z)}{q(z)}\right| < \left|\frac{p'(z)}{p(z)}\right|. \tag{6}$$

Letting $z_1, \ldots, z_\mu$ denote the zeroes of $p(z)$, each listed a number of times equal to its multiplicity, and letting $w_1, \ldots, w_{d-\mu}$ denote the zeroes of $q(z)$, each listed a number of times equal to its multiplicity, we see that equation (6) is equivalent to

$$\left|\sum_{j=1}^{d-\mu} \frac{1}{z - w_j}\right| < \left|\sum_{i=1}^{\mu} \frac{1}{z - z_i}\right|. \tag{7}$$

Consequently, we prove Claim 3.2.1, and hence the theorem, by showing that for $|z| = \rho$

$$\left|\sum_{j=1}^{d-\mu} \frac{1}{z - w_j}\right| < \frac{d-\mu}{R-\rho} < \frac{\mu}{\rho + r} < \left|\sum_{i=1}^{\mu} \frac{1}{z - z_i}\right|. \tag{8}$$

The leftmost inequality in expression (8) follows from the triangle inequality and the fact that for z satisfying $|z| = \rho$ we have $|z - w_j| > R - \rho$. The middle inequality is a simple consequence of equation (4). To complete the proof, we proceed as follows. For ρ fixed, let $z_* = \rho e^{\sqrt{-1}\theta*}$ denote the point that minimizes the rightmost side of expression (8). Then,

$$\left|\sum_{i=1}^{\mu} \frac{1}{z - z_i}\right| \geq \left|\sum_{i=1}^{\mu} \frac{1}{\rho e^{\sqrt{-1}\theta*} - z_i}\right| = \left|\sum_{i=1}^{\mu} \frac{1}{\rho - z_i e^{-\sqrt{-1}\theta*}}\right|.$$

Since $|z_i| \leq r$, we see that Lemma (3.1) applies, and the result is shown. $\square$

We denote the k-th derivative of h by $h^{(k)}(z)$.

Corollary 3.3 *Let $h(z)$ be a degree d polynomial of one complex variable. Assume that $h(z)$ has μ zeroes in the disk $\Delta_r(z_0) := \{z \in \mathbb{C} \,\Big|\, |z - z_0| \leq r\}$ and $d - \mu$ zeroes in the region $\{z \in \mathbb{C} \big| |z - z_0| \geq R\}$,* *where $R > r$. Assume further that $k \leq \mu - 1$. Then, if $\frac{R}{r} > \frac{2\binom{d}{\mu}}{d-\mu+1} - 1$, it follows that $h^{(k)}(z)$ has $\mu - k$ roots in $\Delta_r(z_0)$ and all the remaining $d - \mu$ roots in the region*

$$\left\{ z \in \mathbb{C} \;\middle|\; |z - z_0| > \frac{\binom{\mu}{k}}{\binom{d}{k}}(R + r) - r \right\}.$$

Proof. Use Theorem (3.2) k times. □

Suppose that $h(z)$ is an approximation to an underlying exact polynomial having a root of multiplicity μ. If we increase the precision used to evaluate $h(z)$ sufficiently, it will have a tight cluster of μ roots near the multiple root of the exact polynomial. Refining the precision until the radius r of the cluster is within the bound given by Corollary (3.3) assures us that the roots of $h^{(k)}(z)$ remain clustered. In particular, letting r denote the radius of a disk $\Delta_r(z_0)$ around z_0, the multiplicity weighted average of the μ roots, which contains the cluster of μ roots, and letting R denote the distance from z_0 to the first root outside of $\Delta_r(z_0)$, we have that the conservative estimate

$$\frac{R}{r} \geq \frac{2\binom{d}{\mu}}{d-\mu+1} \tag{9}$$

guarantees that $h^{(k)}(z)$, for all $k \leq \mu - 1$, has exactly $\mu - k$ zeroes in $\Delta_r(z_0)$.

Thus for a polynomial of degree 75, with roots of at most multiplicity 10, $\log_{10}(R/r) \geq 10.41$ is sufficient. For a given d, the worst case happens when μ is approximately $\frac{d}{2}$. Thus $\log_{10}(R/r) \geq 57.3$ is sufficient for a degree 200 polynomial having a worst case root of multiplicity 100. We may also regard $\log_{10}(R/r)$ as a measure of the number of decimal places needed to accurately carry out the computations. Using Stirling's approximation, we see that the number of decimal places needed in case $\mu \approx d/2$, and thus for all μ with the given d is approximately:

$$\log_{10}(R/r) \geq 0.3d. \tag{10}$$

4 Computational Experiments

In this section we report numerical results obtained with PHCpack [28]. This software package has recently been extended with facilities to handle positive dimensional solution components, see [24]. In particular, invoking the executable with the option `-f` gives access to the capability to factor multivariate polynomials.

4.1 A Numerical Implementation

Specialized versions of the path tracking routines in PHCpack have been built to deal with the case of homotopies between systems of one polynomial equation in several variables:

$$f(\mathbf{z}(t, \lambda)) = 0 \tag{11}$$

where

$$\mathbf{z}(t, \lambda) = (\mathbf{x}_0 + t\mathbf{v})\lambda + (\mathbf{y}_0 + t\mathbf{w})(1 - \lambda) \tag{12}$$

defines the movement from the line $\mathbf{x}(t) = \mathbf{x}_0 + t\mathbf{v}$ to the line $\mathbf{y}(t) = \mathbf{y}_0 + t\mathbf{w}$, as λ moves from 1 to 0, i.e.: $\mathbf{z}(t, \lambda) = \mathbf{x}(t)\lambda + \mathbf{y}(t)(1 - \lambda)$.

One motivation for this general rewrite of code is to save linear algebra operations. Without the parametric representation of a general line $\mathbf{x}(t) = \mathbf{x}_0 + t\mathbf{v}$, we have to consider polynomial systems of n equations consisting of one polynomial (the polynomial we wish to factor) and $n - 1$ hyperplanes to cut out the line. Now we can use, like in [17], the method of Weierstrass[1] (also known as Durand-Kerner) to solve $f(\mathbf{x}(t)) = 0$. See [29] for methods to locate zero clusters of univariate polynomials. The other motivation is that we hope to have a better understanding and control of the numerical stability of the algorithms.

As a general rule of thumb, our working precision should contain at least d decimal places when we want to reach a satisfactory accuracy for all roots of a univariate polynomial of degree d. To give an impression of the numerical hardness to compute witness points, just consider the binomial theorem, applied after substitution of $x(t) = x_0 + tv$ into the polynomial $f(x) = x^d$ (in *one* variable x):

$$f(x) = (x_0 + tv)^d = \sum_{k=0}^{d} \binom{d}{k} x_0^{d-k} v^k t^k = 0. \tag{13}$$

For example, the binomial coefficient for $d = 30$ and $k = 15$ occupies nine decimal places.

Extrapolating from this simple to the general case of computing all witness points, solving $f(\mathbf{x}(t)) = f(\mathbf{x}_0 + t\mathbf{v}) = 0$ for t, we warn that even if the original coefficients of f are nice, and if we choose the entries of $\mathbf{x}_0$ and $\mathbf{v}$ on the complex unit circle, for large degrees, the univariate polynomial in t may have coefficients that vary greatly in magnitude. To deal with such polynomials numerically, higher working precision may be required. This also implies that the path tracking in the monodromy phase of the algorithm may need higher precision. In previous work on polynomial systems having positive dimensional solution sets of higher multiplicity [24], we already extended the path-tracking routines in PHCpack to multi-precision. For such general polynomial systems, multi-precision path tracking tends to be computationally expensive, but we expect more reasonable execution times when addressing homotopies of only one polynomial equation.

Our working precision determines the accuracy of the algorithm **RegularFactor**. For instance, consider the polynomial $f(x, y) = xy + 10^{-16}$. Working with standard double precision floating point numbers, the constant in $f(x, y)$ will be ignored and a loop which shows f is irreducible will not be found, and also the validation with linear traces will confirm the breakup into the factors x and y. On the other hand, doubling the precision will show that f is irreducible.

In principle, the groupings of the monodromy algorithm can deal with approximate coefficients if we set the working precision according to the accuracy level of the coefficients. However, this scheme only works for sufficiently low degrees.

To obtain symbolic expressions of the polynomial factors, we applied the interpolation methods using traces, developed and implemented for any degree and any number of variables using multi-precision arithmetic if needed, reported in [22]. Note however that usually we do not need the symbolic representation of the factors to work with them. For instance, with the witness points we can determine whether a point satisfies a factor, via the homotopy membership test of [20].

[1] In [12] this method is qualified as "quite effective and increasingly popular." Convergence is global and quadratic in the limit [14].

4.2 Singularities of Stewart-Gough Platforms

A mechanical device of considerable interest in mechanical engineering is the Stewart-Gough Platform manipulator, consisting of a moving platform supported above a stationary base by six legs. One end of the i-th leg connects to the base via a ball joint centered on point $\mathbf{b}_i$ (given in the base coordinate system) and the other end connects to the platform via a ball joint at point $\mathbf{a}_i$ (given in platform coordinates). The length of the leg, L_i, is controlled by a linear actuator. For fixed leg lengths, the device is generally rigid, and by extending or contracting the legs, one can produce arbitrary six-degree-of-freedom motions (translations and rotations) of the platform within the working volume of the device. This device has been applied to motion platforms for flight simulators, to industrial robots, and to six-axis machining centers. A good general reference discussing this device and its relatives is Merlet [10].

At singular configurations, the rigidity of the device is lost, so that even though the leg lengths are held constant, the platform has at least one twist motion (a combination of velocity and angular velocity) that, up to first order, is unconstrained. Indeed, it can happen that there is a continuous curve or higher-dimensional set of singular configurations, thus allowing a finite motion. Singular configurations are disastrous for manipulator applications, because the extra motion cannot be controlled via the leg lengths. In other cases, one may wish to build a moveable Stewart-Gough mechanism to guide the motion of a link in a machine. Either way, the study of singularities is fundamental and has been undertaken by many researchers. We cannot begin to give a comprehensive view here, but refer to the following sources as a starting point: [9, 11].

The condition for singularity is that the determinant of the 6×6 Jacobian matrix for the mapping from platform motion to leg lengths is zero. Thus, it defines a five-dimensional set in the six-dimensional space of translation and orientation. The Jacobian matrix can be derived as follows. We represent the position of the platform by $\mathbf{p} \in \mathbb{C}^3$ and the orientation by a quaternion $\mathbf{q} \in \mathbb{P}^3$. Given $\mathbf{q}$, the rotation matrix representing the orientation of the platform with respect to the base may be written as

$$R = \hat{R}/Q, \tag{14}$$

where

$$\begin{aligned}
\hat{R} &= \begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2+q_0q_3) & 2(q_1q_3-q_0q_2) \\ 2(q_1q_2-q_0q_3) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3+q_0q_1) \\ 2(q_1q_3+q_0q_2) & 2(q_2q_3-q_0q_1) & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix} && (15) \\
Q &= q_0^2+q_1^2+q_2^2+q_3^2. && (16)
\end{aligned}$$

Let $\mathbf{v}_i$ be the vector, in base coordinates, from base point $\mathbf{b}_i$ to the corresponding platform point $\mathbf{a}_i$:

$$\mathbf{v}_i = -\mathbf{b}_i + \mathbf{p} + R\mathbf{a}_i. \tag{17}$$

The squared length of leg i is $L_i^2 = \mathbf{v}_i \cdot \mathbf{v}_i$, so

$$L_i\dot{L}_i = \mathbf{v}_i \cdot \dot{\mathbf{v}}_i = \mathbf{v}_i \cdot (\dot{\mathbf{p}} + \boldsymbol{\omega} \times (R\mathbf{a}_i)). \tag{18}$$

where "·" is the vector inner product, $\times$ is the vector cross product, and $\boldsymbol{\omega}$ is the angular velocity vector. Using equation (14) and the identities $\mathbf{v} \cdot \boldsymbol{\omega} \times \mathbf{x} = (\mathbf{x} \times \mathbf{v}) \cdot \boldsymbol{\omega}$, $\mathbf{x} \times \mathbf{x} = 0$, we have

$$QL_i\dot{L}_i = (Q(\mathbf{p} - \mathbf{b}_i) + \hat{R}\mathbf{a}_i) \cdot \dot{\mathbf{p}} + ((\hat{R}\mathbf{a}_i) \times (\mathbf{p} - \mathbf{b}_i)) \cdot \boldsymbol{\omega}. \tag{19}$$

Accordingly, the i-th column, $\mathbf{J}_i$, of the Jacobian matrix, $\mathbf{J}$, is

$$\mathbf{J}_i = \begin{bmatrix} Q(\mathbf{p}-\mathbf{b}_i)+\hat{R}\mathbf{a}_i \\ (\hat{R}\mathbf{a}_i)\times(\mathbf{p}-\mathbf{b}_i) \end{bmatrix}, \qquad i=1,\ldots,6, \tag{20}$$

and the singularity condition is simply

$$\det \mathbf{J} = 0. \tag{21}$$

Using equations (15),(16) to substitute into equation (20), $\det \mathbf{J}$ becomes a polynomial in $\mathbf{p},\mathbf{q},\mathbf{a}_i,\mathbf{b}_i$. Taking all of these as variables, the first three rows of $\mathbf{J}$ are cubic and the last three are quadric, so $\det \mathbf{J}$ is a homogeneous polynomial of degree 1728 in 42 variables. Not much understanding is likely to result from analyzing such a complicated object, nor could we begin to deal with it numerically. However, considerable insight can be gained by studying cases where some variables are taken as given, as has been done in [9, 15]. In the next few paragraphs, we study such cases, some never before published, and use our numerical algorithm to factor $\det \mathbf{J}$.

In all of the following examples, we expanded $\det \mathbf{J}$ into monomials for convenient input to our computer code. This made automatic computation of derivatives very simple, but it is a very inefficient way to evaluate the polynomial. It would be much more efficient and accurate to evaluate the matrix entries numerically and then evaluate the determinant by reducing the matrix to triangular form. Therefore, the computation times reported here are far from the best that could be achieved. The examples serve to show how the algorithm works on fully expanded polynomials.

4.2.1 General platform, fixed position

For the general platform, we give $\mathbf{p}$, $\mathbf{a}_i$ and $\mathbf{b}_i$, $i=1,\ldots,6$, as random, complex values; that is, we choose a generic Stewart-Gough platform at a generic position, and look for singularities arising from rotations of the platform. The factorization of one such example will indicate, with probability one, the form of the factorization for *almost all*[2] Stewart-Gough platforms. In this case, $\det \mathbf{J}$ is a homogeneous polynomial of degree 12 in $\mathbf{q}=\{q_0,q_1,q_2,q_3\}$. We find numerically that a generic line hits $\det \mathbf{J}$ in six regular points and two singular points of multiplicity 3. The regular points form one factor of degree six. Using differentiation to remove the multiplicity, we find that the two singular points form one quadratic factor, and interpolating that factor shows it to be precisely Q from equation (16). That is,

$$\det \mathbf{J} = F_1(\mathbf{q})Q^3, \tag{22}$$

where $F_1(\mathbf{q})$ is a sextic. Since a quaternion with zero norm does not represent a valid rotation matrix (see Eq.14), the factor of $Q^3=0$ is not of physical significance.

The computed factorization is certified with linear traces by comparing the value at the linear trace with the calculated sum of the witness points on each factor. The maximal difference in the comparison for the two factors is `2.049E-13`. If we multiply the interpolated factors and take the difference with the original polynomial, then the largest norm of the coefficients of the difference polynomial is `1.919E-05`, as explained by roundoff in the interpolation of high degree polynomials.

The data for the cluster analysis, comparing cluster radius r and distance R to the nearest other root outside the cluster, is given in Table 1. For $d=12$ and $\mu=3$, the right-hand side of the estimate (9) evaluates to 44. This bound is clearly smaller than 10^4, so the initial approximations for the multiple root are accurate enough for the differentiation process.

[2] The exceptions will be an algebraic subset of the space of all platform devices as parameterized by $\mathbf{p}$, $\mathbf{a}_i$, $\mathbf{b}_i$, $i=1,\ldots,6$.

cluster	r	R	R/r
one	1.7E-05	3.4E-01	2.0E+04
two	4.9E-06	1.7E-01	3.6E+04

Table 1: Cluster radius r versus distance R to the nearest root outside the cluster, for the first case of general platform, fixed position. There are three roots in every cluster.

Table 2 lists the execution times for each stage in the factorization: monodromy grouping, certification with linear traces, interpolation in the factors, and finally, the comparison between the product of the factors with the original polynomials. The evaluation of a polynomial of degree 12 in four variables with 910 terms is responsible for the dominance of stages one and three in the overall execution time.

Elapsed user CPU times on 2.4Ghz WindowsXP			
1.	monodromy grouping	:	0h 6m 40s 469ms
2.	linear traces certification	:	0h 0m 30s 672ms
3.	interpolation at factors	:	1h 41m 53s 78ms
4.	multiplication validation	:	0h 0m 8s 156ms
	total time for all 4 stages	:	1h 49m 12s 391ms

Table 2: Execution times for the first case of general platform, fixed position.

4.2.2 Planar base and platform, fixed position

This is the same as the former case, except that the third component of each of $\mathbf{a}_i$, $\mathbf{b}_i$, $i = 1 \ldots, 6$ is zero, meaning that the points of the base are all in a common plane, as are the points of the platform. Now $\det \mathbf{J}$ is still homogeneous of degree 12, and it still factors in two pieces: one irreducible single factor of degree six, and the quadratic factor having multiplicity three.

The maximal difference in certifying with linear traces is `4.147E-11`, i.e.: the difference between the calculated sum in the roots and the value predicted by the linear trace of the factor is `4.147E-11`, showing the influence of roundoff in evaluation a polynomial of degree 12 in four variables with 910 terms. The influence of roundoff in the comparison between the original and product of the interpolated factors is even more obvious: we see `9.483E-05` as the maximal norm of the difference in the coefficients.

In Table 3 we summarize the results of the cluster analysis for two clusters which contain witness points of multiplicity three. As in Table 1 we can make the same observations, to conclude that $R/r \approx 10^4 > 44$ is safe for the differentiations. Table 4 shows the execution times. The algorithm **RegularFactor** takes so much time because of the cost of evaluating a polynomial of degree 12 in four variables with 910 terms.

cluster	r	R	R/r
one	`6.2E-05`	`2.4E-01`	`3.8E+04`
two	`4.8E-05`	`6.0E-01`	`1.2E+04`

Table 3: Cluster radius r versus distance R to the nearest root outside the cluster, for the second case of planar base and platform, fixed position. There are three roots in every cluster.

Elapsed user CPU times on 2.4Ghz WindowsXP			
1.	monodromy grouping	:	`0h 17m 34s 735ms`
2.	linear traces certification	:	`0h  0m 27s 359ms`
3.	interpolation at factors	:	`1h 24m 45s 766ms`
4.	multiplication validation	:	`0h  0m  8s 172ms`
	total time for all 4 stages	:	`1h 42m 56s  32ms`

Table 4: Execution times for the second case of planar base and platform, fixed position.

4.2.3 Planar base and platform, parallel planes

In this case, which was studied in [9, 15], we consider a device with planar base and platform in a configuration with the two planes parallel to each other. The condition of parallelism means that the platform is rotated only about the third axis, so $q_1 = q_2 = 0$. The position, $\mathbf{p}$, is now left as variable, and $\det \mathbf{J}$ becomes cubic in $\mathbf{p}$, homogeneous of degree 12 in (q_0, q_3), and degree 15 in $(\mathbf{p}, \mathbf{q})$ together. One does not know *a priori* that the contribution of $\mathbf{p}$ will factor out separately, but in fact it does. The computed factorization is

$$\det \mathbf{J} = a p_3^3 (q_0 + b q_3)(q_0 + c q_3)(q_0 + i q_3)^5 (q_0 - i q_3)^5, \tag{23}$$

where a, b, c are constants that depend on the choice of $\mathbf{a}_i, \mathbf{b}_i$. This result is in agreement with [15], when we consider that, over the complex numbers, any homogeneous polynomial in two variables breaks into linear factors. Notice that the multiplicity five factors are points on Q from equation (16) and therefore are not of physical significance. The condition $p_3 = 0$ means that the two planes coincide, which is clearly singular since then the legs provide no support perpendicular to the plane. Otherwise, singularity does not depend on position at all, as the other factors depend only on orientation. This fact is used to advantage in [15] to characterize the singularities of the planar-planar Stewart-Gough platforms.

The numerical results give a maximal difference over all factors between the computed sum of roots and the value at the linear trace as `8.047E-08`. When we interpolate the factors and compare the multiplied factors with the original polynomial, we find `3.599E-07` as the highest norm of the difference between the coefficients.

In Table 4.2.3 we summarize the results of the cluster analysis for the three factors occurring with multiplicities three, five, and five. Observe that the cluster radius grows as the multiplicity gets larger. The bound in the estimate of the right-hand side of (9) now evaluates to 546 using $d = 15$ and $\mu = 5$. While the approximation for R/r lies now much closer to this bound, numerically we can still apply the differentiation procedure successfully.

Table 6 shows the execution times. Since there are three witness points of multiplicity three, no monodromy is needed to isolate this factor. While the other polynomials each have 910 terms, this polynomial is much sparser: only 24 terms. The sparsity reduces the cost of evaluation and explains why this polynomial can be factored much faster than the other two cases.

cluster	r	R	R/r
one	5.1E-07	1.0E+00	2.0E+06
two	7.3E-04	3.4E-01	4.7E+02
three	4.0E-03	7.2E-01	1.8E+02

Table 5: Cluster radius r versus distance R to nearest root outside the cluster, for the third case of planar base and platform, parallel planes. There are three roots in the first cluster, and five roots in the other two clusters.

Elapsed user CPU times on 2.4Ghz WindowsXP			
1.	monodromy grouping	:	1m 13s 656ms
2.	linear traces certification	:	0m 3s 891ms
3.	interpolation at factors	:	0m 4s 734ms
4.	multiplication validation	:	0m 1s 657ms
	total time for all 4 stages	:	1m 23s 938ms

Table 6: Execution times for the third case of planar base and platform, parallel planes.

4.3 Monodromy Compared to the Enumeration Method

In this section we show that for the polynomials of modest degrees that we factored in this paper, the enumeration method outperforms the monodromy algorithm. In Table 7 we list execution times for the three cases treated above. We see the enumeration method as a clear winner. Recall that the highest degree of an irreducible factor is six, so only relatively few tests are needed in the enumeration method.

User CPU times on 2.4Ghz Windows XP		
case	monodromy	enumeration
1	6m 40s 460ms	40s 750ms
2	17m 34s 735ms	31s 657ms
3	1m 13s 656ms	3s 0ms

Table 7: Execution times for the factorization using monodromy, compared to enumerating factors, for the three cases of the singularities of the Stewart-Gough platform.

Irreducible polynomials are the most difficult for the enumeration method. In Table 8 we compare execution times again, but now for random irreducible polynomials of five monomials, and for increasing degrees. We see that the ratio of monodromy time to enumeration time drops from 18 to about 6 as the degree increases.

User CPU times on 2.4Ghz Windows XP		
degree	monodromy	enumeration
10	5s 484ms	312ms
15	8s 187ms	1s 453ms
16	16s 63ms	2s 875ms

Table 8: Execution times for the factorization using monodromy, compared to enumerating the irreducible factors, for three very sparse random polynomials of increasing degrees.

5 Conclusions

In this paper we show how the general monodromy breakup algorithm with linear trace certification can be specialized to the numerical factorization of polynomials in several variables with approximate complex coefficients. To deal with factors of multiplicity $\mu > 1$, we symbolically differentiate the polynomial $\mu - 1$ times, thus replacing singular roots with nonsingular ones. In floating point calculations, a multiple root becomes a cluster of nearly singular roots. Via a result of Marden and Walsh, we can estimate the precision needed to successfully apply differentiation to replace such clusters with one nonsingular root. To illustrate the methods, we applied the implemented algorithms in a study of singularities of Stewart-Gough platforms.

Since the polynomials we can factor with standard arithmetic have modest degrees, the enumeration methods of Galligo and Rupprecht ([4, 5], [16]) proved to be faster than the monodromy method in our tests. If one were to factor polynomials of higher degree, the speed advantage might reverse, due to the exponential growth of the enumeration method.

References

[1] R.M. Corless, M.W. Giesbrecht, M. van Hoeij, I.S. Kotsireas and S.M. Watt. Towards factoring bivariate approximate polynomials. In *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation (ISSAC 2001)*, edited by B. Mourrain, pages 85–92, ACM 2001.

[2] R.M. Corless, A. Galligo, I.S. Kotsireas, and S.M. Watt. A geometric-numeric algorithm for factoring multivariate polynomials. In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation (ISSAC 2002)*, edited by T. Mora, ACM 2002.

[3] R.M. Corless, E. Kaltofen, and S.M. Watt. Hybrid methods. In *Computer Algebra Handbook*, edited by J. Grabmeier, E. Kaltofen, and V. Weispfenning, pages 112-125, Springer-Verlag, 2002.

[4] A. Galligo and D. Rupprecht. Semi-numerical determination of irreducible branches of a reduced space curve. In *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation (ISSAC 2001)*, edited by B. Mourrain, pages 137–142, ACM 2001.

[5] A. Galligo and D. Rupprecht. Irreducible decomposition of curves. *J. Symbolic Computation* 33(5):661–677, 2002.

[6] Y. Huang, W. Wu, H.J. Stetter, and L. Zhi. Pseudofactors of multivariate polynomials. In *Proceedings of the 2000 International Symposium on Symbolic and Algebraic Computation (ISSAC 2000)*, edited by C. Traverso, pages 161-168, 2000.

[7] E. Kaltofen. Challenges of symbolic computation: my favorite open problems, *J. Symbolic Computation* 29(6): 891–919, 2000.

[8] M. Marden. *The Geometry of the Zeroes of A Polynomial in a Complex Variable.* Volume 3 of *Mathematical Surveys*, American Mathematical Society, New York, 1949.

[9] B. Mayer St-Onge and C.M. Gosselin. Singularity analysis and representation of the general Gough-Stewart platform. *Int. J. Robotics Research*, 19(3):271–288, 2000.

[10] J.P. Merlet. *Parallel Robots*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.

[11] J.P. Merlet. Singular configurations of parallel manipulators and Grassmann geometry. *Int. J. Robotics Research*, 8(5):45–56, 1989.

[12] V. Pan. Solving a polynomial equation: some history and recent progress. *SIAM Review* 39(2):187–220, 1997.

[13] V. Pan. Computation of approximate polynomial GCDs and an extension. *Information and Computation* 167:71–85, 2001.

[14] L. Pasquini and D. Trigiante. A globally convergent method for simultaneously finding polynomial roots. *Math. Comp.* 44(169):135–149, 1985.

[15] F. Pernkopf and M.L. Husty. Singularity analysis of spatial Stewart-Gough platforms with planar base and platform. *Proc. ASME Design Eng. Tech. Conf.*, Montreal, Canada, Sept. 30–Oct. 2, 2002.

[16] D. Rupprecht. Semi-numerical absolute factorization of polynomials with integer coefficients. *J. Symbolic Computation*, to appear.

[17] T. Sasaki. Approximate multivariate polynomial factorization based on zero-sum relations. In *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation (ISSAC 2001)*, edited by B. Mourrain, pages 284–291, ACM 2001.

[18] A.J. Sommese and J. Verschelde. Numerical homotopies to compute generic points on positive dimensional algebraic sets. *J. Complexity* 16(3):572–602, 2000.

[19] 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.

[20] A.J. Sommese, J. Verschelde, and C.W. Wampler. Numerical irreducible decomposition using projections from points on the components. In *J. Symbolic Computation: Solving Equations in Algebra, Geometry, and Engineering*, volume 286 of *Contemporary Mathematics*, edited by E.L. Green, S. Hoşten, R.C. Laubenbacher, and V. Powers, pages 37–51. AMS 2001.

[21] A.J. Sommese, J. Verschelde, and C.W. Wampler. Using monodromy to decompose solution sets of polynomial systems into irreducible components. In *Application of Algebraic Geometry to Coding Theory, Physics and Computation*, edited by C. Ciliberto, F. Hirzebruch, R. Miranda, and M. Teicher. Proceedings of a NATO Conference, February 25 - March 1, 2001, Eilat, Israel. Pages 297–315, Kluwer Academic Publishers.

[22] 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.

[23] A.J. Sommese, J. Verschelde, and C.W. Wampler. A method for tracking singular paths with application to the numerical irreducible decomposition. In *Algebraic Geometry, a Volume in Memory of Paolo Francia*, edited by M.C. Beltrametti, F. Catanese, C. Ciliberto, A. Lanteri, C. Pedrini. W. de Gruyter, pages 329-345, W. de Gruyter, 2002.

[24] A.J. Sommese, J. Verschelde, and C.W. Wampler. Numerical irreducible decomposition using PHCpack. In *Algebra, Geometry, and Software Systems*, edited by M. Joswig and N. Takayama, pages 109–130, Springer-Verlag, 2003.

[25] A.J. Sommese, J. Verschelde, and C.W. Wampler. Advances in polynomial continuation for solving problems in kinematics. In *Proc. ASME Design Engineering Technical Conf.* (CDROM), Paper DETC2002/MECH-34254. Montreal, Quebec, Sept. 29-Oct. 2, 2002.

[26] A.J. Sommese, J. Verschelde, and C.W. Wampler. Introduction to Numerical Algebraic Geometry. Outline of notes for the CIMPA Graduate School, to be published by INRIA, 2003.

[27] A.J. Sommese and C.W. Wampler. Numerical algebraic geometry. In J. Renegar, M. Shub, and S. Smale, editors: *The Mathematics of Numerical Analysis*, volume 32 of *Lectures in Applied Mathematics*, 749–763, 1996. Proceedings of the AMS-SIAM Summer Seminar in Applied Mathematics, Park City, Utah, July 17-August 11, 1995, Park City, Utah.

[28] J. Verschelde. Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Transactions on Mathematical Software* 25(2): 251–276, 1999. Software available at `http://www.math.uic.edu/~jan`.

[29] J.-C. Yakoubsohn. Finding a cluster of zeros of univariate polynomials. *J. Complexity* 16(3):603–638, 2000.