Using Monodromy to Decompose Solution Sets of Polynomial Systems into Irreducible Components

Andrew J. Sommese, Jan Verschelde, and Charles W. Wampler

March 25, 2001

Abstract

To decompose solution sets of polynomial systems into irreducible components, homotopy continuation methods generate the action of a natural monodromy group which partially classifies generic points onto their respective irreducible components. As illustrated by the performance on several test examples, this new method achieves a great increase in speed and accuracy, as well as improved numerical conditioning of the multivariate interpolation problem.
2000 Mathematics Subject Classification. *Primary* 65H10; *Secondary* 13P05, 14Q99, 68W30.
Keywords. components of solutions, embedding, generic points, homotopy continuation, irreducible components, monodromy group, numerical algebraic geometry, polynomial system, primary decomposition.

Contents

1 Introduction

To describe positive dimensional solution sets of polynomial systems, we work with *generic points* [22] of irreducible components. These points are produced by slicing the solution set with general linear subspaces of the ambient Euclidean space. In particular, slicing the union Z_i of the i-dimensional components of the solution set in $\mathbb{C}^n$ with a general linear space of dimension $n - i$ will result in a set W_i of smooth points of the components. The cardinality of this set equals the degree of Z_i. A decomposition of a positive dimensional solution set into irreducible components is realized by a partition of the whole set W_i of generic points into subsets of points that lie on the same irreducible component.

In [20] and [21], this decomposition was achieved by incrementally building up interpolating polynomials. In this paper we propose to apply the actions of a natural monodromy group to find a find a partition of all generic points that is compatible with the partition into irreducible components, i.e., a subset generated by this new decomposition consists of points on an irreducible component, but not necessarily all of them. This reduces, in many cases significantly, the work needed to compute the filtering polynomials. Given a grouping G of generic points from the breakup of W_i induced by the monodromy action, we construct a polynomial p, that would be a filtering polynomial, if the grouping did coincide with the points of W_i lying on an irreducible component of the solution set. Evaluation of p on additional test points obtained by homotopy paths starting in G checks whether the grouping G is the grouping associated to an irreducible component. If so, then p is a filtering polynomial for the irreducible component. If not, then p can be used as a starting point for constructing a filtering polynomial of the irreducible component containing G.

An important added advantage is that a set G of generic points on which the monodromy action is transitive, is well suited for use of generalized divided differences [15] to write down a well-conditioned multivariate polynomial p that interpolates the points. This will be covered in a sequel.

The method applies to all components, but because we only have an efficient implementation of path tracking for paths on components having multiplicity one, in this paper, we apply the new technique only to such components.

This paper is organized in four parts. In section 2 we present the fundamentals on monodromy actions, and we outline the algorithm to decompose solution sets with the monodromy group. In section 3 we discuss the extra processing needed if the breakup achieved, using monodromy, does not equal the breakup corresponding to the decomposition of Z_i into its irreducible components. Section 4 addresses the condition of the decomposition problem on the special problem of factoring multivariate polynomials. In section 5, we apply our new approach to systems from the literature.

2 Monodromy Group Actions

2.1 Fundamentals on Monodromy Actions

Let Z denote the reduction of the algebraic set defined by a system of polynomials f on $\mathbb{C}^N$, that is, Z is the underlying set of points of the solution set of $f = \mathbf{0}$. Z is an algebraic set. Let us consider Z_i, the union of the i-dimensional irreducible components of Z. Let S_i denote the algebraic subset of Z_i consisting of all points of Z_i that lie on at least two distinct components of Z. We consider the space U of all affine linear subspaces $\mathbb{C}^{N-i} \subset \mathbb{C}^N$. Most $L \in U$ meet Z_i in a set of $\deg Z_i$ distinct points contained in $Z_i \setminus S_i$. The distinctness of the points implies, in particular, that the points are smooth points of the set Z_i. Let $\mathcal{D}$ denote the set of points of U for which this is not true. Choose an $L \in U \setminus \mathcal{D}$. Taking a piecewise smooth map of the unit circle $\gamma : S^1 \to U \setminus \mathcal{D}$ with $\gamma(0) = \gamma(1) = L$, we can trace the homotopy paths of the points $L \cap Z_i$ as we traverse around the unit circle. The mapping from the starting points at $\gamma(0)$ to the corresponding endpoints at $\gamma(1)$ is a bijection of the set $L \cap Z_i$. Since the elements of $L \cap Z_i$ stay on the same irreducible component under the continuation along γ, it follows that if a point $\mathbf{a}$ of $L \cap Z_i$ is taken to a point $\mathbf{b}$ of $L \cap Z_i$, then $\mathbf{a}$, $\mathbf{b}$ are on the same irreducible component. The key observation of this paper is that it is easy — and numerically stable — to generate many loops γ, and the associated bijections can be used to decompose $L \cap Z_i$ into "monodromy groupings." These groupings are compatible with the grouping according to irreducible components, which is the objective of our previous papers [22, 20, 21].

It is an elementary fact that the converse is also true; that is, two points of $L \cap Z_i$ are on the same irreducible component of Z_i only if they are connected by a monodromy action. Hence, a complete irreducible decomposition is determined by a sufficient set of monodromy loops. We do not know of any efficient algorithm that is guaranteed to find such a set of loops, but we can test whether a given monodromy grouping is sufficient in this sense. If not, one can generate another round of loops, or one can proceed to an approach like that in [20, 21], which may be expensive, but which is guaranteed to terminate.

2.2 An Illustrative Example

For example, let us take as system one equation $f(x, y) = xy - 1$ and slice it with $L = x + y - t = 0$, where t is some parameter. Then, for all $t \neq \pm 2$, the system

$$\left\{ \begin{array}{c} xy - 1 = 0 \\ x + y - t = 0 \end{array} \right. \tag{1}$$

has two solutions $(x_1(t), y_1(t))$ and $(x_2(t), y_2(t))$:

$$\left(x_1(t) = \frac{t}{2} + \frac{1}{2}\sqrt{t^2 - 4}, y_1(t) = \frac{t}{2} - \frac{1}{2}\sqrt{t^2 - 4} \right) \tag{2}$$

and

$$\left(x_2(t) = \frac{t}{2} - \frac{1}{2}\sqrt{t^2 - 4}, y_2(t) = \frac{t}{2} + \frac{1}{2}\sqrt{t^2 - 4} \right). \tag{3}$$

For $t = \pm 2$ the two solutions coincide.

To illustrate a monodromy action, we choose a parameterized closed loop $t = 2(1 + e^{i\theta})$, which starts and ends at $t = 4$ as θ takes real values going from 0 to 2π. This loop does not contain either of the double roots, and so it qualifies as a monodromy loop. Substitution of the parameterization for t into Eq.(2) gives, after some simplification,

$$x_1(t) = (1 + e^{i\theta}) + e^{i\theta/2}\sqrt{2 + e^{i\theta}}. \tag{4}$$

For continuity, the square root in this expression is taken to lie always in the right half of the complex plane, while $e^{i\theta/2}$ goes from 1 to -1 as θ goes from 0 to 2π. The other expressions in Eqs.(2,3) follow a similar pattern, and as a consequence we see that after one trip around the loop, the first solution $(x_1(\theta), y_1(\theta))$ has moved to the second solution $(x_2(\theta), y_2(\theta))$, and vice versa. This shows that the two solutions lie on the same connected component.

2.3 The Decomposition Algorithm

Suppose that f is a polynomial system that has an i-dimensional solution set $Z_i = \cup_{j\in I_i} Z_{ij} \subset \mathbb{C}^N$, where the Z_{ij} are the irreducible components of dimension i. We follow the convention of letting L stand for both a system of i linear equations of rank i on $\mathbb{C}^N$ and for the $(N-i)$-dimensional linear subspace of solutions of L on $\mathbb{C}^N$. Let us pick two such subspaces, L_0 and L_1, at random, and suppose that we have the set of solution points $W := Z_i \cap L_0$. (These are the "witness points" for dimension i as described in [20].) With probability one, $\deg(Z_i) = \#(Z_i \cap L_0) = \#(Z_i \cap L_1)$. For general subspaces L_j and L_k, we define the homotopy

$$H_{jk\lambda}(\mathbf{x}(t), t) = \lambda(1-t)\begin{pmatrix} f \\ L_k \end{pmatrix} + t\begin{pmatrix} f \\ L_j \end{pmatrix} = \mathbf{0}, \quad \lambda \in \mathbb{C}, t \in [0,1]. \tag{5}$$

For generic λ, the solution set to $H_{jk\lambda}(\mathbf{x}(t), t) = \mathbf{0}$ on Z_i consists of exactly $\deg(Z_i)$ paths $\mathbf{x}(t)$ starting at points of $Z_i \cap L_j$ and ending at points of $Z_i \cap L_k$ as t goes from 1 to 0. This gives a bijection from $Z_i \cap L_j$ to $Z_i \cap L_k$. A bijection mapping of W onto itself can be constructed by concatenating two or more such homotopy mappings, such as $H_{01\lambda_1}(\mathbf{x}(t), t) = \mathbf{0}$ and $H_{10\lambda_2}(\mathbf{x}(t), t) = \mathbf{0}$. (If $\lambda_1 \neq \lambda_2$, the forward and return paths are not identical, and the bijection is not necessarily the identity mapping.) In general, we can concatenate any number of such homotopies, returning at the end to the same $Z_i \cap L_j$ from which we start, to generate additional bijections.

There are many ways one could choose to set up bijections using homotopies. We choose to proceed as follows. We start with $W_i := Z_i \cap L_0$ given. For simplicity, we refer to W_i as W. At stage k, we pick a new L_k and λ_{0k} at random and compute the homotopy paths for $H_{0k\lambda_{0k}}(\mathbf{x}(t), t) = \mathbf{0}$, getting the ordered list of solutions $X_k := Z_i \cap L_k$ and the mapping ${}_kh_0 : W \to X_k$. (Here and below, the left and right subscripts of ${}_kh_0$ indicate, respectively, the output and input spaces.) Then, choosing new random constants λ_{kj}, we compute k homotopy paths $H_{kj\lambda_{kj}}(\mathbf{x}(t), t) = \mathbf{0}$, for $j = 0, \ldots, k-1$, thereby obtaining mappings ${}_jh_k : X_k \to X_j$. Altogether, we have k new bijections being maps from W to itself, namely, $T_{0,k,0} := {}_0h_k \circ {}_kh_0$ and $T_{0,k,j,0} := {}_0h_j \circ {}_jh_k \circ {}_kh_0$, for $j = 1, \ldots, k-1$. The L_k for $k \neq 0$ do not have to be distinct, but we find that the algorithm is more robust if we choose each one independently.

We accumulate the monodromy groupings from the individual bijections as follows. Begin with a partition in which each point in W assigned to its own subset. For a bijection generated as above, check if any point is mapped to a point from a different subset. If so, a new partition is formed by joining the two subsets into one. The partition is updated by each subsequent bijection in a similar manner.

A basic version of an implementation of this idea is given below in Algorithm **MonodromyGrouping**. The subroutine **HomotopyMap** uses the homotopy (5) to determine mappings ${}_kh_j$ as described above and procedure **Partition** forms a new partition from a previous one according to any connections between subsets implied by the associated bijections. It may be that a newly computed bijection does not find any new connections so that the partitioning is unchanged. We call this a "stable" iteration. If the partitioning achieves a complete irreducible decomposition, then all further iterations must be stable, but a stable iteration does not imply that the irreducible decomposition is in hand: it may just be that the random constants chosen in the homotopies are not fortuitous. Therefore, a termination condition S, an integer, is specified, such that **MonodromyGrouping** terminates when S consecutive stable iterations are computed. The larger S is, the longer the algorithm will persist in the face of fruitless iterations. The algorithm must terminate in at most $(\#(W)-1)S$ iterations, since at least one connection must be found every S iterations and after $(\#(W)-1)$ connections are made, the algorithm terminates with all points in a single group. In practice, the algorithm generally terminates much sooner.

Algorithm 2.1 $[P] =$ **MonodromyGrouping**(f, L_0, W, S)

Input: Polynomial system f on $\mathbb{C}^N$;
Affine linear space $L_0 \subset \mathbb{C}^N$ of dimension $N-i$;
Generic points W of f on L_0; Termination condition S.
Output: P is partition of W.

$s := 0;$	*[counter for stable iterations]*
$P := \{\ \{\mathbf{w}\} \mid \mathbf{w} \in W\ \};$	*[initial partition is finest one]*
$k := 0;$	*[counter for forward paths]*
loop	
$\quad k := k+1;$	*[increment counter]*
$\quad L_k :=$ **RandomLinear**$(N-i);$	*[random $(N-i)$-space]*
$\quad {}_kh_0 :=$ **HomotopyMap**$(f, W, L_0, L_k);$	*[paths from L_0 to L_k]*
$\quad j := 0;$	*[counter for return paths]*
$\quad$ while $j < k$ do	
$\qquad {}_jh_k :=$ **HomotopyMap**$(f, W, L_k, L_j);$	*[return paths]*
$\qquad$ if $j = 0$	*[form bijection T from maps]*
$\qquad\quad$ then $T := {}_0h_k \circ {}_kh_0;$	*[path 0 to k and back]*
$\qquad\quad$ else $T := {}_0h_j \circ {}_jh_k \circ {}_kh_0;$	*[path 0 to k and back via j]*
$\qquad$ end if;	
$\quad P' :=$ **Partition**$(P, T);$	*[merge subsets connected by paths]*
$\quad$ if $\#P' = \#P$	*[compare with previous partition]*

```
      then s := s + 1;                        [another stable iteration]
      else  s := 0;                           [reset stable iteration counter]
    end if;
    P := P';                                  [update partition]
    exit when ((s = S) or (#P = 1));          [termination condition]
    j := j + 1;                               [increment counter]
  end while;
end loop.
```

An additional output of **MonodromyGrouping** could be the list of samples on the new random slices L_k used in the homotopy (5). This list can be used intermediately to determine the linear span of the solution components. In case all components are linear, no further computation of filtering polynomials is necessary.

To classify the generic points on solution sets of several different dimensions, we apply **MonodromyGrouping** at each dimension. Before starting at each dimension, we filter out points on higher dimensional components using the homotopy membership test proposed in [21].

3 Further Processing and Validation

A basic operation in our approach to computing an irreducible decomposition is the determination of a filtering polynomial that vanishes on an irreducible component and whose degree is equal to the degree of the component. In [20, 21], this was accomplished by sampling a component via homotopy paths extending from a single solution point and successively testing for higher and higher degrees for the filtering polynomial. The monodromy grouping described in the previous section gives us two advantages in finding a filtering polynomial. First, the number of points in a monodromy group is a lower bound on the degree of the irreducible component. Second, we may sample the component via homotopy paths extending from all of the points in the group. We will show in a sequel that by organizing these sample points into a grid, we can use divided differences to more efficiently compute the interpolating polynomial.

Assume that **MonodromyGrouping** gives a partition P of W into disjoint subsets $G_1, \ldots, G_m$. We can construct a polynomial p_j for each G_j, which would be a filtering polynomial, if $G_j \subset W$ is the set of $\deg Z_{ij}$ generic points of an irreducible component Z_{ij} of Z_i. By further sampling, we can check which, if any, of the p_j are not filtering polynomials. Since further processing is needed only for these groupings, we can assume by renaming that we have

1. a set W of generic points of some set of i-dimensional irreducible components of $f^{-1}(\mathbf{0})$;
2. a partition P of W into disjoint subsets $G_1, \ldots, G_m$; and
3. polynomials q_j of degree $\#(G_j)$ vanishing on G_j and a subset of samples from the irreducible component Z_{ij} that contains G_j and satisfies $\deg Z_{ij} > \#(G_j)$.

Observe that if $m = 2$, we are done since the only possibility is that the irreducible component Z_{i1} containing G_1 must also contain G_2, and has degree $\#(G_1) + \#(G_2)$. So we can assume $m \geq 3$.

Order the groupings by size

$$\#(G_1) \geq \#(G_2) \geq \ldots \geq \#(G_m).$$

Find the filtering polynomial p_1 for the irreducible component Z_{1j} containing G_1. This can be done by the technique of [20]. (In a sequel we will show how to take advantage of having already produced q_1.) Using the filtering polynomial p_1, we can check which G_j lie on Z_{i1}. This lets us remove at least two of the G_j from consideration. We now repeat this procedure each time decreasing m by at least two.

4 Numerical Conditioning: A Sensitivity Experiment

In this section, we have created some special systems to test the numerical behavior of our algorithms. We examine how the monodromy grouping algorithm behaves on polynomials in the neighborhood of a polynomial that factors.

Suppose we have a polynomial whose coefficients are very near to one that factors into several components. This could correspond to two opposing scenarios: in one, the given polynomial could be exact, and we hope to find that it does not factor, while in the other, the polynomial is a numerical approximation to the nearby factorizable polynomial, and we hope to find the approximate factorization. We wish to examine the behavior of the monodromy grouping algorithm under such conditions.

To conduct our experiment, we generate three dense quartics in two variables with random coefficients on the complex unit circle and multiplied them together to form a factorizable polynomial of degree 12. This polynomial is then perturbed, adding a random complex number of modulus ϵ to each coefficient. With mathematical exactness, none of the polynomials factors for any positive ϵ. For $\epsilon = 10^{-i}$, $i = 0, 1, \ldots, 14$, a collection of 15 test polynomials is obtained. The test suite consisted of six such collections. The algorithm **MonodromyGrouping** was applied to all polynomials in the test suite, with termination condition $S = 10$. All calculations here were done with standard machine arithmetic (16 decimal places with double precision floating-point numbers). The results are summarized in Table 1.

For the case of perturbations with $\epsilon = 1$, we see from the top row of Table 1, that in all six cases **MonodromyGrouping** found that all twelve roots belong to a single component; that is, the polynomial does not factor. In contrast, for $\epsilon = 10^{-14}$, the polynomial was always predicted to break up into three factors. As ϵ gets smaller, the test polynomial resembles more and more a product, monodromy actions are harder to find and the algorithm terminates without connecting the points into a single component.

This behavior of the algorithm is expected: for sufficiently small perturbations the test polynomial is numerically indistinguishable from the nearby factorizable polynomial. For different perturbations on different coefficients it is hard to quantify this effect precisely —

ϵ	c	n	c	n	c	n	c	n	c	n	c	n
1.0E+00	1	4	1	3	1	3	1	3	1	3	1	5
1.0E–01	1	6	1	3	1	3	1	7	1	4	1	4
1.0E–02	1	7	1	3	1	3	1	9	1	14	1	5
1.0E–03	1	11	1	16	2	14	1	5	1	14	1	7
1.0E–04	2	19	2	13	1	11	3	12	2	13	3	15
1.0E–05	3	12	2	13	3	13	3	15	3	14	2	19
1.0E–06	3	15	2	21	3	13	3	12	3	14	3	13
1.0E–07	3	17	2	24	3	14	3	13	3	14	3	15
1.0E–08	3	16	3	14	2	13	3	15	3	15	3	13
1.0E–09	3	17	3	17	3	15	3	12	3	13	2	16
1.0E–10	3	14	3	13	3	16	3	14	3	15	3	16
1.0E–11	3	14	3	17	3	12	3	12	3	12	3	19
1.0E–12	3	13	3	16	3	13	3	14	2	25	3	14
1.0E–13	3	14	3	13	3	14	3	13	3	12	3	13
1.0E–14	3	13	3	18	3	14	3	14	3	12	3	12

Table 1: Six experiments on a perturbed product of three quartics, for various values ϵ of the magnitude of the error. The column header "c" lists the number of components, while "n" is the number of iterations needed to acquire this factorization.

we cannot really point out a clear threshold for ϵ. When the perturbations are numerical artifacts, such as roundoff, it is useful to discover a nearby factored form.

5 Applications

The algorithms in this paper have been implemented as a separate module of PHCpack [23]. All computations where done on a dual processor Pentium III 800 Mhz Linux machine.

5.1 The Cyclic n-roots Problem

The cyclic n-roots problems is one of the most notorious benchmark systems for polynomial system solvers, brought to the computer algebra community in [5]. The systems come from an application involving Fourier transforms, see [1], [2], and [3]. R. Fröberg conjectured (reported in [17]) that, if n has a quadratic divisor, then there are infinitely many solutions and that, in case the number of solutions is finite, this number is $\dfrac{(2n-2)!}{(n-1)!^2}$. U. Haagerup showed in [11] that for n prime, the number of solutions is always finite and confirmed the conjectured number of solutions.

In this section we confirm earlier results obtained in [4] (for $n = 8$) and in [10] (for $n = 9$). Our former methods of [20] and [21] were limited to the reduced version (credited to J. Canny [8]) of this problem. See [9] for polyhedral root counts. For $n = 10$ and $n = 11$, all solutions are isolated and thus "easier" to solve numerically (see the companion web site to [23] with test examples for the solution to these problems). Note that G. Björck found all distinct isolated 184,756 cyclic 11-roots (reported as unpublished result in [11]). Here we report on how homotopy methods bridged the gap for this problem between $n = 7$ and $n = 10$. Faster computers are needed for $n = 12$.

5.1.1 The Cyclic 8-roots Problem

Starting from the given list of generic points, the decomposition of the one dimensional component of the cyclic 8-roots system [4] of degree 144 was achieved in 6m 24s 930ms user CPU time. This took 21 iterations of the algorithm. The drop in cardinalities of the partition went as follows:

$$\begin{array}{c} 144 \rightarrow 102 \rightarrow 70 \rightarrow 63 \rightarrow 33 \rightarrow 30 \rightarrow 29 \rightarrow 18 \\ \rightarrow 18 \rightarrow 18 \rightarrow 18 \rightarrow 16 \rightarrow \cdots \end{array} \tag{6}$$

where the last ten iterations were stable. So the monodromy breakup predicted 16 components: eight quadrics and eight curves of degree 16. This breakup was subsequently confirmed with Newton interpolation. The numerical results of the certification by interpolating filtering polynomials are presented in Table 2. The quadrics were computed with standard arithmetic, while 32 decimal places were used for the 16-th degree polynomials. It took 41m 54s 780ms user CPU time to complete the certification process.

The test points are samples used to determine the linear span of the component. In case of the quadrics, we got four test points and with the 16 degree polynomials six test points were used. Since only the magnitude of the residual is important, we list –16 instead of 3.254E–16.

5.1.2 The Cyclic 9-roots Problem

The cyclic 9-roots problem has a two dimensional solution component of degree 18. We found this to break up into six components each of degree three. The cardinalities in the partition reduce as follows:

$$18 \rightarrow 18 \rightarrow 14 \rightarrow 11 \rightarrow 6 \rightarrow \cdots \tag{7}$$

where the last ten iterations were stable.

Achieving this breakup takes only 2m 32s 400ms. The approach of [20, 21] finishes in about the same time. Table 3 displays the numerical results of the symbolic certification, once with standard floating-point machine arithmetic and once with multi-precision floating numbers of 32 decimal places long. This validation required 59s 250ms and 14m 56s 570ms for the respective machine and multi-precision numbers. The results in the first half of Table 3 are somehow "lucky": in many computed instances machine arithmetic did not lead to small residuals. At the expense of a slowdown with a factor 15 we always get reliable results with 32 decimal places.

Polyhedral homotopies are required to exploit the sparse structure of the cyclic n-roots problem. For $n = 9$, the program available with the paper [16] has been used to compute the mixed volume of the polynomial system, sliced and embedded according to the techniques of [19]. With the program of [16], it took 13m 4s 540ms to compute the mixed volume of the embedded system. Tracing all 20,376 paths to solve a random coefficient start system

d	eps	distance	grid res	test res
2	6.877E–16	3.680E+00	1.665E–16	–14,–14,–14,–14
2	1.113E–15	3.961E+00	4.663E–15	–14,–14,–14,–15
2	4.909E–16	6.719E+00	4.330E–15	–16,–16,–15,–14
2	5.532E–16	3.639E+00	5.801E–15	–14,–13,–14,–14
2	2.211E–15	5.456E+00	1.665E–15	–15,–15,–15,–15
2	1.717E–15	7.517E+00	5.551E–15	–15,–14,–14,–13
2	4.116E–16	4.129E+00	1.941E–16	–13,–14,–13,–14
2	7.944E–16	7.286E+00	2.442E–15	–15,–15,–15,–15
16	1.387E–27	1.718E+00	9.700E–21	–16,–24,–24,–24,–31,–16
16	2.584E–28	1.026E+00	2.200E–21	–17,–23,–15,–22,–23,–17
16	1.547E–28	1.026E+00	1.130E–21	–18,–18,–24,–21,–19,–15
16	1.199E–25	1.732E+00	5.000E–20	–28,–28,–23,–25,–28,–23
16	6.074E–27	1.734E+00	1.600E–20	–27,–26,–25,–29,–26,–24
16	1.201E–27	1.695E+00	2.700E–20	–29,–30,–17,–30,–23,–23
16	2.290E–28	1.053E+00	5.100E–21	–12,–20,–20,–21,–17,–16
16	3.399E–27	1.053E+00	1.100E–21	–25,–17,–19,–11,–15,–16

Table 2: Numerical results of the certification of cyclic 8-roots. The columns contain the degree d, the maximal error (eps) on the samples in the grid, the minimal distance between the samples, the largest value of the interpolating filter evaluated at all samples (grid res) and at the test points (test res) used to compute the linear span of the component.

required with PHC 4h 4m 29s 730ms, and solving the embedded system to reach the 18 generic witness points took an additional 4h 54m 53s 550ms. Thus, we see that the computation of the decomposition by monodromy, even when using multiple precision in the validation, is small compared to the overhead of computing the witness points.

5.2 Adjacent Minors of a General $2 \times (n+1)$-Matrix

In [6], it was shown that the number of components of the ideal of all adjacent 2×2-minors of a general $2 \times (n+1)$-matrix is radical, of degree 2^n and that it breaks up into F_n components, F_n being the nth Fibonacci number. We found this system an interesting benchmark. See [12] for methods dedicated to binomial ideals. Table 4 illustrates the performance of **MonodromyGrouping** on this class of systems. Compared to our earlier methods in [20, 21], three more cases could be solved.

Results on the symbolic certification for the case $n = 10$ are presented here. There are 20 components of degree less than or equal to five which were treated with standard machine arithmetic, see Table 5. Note that in Table 5, there is no interpolating polynomial constructed for $d = 1$, since in that case the linear span completely describes the component. To interpolate the other 69 higher degree components 64 decimal places were used, see Table 6 and Table 7. This certification took 23h 56m 28s 850ms, and is thus a lot more expensive than the prediction of the breakup 3h 6m 48s 750ms.

d	eps	distance	grid res	test res
3	1.618E–11	2.310E+00	2.383E–11	–12,–10,–11,–11,–11
3	2.484E–11	2.335E+00	9.526E–11	–10,–11,–10,–11,–11
3	4.631E–11	1.837E+00	7.218E–11	–12,–12,–12,–11,–11
3	4.561E–11	1.818E+00	2.360E–09	–10,–9,–9,–9,–9
3	6.438E–11	2.597E+00	3.986E–10	–11,–12,–11,–12,–11
3	2.193E–11	1.515E+00	4.687E–11	–11,–8,–9,–9,–8
3	1.470E–26	2.103E+00	2.712E–26	–26,–26,–26,–26,–26
3	1.063E–26	2.812E+00	1.642E–24	–24,–25,–24,–25,–24
3	9.400E–27	1.972E+00	7.565E–27	–28,–29,–27,–27,–27
3	4.283E–27	2.363E+00	5.765E–26	–26,–26,–25,–25,–26
3	1.238E–26	2.158E+00	7.215E–25	–26,–26,–26,–26,–25
3	1.493E–26	2.243E+00	5.202E–26	–25,–27,–27,–25,–26

Table 3: Numerical results of the certification of cyclic 9-roots, first done with standard floating-point arithmetic and in the second half of the table redone with 32 decimal places. The columns contain the degree d, the maximal error (eps) on the samples in the grid, the minimal distance between the samples, the largest value of the interpolating filter evaluated at all samples (grid res) and at five test points (test res).

5.3 A Moving Stewart-Gough Platform

Stewart-Gough platforms are mechanical devices consisting of a rigid base and a rigid endplate, joined via six legs using ball joint connections. In motion simulators and other robotic applications, the lengths of the legs are actuated under computer control to move the endplate with respect to the base. Generally, once the leg lengths are fixed, the entire structure becomes rigid, but the same set of leg lengths may be compatible with multiple endplate locations. The problem of determining all possible endplate locations given the leg lengths has a long history. For generic choices of the mechanical parameters, the problem has forty isolated solutions, a fact first established by continuation [18] and later proven analytically [24, 13]. One of the more recent results [7] involved the demonstration (obtained by methods of numerical homotopy continuation) that platforms exist that have the forty real solutions.

For special choices of the parameters, a Stewart-Gough platform may have solution curves or other higher dimensional solution components, instead of only isolated solutions. We have tested a special case called “Griffis-Duffy type” by [14]. This mechanism has, besides 12 lines (which correspond to degenerate assemblies), a single solution curve of degree 28. Our monodromy method confirmed this result by finding connections between all of the generic points on that solution curve. Computing the 40 generic points with PHC took 1m 12s 480ms cpu time. The drop in cardinalities in the partitions generated by **MonodromyGrouping** went as follows

$$40 \rightarrow 28 \rightarrow 27 \rightarrow 22 \rightarrow 17 \rightarrow 16 \rightarrow 15 \rightarrow 13 \rightarrow \cdots \tag{8}$$

where the last 10 iterations were stable. It took only 33s 430ms to achieve this result. At first glance, the result of degree 28 for the curve may appear to be in conflict with Husty

n	d	c	it	User CPU time
3	8	3	15	3s 260ms
4	16	5	16	15s 670ms
5	32	8	17	43s 340ms
6	64	13	20	2m 19s 140ms
7	128	21	27	8m 47s 940ms
8	256	34	22	20m 20s 420ms
9	512	55	20	45m 44s 50ms
10	1024	89	35	3h 6m 48s 750ms
11	2048	144	24	6h 6m 27s 890ms

Table 4: Results of the monodromy breakup algorithm on the systems of all adjacent minors of a general $2\times(n+1)$-matrix. The sum of the degrees $d = 2^n$, c is the number of components, "it" the number of iterations, and lastly the User CPU time.

and Karger, who claim that the curve is of degree 20. The conflict is resolved by noting that the curve is degree 28 in the full space of rotation and translation (represented in Study coordinates), but its degree falls to 20 when the curve is projected onto its rotational component only.

Since the lines correspond to degenerate assemblies (validating a line requires only three samples anyway), no further validation with interpolation is needed for this problem. This is fortunate, because validation for high degree components can be expensive due to the number of monomials that appear and the numerical sensitivity of high degree equations, forcing the use of multi-precision arithmetic. If one were to compute an interpolating polynomial for the case at hand (even though this is not necessary), one would find that a general polynomial of degree 28 in two variables has 435 monomials. Different methods to construct the interpolating polynomial require between 435 (direct approach with linear system) and 812 (Newton interpolation) samples. While these numbers are modest for homotopies on modern machines, the use of software driven multi-precision arithmetic imposed by the relatively high degree will constitute a serious speed bump.

This is illustrated in Table 8, which displays the results of the validation for the curve of degree 28, executed with 64 decimal places as working precision. The Newton form of the interpolating polynomial constructed with generalized divided differences required 812 samples. Creating the grid of 812 samples took 39m 53s 960ms, which is of the same magnitude as 37m 47s 920ms, which is the time it took to evaluate all 812 samples in that high degree interpolating polynomial. While other operations (finding linear span of the component, construction Newton form, and evaluating at extra test samples) were relatively small, the total time to get the results in Table 8 was 1h 19m 13s 110ms.

6 Conclusions

Using the monodromy we presented an algorithm to predict the breakup of a positive dimensional component into irreducible ones. This predicted breakup is then subsequently

d	eps	distance	grid res	test res
4	1.343E–14	9.002E–01	7.730E–14	–13,–16,–14,–12,–15
3	3.068E–14	7.500E–01	6.772E–13	–12,–10,–9,–10,-12
4	4.646E–14	7.567E–01	5.144E–13	–11,–9,–12,–13,–12
3	1.856E–12	5.789E–01	1.778E–13	–14,–15,–11,–12,–14
4	1.496E–14	6.836E–01	8.475E–11	–11,–10,–9,–6,–7
4	5.168E–14	1.378E+00	1.776E–13	–10,–9,–14,–13,–11
4	2.457E–14	5.508E–01	8.704E–14	–15,–12,–12,–15,–14
4	6.305E–15	3.400E–01	3.708E–14	–14,–15,–10,–12,–10
5	8.814E–14	5.100E–01	1.819E–10	–12,–13,–12,–13,–13,–13,–12
4	7.439E–15	1.122E+00	7.849E–14	–13,–14,–15,–15,–15
3	1.940E–15	6.817E–01	1.035E–13	–11,–12,–12,–13,–12
5	5.953E–12	1.145E+00	2.612E–11	–3,–9,–10,–6,–2,–11,–9
3	1.103E–13	1.028E+00	6.972E–14	–11,–11,–12,–14,–12
1	–	–	–	–
3	1.446E–14	1.327E+00	1.730E–12	–11,–10,–9,–11,–10
4	4.056E–14	1.357E+00	1.674E–13	–14,–10,–14,–12,–11
4	1.095E–14	5.225E–01	3.908E–14	–11,–12,–9,–12,–11
5	3.672E–14	1.780E+00	5.586E–13	–5,–8,–3,–8,–13,–9,–11
5	6.821E–12	9.090E–01	2.274E–11	–9,–13,–8,–9,–13,–6,–10
4	1.828E–14	7.402E–01	8.527E–14	–12,–14,–14,–13,–14

Table 5: Numerical results of the certification of the system of adjacent minors, for $n = 10$, for components of degree $d \leq 5$. The columns furthermore contain the maximal error (eps) on the samples in the grid, the minimal distance between the samples, the largest value of the interpolating filter evaluated at all samples (grid res) and at the test points (test res) used to compute the linear span of the component.

validated by computing interpolating polynomials.

Compared to our previous approaches described in [20, 21], we point out several advantages. First of all, in almost all cases, standard floating-point machine arithmetic suffices to execute the algorithm **MonodromyGrouping**. This has put more difficult applications within our reach. Related to this issue is the experience that the running times for this breakup remain of the same order of magnitude regardless of the geometry of the breakup. For example, whether a curve of degree forty breaks up in two, or in twenty pieces does not cause major fluctuations in the needed running time. By contrast, the symbolic validation of two curves of degree 20 is much more expensive than the interpolation of twenty quadrics. Thirdly, with the predicted breakup we can set up a structured grid of samples and apply generalized divided differences [15] to construct the polynomial equations that cut out the components with Newton interpolation. We observed an improved conditioning of the interpolation problem and will elaborate on this in a sequel.

References

[1] G. Björck. Functions of modulus one on Z_p whose Fourier transforms have constant modulus. In *Proceedings of the Alfred Haar Memorial Conference, Budapest*, volume 49 of *Colloquia Mathematica Societatis János Bolyai*, pages 193–197. 1985.

[2] G. Björck. Functions of modulus one on Z_n whose Fourier transforms have constant modulus, and "cyclic n-roots". In J.S. Byrnes and J.F. Byrnes, editors, *Recent Advances in Fourier Analysis and its Applications*, volume 315 of *NATO Adv. Sci. Inst. Ser. C: Math. Phys. Sci.*, pages 131–140. Kluwer, 1989.

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

[4] G. Björck and R. Fröberg. Methods to "divide out" certain solutions from systems of algebraic equations, applied to find all cyclic 8-roots. In M. Gyllenberg and L.E. Persson, editors, *Analysis, Algebra and Computers in Math. research*, volume 564 of *Lecture Notes in Applied Mathematics*, 57–70. Marcel Dekker, 1994.

[5] J. Davenport. Looking at a set of equations. Technical report 87-06, Bath Computer Science, 1987.

[6] P. Diaconis, D. Eisenbud, and B. Sturmfels. Lattice Walks and Primary Decomposition. In B.E. Sagan, R.P. Stanley, editors: *Mathematical Essays in Honor of Gian-Carlo Rota*, volume 161 of *Progress in Mathematics*, 173–193, Birkhäuser, 1998.

[7] P. Dietmaier. The Stewart-Gough platform of general geometry can have 40 real postures. In J. Lenarcic and M.L. Husty, editors, *Advances in Robot Kinematics: Analysis and Control*, pages 1–10. Kluwer Academic Publishers, Dordrecht, 1998.

[8] I.Z. Emiris. *Sparse Elimination and Applications in Kinematics.* PhD thesis, Computer Science Division, Dept. of Electrical Engineering and Computer Science, University of California, Berkeley, 1994.

[9] I.Z. Emiris and J.F. Canny. Efficient incremental algorithms for the sparse resultant and the mixed volume. *J. Symbolic Computation*, 20(2):117–149, 1995. Software available at `http://www.inria.fr/saga/emiris`.

[10] J.C. Faugère. A new efficient algorithm for computing Gröbner bases (F_4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999. Proceedings of MEGA'98, 22–27 June 1998, Saint-Malo, France.

[11] U. Haagerup. Orthogonal maximal abelian *-algebras of the n x n matrices and cyclic n-roots. in *Operator Algebras and Quantum Field Theory*, 296–322. International Press, Cambridge, MA, 1996.

[12] S. Hosten and J. Shapiro. Primary Decomposition of Lattice Basis Ideals. *Journal of Symbolic Computation* 29(4&5): 625–639, 2000.

[13] M.L. Husty An algorithm for solving the direct kinematics of general Stewart-Gough Platforms *Mech. Mach. Theory,* 31(4):365–380, 1996.

[14] M.L. Husty and A. Karger Self-motions of Griffis-Duffy type parallel manipulators *Proc. 2000 IEEE Int. Conf. Robotics and Automation*, CDROM, San Francisco, CA, April 24–28, 2000.

[15] Isaacson, E. and Keller, H.B. *Analysis of Numerical Methods.* Dover Publications, 1994.

[16] T.Y. Li and X. Li. Finding mixed cells in the mixed volume computation. To appear in *Found. Comput. Math.* Software available at `http://www.math.msu.edu/~li`.

[17] H.M. Möller. Gröbner bases and numerical analysis. In B. Buchberger and F. Winkler, editors, *Gröbner Bases and Applications*, volume 251 of *London Mathematical Lecture Note Series*, pages 159–178. Cambridge University Press, 1998.

[18] M. Raghavan The Stewart platform of general geometry has 40 configurations *ASME J. Mech. Design,* 115:277–282, June 1993.

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

[20] 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.

[21] A.J. Sommese, J. Verschelde and C.W. Wampler. Numerical irreducible decomposition using projections from points on the components. Accepted by *Contemporary Mathematics.* Available at `http://www.nd.edu/~sommese` and `http://www.math.uic.edu/~jan`.

[22] 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.

[23] 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`.

[24] C.W. Wampler. Forward displacement analysis of general six-in-parallel SPS (Stewart) platform manipulators using soma coordinates. *Mech. Mach. Theory* 31(3): 331–337, 1996.

Andrew J. Sommese
Department of Mathematics
University of Notre Dame
Notre Dame, IN 46556, U.S.A.
sommese@nd.edu
http://www.nd.edu/~sommese

Jan Verschelde
Department of Mathematics, Statistics, and Computer Science
University of Illinois at Chicago
851 South Morgan (M/C 249)
Chicago, IL 60607-7045, U.S.A.
jan@math.uic.edu
jan.verschelde@na-net.ornl.gov
http://www.math.uic.edu/~jan

Charles W. Wampler
General Motors Research Laboratories
Enterprise Systems Lab
Warren, MI 48090, U.S.A.
Charles.W.Wampler@gm.com

d	eps	distance	grid res	test res
12	8.361E–57	1.007E–01	1.085E–42	–45,–44,–44,–41,–44,–43,–44,–43,–39
20	9.967E–57	2.416E–01	4.340E–45	–56,–58,–41,–49,–45,–48,–45,–37,–53
8	9.344E–57	4.516E–01	5.000E–64	–61,–57,–63,–63,–61,–53,–63
12	4.368E–55	1.071E+00	6.300E–54	–51,–52,–50,–50,–49,–52,–51,–50,–49
8	9.911E–57	4.453E–01	7.600E–64	–59,–55,–53,–54,–57,–60,–61
8	6.377E–57	3.961E–01	4.346E–54	–57,–55,–54,–55,–57,–60,–55
15	8.365E–57	5.533E–01	9.200E–55	–45,–41,–26,–55,–60,–57,–54,–54,–56
15	9.385E–57	5.574E–01	4.016E–54	–47,–50,–46,–42,–51,–45,–46,–49,–46
24	9.712E–57	1.490E–01	4.700E–44	–57,–38,–6,–31,–20,–22,–51,–48,–42
21	4.148E–59	2.994E–01	3.900E-47	–22,–37,–45,–46,–39,–43,–45,–49,–33,–51,–41
16	9.370E–57	6.769E–01	3.901E–49	–53,–51,–56,–36,–36,–53,–36
8	6.826E–57	5.912E–01	1.110E–56	–50,–50,–50,–51,–52,–52,–52
16	1.089E–58	2.743E–01	5.270E–46	–42,–54,–37,–50,–56,–52,–53,–49,–38,–58,–42
20	9.991E–57	2.057E–01	1.920E–47	–45,–42,–59,–55,–43,–58,–46,–46,–49
12	9.229E–59	7.635E–01	1.623E–44	–37,–38,–37,–38,–37,–39,–37
16	8.607E–57	2.514E–01	4.800E–51	–62,–54,–49,–65,–44,–67,–65,–46,–57
16	3.796E–59	1.796E–01	2.620E–49	–58,–53,–54,–53,–49,–29,–53,–45,–39,,–22,–57
24	4.618E–57	4.110E–01	6.660E–36	–30,–31,–29,–36,–34,–26,–32,–27,–28
24	2.141E–59	2.436E–01	2.894E–42	–51,–57,–56,–51,–58,–60,–43,–52,–58,–55,–58
8	9.755E–57	7.382E–01	7.000E–58	–59,–54,–58,–57,–56,–55,–55
9	9.659E–57	5.337E–01	1.020E–58	–59,–54,–49,–57,–57,–52,–53
16	5.846E–59	7.444E–01	2.500E–48	–22,–45,–38,–28,–40,–35,–50,–38,–44
25	9.358E–57	1.996E–01	1.160E–47	–47,–9,–9,–25,–56,–39,–26,–54,–56,–56,–26
8	9.990E–57	4.346E–01	1.600E–57	–59,–61,–56,–61,–57,–54,–62
12	9.564E–57	2.609E–01	4.654E–48	–47,–51,–50,–50,–49,–48,–50
11	1.059E–57	7.889E–01	7.100E–49	–50,–49,–39,–54,–47,–53,–52,–48,–51,–54,–46,–54,–40
24	9.732E–57	3.537E–01	3.800E–46	–54,–19,–52,–42,–54,–38,–22,–20,–50
9	9.872E–57	2.681E–01	1.500E–57	–58,–60,–55,–59,–52,–58,–60
12	9.526E–57	3.511E–01	1.690E–57	–60,–56,–59,–55,–60,–49,–60
12	9.421E–57	6.753E–01	1.938E–53	–60,–56,–54,–49,–47,–60,–61,–55,–59
12	9.432E–57	3.142E–01	2.200E–57	–48,–45,–32,–60,–53,–47,–61
8	9.179E–57	6.102E–01	1.300E–61	–62,–57,–57,–60,–50,–60,–59
15	2.898E–59	4.977E–01	3.100E–51	–48,–44,–47,–43,–50,–49,–43,–45,–49
24	9.766E–57	2.674E–01	5.000E–41	–47,–39,–23,–36,–44,–47,–51,–39,–29
15	8.335E–57	2.226E–01	9.200E–52	–56,–54,–58,–59,–54,–53,–49,–55,–50

Table 6: Numerical results of the certification of the system of adjacent minors, for $n = 10$, for components of degree $d \leq 5$, part A. The columns furthermore contain the maximal error (eps) on the samples in the grid, the minimal distance between the samples, the largest value of the interpolating filter evaluated at all samples (grid res) and at the test points (test res) used to compute the linear span of the component.

d	eps	distance	grid res	test res
12	9.840E–57	5.607E–01	1.111E–56	–55,–39,–41,–42,–49,–53,–41
24	2.029E–56	4.163E–01	6.310E–17	–24,–29,–21,–23,–20,–27,–27,–30,–18
12	9.009E–57	4.460E–01	7.500E–54	–40,–51,–49,–50,–55,–53,–44
27	9.903E–57	2.949E–01	1.690E–33	–30,–24,–30,–19,–47,–34,–41,–34,–40
9	3.759E–59	9.179E–01	7.600E–60	–54,–51,–58,–50,–54,–59,–55,–50,–49,–50,–48
8	9.172E–60	3.264E–01	3.955E–58	–53,–48,–52,–52,–55,–53,–50
16	9.685E–57	3.339E–01	1.200E–52	–56,–57,–55,–54,–44,–46,–51,–53,–53
7	1.553E–58	1.340E+00	1.990E–54	–56,–53,–58,–57,–53,–57,–58,–55,–56
12	9.030E–57	9.606E–01	1.000E–54	–52,–42,–54,–60,–45,–56,–59,–52,–53
15	9.849E–57	1.732E–01	3.973E–55	–40,–59,–52,–59,–59,–55,–54,–43,–33
9	1.678E–58	7.574E–01	2.700E–59	–53,–47,–53,–55,–53,–56,–52,–50,–51,–44,–53
12	9.657E–57	2.657E–01	6.000E–57	–38,–58,–55,–53,–62,–49,–50
12	9.948E–57	6.428E–01	1.717E–53	–49,–58,–50,–51,–46,–51,–54
9	9.934E–57	4.570E–01	3.500E–58	–56,–56,–56,–57,–57,–57,–53
12	9.886E–57	1.096E+00	1.803E–53	–53,–42,–52,–31,–56,–27,–54
8	9.920E–57	4.307E–01	2.700E–61	–61,–54,–59,–60,–59,–59,–58
21	4.976E–57	2.592E–01	1.486E–49	–45,–46,–43,–43,–45,–46,–41,–49,–42,–49,–49
9	8.021E–57	8.084E–01	2.200E–57	–42,–56,–52,–57,–52,–57,–55
12	1.824E–57	7.706E–01	9.940E–50	–44,–49,–47,–43,–44,–46,–48,–53,–47
7	2.164E–57	6.063E–01	3.500E–59	–54,–51,–50,–47,–54,–57,–58,–52,–50
7	5.919E–59	3.709E–01	1.180E–57	–55,–55,–53,–51,–49,–55,–55,–51,–55
24	4.255E–58	1.023E–01	6.830E–26	–32,–36,–25,–29,–32,–31,–36,–28,–32,–39,–27
9	1.114E–58	5.179E–01	1.087E–47	–43,–40,–43,–41,–43,–42,–43
12	8.536E–59	4.588E–01	5.020E–47	–50,–42,–37,–47,–41,–39,–43,–45,–42
12	7.665E–57	5.159E–01	3.828E–57	–56,–49,–58,–54,–48,–50,–53
8	8.207E–57	1.138E–01	4.100E–59	–62,–58,–58,–62,–63,–62,–55
12	9.968E–57	3.490E–01	2.000E–50	–54,–50,–46,–46,–50,–47,–44
9	2.355E–57	8.551E–02	2.000E–56	–49,–53,–51,–56,–50,–52,–54
24	9.932E–57	1.234E–01	3.166E–13	–48,–46,–47,–47,–40,–41,–41,–48,–41
12	9.167E–57	4.959E–01	1.139E–50	–48,–50,–44,–48,–46,–33,–44
8	1.498E–58	7.819E–01	6.425E–53	–57,–55,–56,–54,–57,–55,–55
8	9.153E–57	8.786E–01	1.190E–56	–53,–55,–54,–59,–48,–54,–55
15	4.202E–58	4.008E–01	1.971E–37	–23,–21,–23,–23,–20,–21,–23,–23,–22
20	9.680E–57	3.669E–01	1.000E–43	–55,–30,–39,–38,–40,–40,–47,–43,–34

Table 7: Numerical results of the certification of the system of adjacent minors, for $n = 10$, for components of degree $d \leq 5$, part B. The columns furthermore contain the maximal error (eps) on the samples in the grid, the minimal distance between the samples, the largest value of the interpolating filter evaluated at all samples (grid res) and at the test points (test res) used to compute the linear span of the component.

d	eps	distance	grid res	test res
28	1.316E-59	2.395E-01	3.800E-37	–49,–44,–48,–56,–40,–41,–20,–40,–57

Table 8: Numerical results of the certification of the system of a moving Stewart-Gough platform, done with 64 decimal places. The columns furthermore contain the maximal error (eps) on the samples in the grid, the minimal distance between the samples, the largest value of the interpolating filter evaluated at all samples (grid res) and at the test points (test res) used to compute the linear span of the component.