Solving polynomial systems via homotopy continuation and monodromy

Timothy Duff * Georgia Tech
Cvetelina Hill * Georgia Tech
Anders Jensen Aarhus Universitet
Kisun Lee * Georgia Tech
Anton Leykin * Georgia Tech
Jeff Sommars University of Illinois at Chicago

June 7, 2017

Abstract

We study methods for finding the solution set of a generic system in a family of polynomial systems with parametric coefficients. We present a framework for describing monodromy based solvers in terms of decorated graphs. Under the theoretical assumption that monodromy actions are generated uniformly, we show that the expected number of homotopy paths tracked by an algorithm following this framework is linear in the number of solutions. We demonstrate that our software implementation is competitive with the existing state-of-the-art methods implemented in other software packages.

Contents

*Research of TD, CH, KL, and AL is supported in part by NSF grant DMS-1151297.

1 Introduction

Homotopy continuation has become a standard technique to find approximations of solutions of polynomial systems. There is an early popular text on the subject and its applications by Morgan [26]. This technique is the backbone of Numerical Algebraic Geometry, the area which classically addresses the questions of complex algebraic geometry through algorithms that employ numerical approximate computation. The chapter by Sommese, Verschelde, and Wampler [30, §8] is the earliest introduction and the book by Sommese and Wampler [31] is the primary reference in the area.

Families of polynomial systems with parametric coefficients play one of the central roles. Most homotopy continuation techniques could be viewed as going from a generic system in the family to a particular one. This process is commonly referred to as *degeneration*. Going in the reverse direction, it may be called *deformation*, *undegeneration*, or *regeneration* depending on the literature. Knowing the solutions of a generic system one can use *coefficient-parameter homotopy* [31, §7] to get to the solution of a particular one.

The main problem that we address here is how to solve a generic system in a family of systems

$$F_p = (f_p^{(1)}, \dots, f_p^{(N)}) = 0, \quad f_p^{(i)} \in \mathbb{C}[p][x],\ i = 1, \dots, N,$$

with finitely many parameters p and n variables x. In the main body of the paper we restrict our attention to *linear parametric* families of systems, which we define as systems with affine linear parametric coefficients such that for a generic x there exists p with $F_p(x) = 0$. These restrictions are made for the sake of simplicity. We explain what modifications are needed to apply our approach in more general settings in §7.

Linear parametric systems form a large class that includes *sparse polynomial systems*. Such a system is a square ($n = N$) system with a fixed monomial support for each equation and a distinct parameter for the coefficient of each monomial. *Polyhedral homotopy* methods for solving sparse systems stem from the BKK (Bernstein, Khovanskii, Kouchnirenko) bound on the number of solutions [3]; the early work on algorithm development was done in [17, 33]. Polyhedral homotopies provide an optimal solution to sparse systems in the sense that they are designed to follow exactly as many paths as the number of solutions of a generic system (the BKK bound).

The method that we propose is clearly not optimal in the above sense. The expected number of homotopy paths followed can be larger than the number of solutions, though *not significantly larger*. We also use linear segment homotopies that are *significantly simpler* and less expensive to follow in practice. Our current implementation shows it is competitive with the state-of-the-art implementations of polyhedral homotopies in PHCpack [32] and HOM4PS2 [19] for solving sparse systems. In a more general than sparse setting, we demonstrate examples of linear parametric systems for which our implementation exceeds the capabilities of the existing sparse system solvers and blackbox solvers based on other ideas.

The idea of using the monodromy action induced by the fundamental group of the regular locus of the parameter space has been successfully employed throughout Numerical Algebraic Geometry. One of the main tools in the area, *numerical irreducible decomposition*, can be efficiently implemented using the *monodromy breakup* algorithm, which first appeared in [28]. One parallel incarnation of the monodromy breakup algorithm is described in [23]. In fact, the main idea in that work is close in spirit to what we propose in this article. The idea to use monodromy to find solutions drives numerical implicitization [6] and appears in other works such as [7]. Computing monodromy groups numerically, as in [21] and [15], requires more computation than just finding solutions. One can approach this computation with the same methodology as we propose (§7.4).

Our main contribution is a new framework to describe algorithms for solving polynomial systems using monodromy. We analyze the complexity of our main algorithm both theoretically assuming a certain statistical model and experimentally on families of examples. The analysis gives us grounds to say that the expected number of paths tracked by our method is linear, with a small coefficient, as the number of solutions grows. Our method and its implementation not only provide a new general tool for solving polynomial systems, but also can solve some problems out of reach for other existing software.

The structure of the paper is as follows. We give a brief overview of our method intermingled with some necessary preliminaries in §2. An algorithm

following our framework depends on a choice of strategy, with several possibilities outlined in §3. Statistical analysis of the method is the topic of §4. The implementation is discussed in §5 together with the side topic of *certification* of the solution set. The results of our experiments on selected example families highlighting various practical computational aspects are in §6. The reader may also want look at examples of systems in §6.1 and §6.2 before reading some earlier sections. Possible generalizations of our method and the future directions to explore are presented in §7.

2 Background and framework overview

Let $m, n \in \mathbb{N}$. We consider the complex linear space of square systems F_p, $p \in \mathbb{C}^m$, where the monomial support of $f_p^{(1)}, \ldots, f_p^{(n)}$ in the variables $x = (x_1, \ldots, x_n)$ is fixed and the coefficients vary. By a *base space* B we mean a parametrized linear variety of systems. We think of it as the image of an affine linear map $\varphi : p \mapsto F_p$ from a parameter space $\mathbb{C}^m$ with coordinates $p = (p_1, \ldots, p_m)$ to the space of systems.

We assume the structure of our family is such that the projection π from the *solution variety*

$$V = \{(F_p, x) \in B \times \mathbb{C}^n \mid F_p(x) = 0\}$$

to B gives us a *branched covering*, i.e., the fiber $\pi^{-1}(F_p)$ is finite of the same cardinality for a generic p. The discriminant variety D in this context is the subset of the systems in the base space with nongeneric fibers and is also known as the *branch locus* of π.

Our goal is to find the fiber of one generic system in our family. Our method is to find one pair $(p_0, x_0) \in V$ and use the monodromy action of the fundamental group of $B \setminus D$ on the fiber $\pi^{-1}(F_{p_0})$ to find its points. We assume that this action is transitive, which is the case if and only if the solution variety V is irreducible. If V happens to be reducible, we replace V with its unique dominant irreducible component as explained in Remark 2.1.

2.1 Monodromy

We briefly review the basic facts concerning monodromy groups of branched coverings. With notation as before, fix a system $F_p \in B \setminus D$ and consider a loop τ without branch points based at F_p; that is, a continuous path

$$\tau : [0, 1] \to B \setminus D$$

such that $\tau(0) = \tau(1) = F_p$. Suppose we are also given a point x_i in the fiber $\pi^{-1}(F_p)$ with d points $x_1, x_2, \ldots, x_d$. Since π is a covering map, the pair (τ, x_i) corresponds to a unique *lifting* $\widetilde{\tau}_i$, a path

$$\widetilde{\tau}_i : [0, 1] \to V$$

such that $\widetilde{\tau_i}(0) = x_i$ and $\widetilde{\tau_i}(1) = x_j$ for some $1 \leq j \leq d$. Note that the reversal of τ and x_j lift to a reversal of $\widetilde{\tau_i}$. Thus, the loop τ induces a permutation of the set $\pi^{-1}(F_p)$. We have a group homomorphism

$$\varphi : \pi_1(B \setminus D, F_p) \to S_d$$

whose domain is the usual fundamental group of $B \setminus D$ based at F_p. The image of φ is the *monodromy group* associated to $\pi^{-1}(F_p)$. The monodromy group acts on the fiber $\pi^{-1}(F_p)$ by permuting the solutions of F_p.

We have not used any algebraic properties so far. The construction of the monodromy group above holds for an arbitrary covering with finitely many sheets. The monodromy group is a transitive subgroup of S_d whenever the total space is connected. In our setting, since we are working over $\mathbb{C}$, this occurs precisely when the solution variety is irreducible.

Remark 2.1 For a linear family, we can show that there is at most one irreducible component of the solution variety V for which the restriction of the projection $(F_p, x) \mapsto x$ is dominant (that is its image is dense). We call such component the *dominant component*. Indeed, let U be the locus of points $(F_p, x) \in \pi^{-1}(B \setminus D)$ such that

- the restriction of x-projection map is locally surjective, and
- the solution to the linear system of equations $F_p(x) = 0$ in p has the generic dimension.

Being locally surjective could be interpreted either in the sense of Zariski topology or as inducing surjection on the tangent spaces. Then either U is empty or $\overline{U}$ is the dominant component we need, since it is a vector bundle over an irreducible variety, and is hence irreducible.

In the rest of the paper, when we say *solution variety*, we mean the *dominant component of the solution variety*. In particular, for sparse systems restricting the attention to the dominant component translates into looking for solutions only in the torus $(\mathbb{C}^*)^n$.

2.2 Homotopy continuation

Given two points F_{p_1} and F_{p_2} in the base space B, we may form the family of systems

$$H(t) = (1-t)F_{p_1} + tF_{p_2}, \quad t \in [0,1],$$

known as the *linear segment homotopy* between the two systems. If p_1 and p_2 are sufficiently generic, for each $t \in [0,1]$ we have $H(t)$ outside the real codimension 2 set D. Consequently, each system $H(t)$ has a finite and equal number of solutions.

Remark 2.2 Note that γF_p for $\gamma \in \mathbb{C} \setminus \{0\}$ has the same solutions as F_p. Let us scale both ends of the homotopy by taking a homotopy between $\gamma_1 F_{p_1}$ and $\gamma_2 F_{p_2}$ for generic γ_1 and γ_2. If the coefficients of F_p are homogeneous in p then

$$H'(t) = (1-t)\gamma_1 F_{p_1} + t\gamma_2 F_{p_2} = F_{(1-t)\gamma_1 p_1 + t\gamma_2 p_2}$$

is a homotopy in the family matching solutions $\pi^{-1}(F_{p_1})$ and $\pi^{-1}(F_{p_2})$ where the matching is potentially different from that given by $H(t)$. Similarly, for an affine linear family, $F_p = F'_p + C$ where F'_p is homogeneous in p and C is a constant system, we have

$$H'(t) = (1-t)\gamma_1 F_{p_1} + t\gamma_2 F_{p_2} = F'_{(1-t)\gamma_1 p_1 + t\gamma_2 p_2} + ((1-t)\gamma_1 + t\gamma_2)C.$$

Dividing the above by $(1-t)\gamma_1 + t\gamma_2$ we can write down another homotopy

$$H''(t) = F'_{\frac{(1-t)\gamma_1 p_1 + t\gamma_2 p_2}{(1-t)\gamma_1 + t\gamma_2}} + C = F_{\frac{(1-t)\gamma_1 p_1 + t\gamma_2 p_2}{(1-t)\gamma_1 + t\gamma_2}},$$

with the same homotopy paths as $H'(t)$, due to genericity, and staying in B.

Assuming that the solutions for F_{p_1} are known, methods of *numerical homotopy continuation*, described, for instance, in [31, §2.3], track the solutions as t changes from 0 to 1 thus finding the solutions of the system F_{p_2}. In some situations the path in B may pass close to the branch locus D and numerical issues must be considered.

Remark 2.3 If the family F_p is nonlinear in the parameters p, one has to take the *parameter linear segment homotopy* in the parameter space, i.e., $H(t) = F_{(1-t)p_1 + tp_2}$, $t \in [0,1]$. This does not change the overall construction; however, the freedom to replace the systems F_{p_1} and F_{p_2} at the ends of the homotopy with their scalar multiples as in Remark 2.2 is lost.

2.3 Graph of homotopies: main ideas

To organize the discovery of new solutions we represent the set of homotopies by a finite undirected graph G. Let $E(G)$ and $V(G)$ denote the edge and vertex set of G, respectively. Any vertex v in $V(G)$ is associated to a point F_p in the base space. An edge e in $E(G)$ connecting v_1 and v_2 in $V(G)$ is decorated with two complex numbers γ_1 and γ_2 and represents the linear homotopy connecting $\gamma_1 F_{p_1}$ and $\gamma_2 F_{p_2}$ along a line segment (Remark 2.2). We assume that both p_i and γ_i are chosen so that the segments do not intersect the branch locus. Choosing these at random satisfies the assumption, since the exceptional set of choices where such intersections happen is contained in a real Zariski closed set, see [31, Lemma 7.1.3].

We allow multiple edges between two distinct vertices but no loops, since the latter induce trivial homotopies. For a graph G to be potentially useful in a monodromy computation, it must contain a cycle. Some of the general ideas behind the structure of a graph G are listed below.

- For each vertex v_i, we maintain a subset of *known* points $Q_i \subset \pi^{-1}(F_{p_i})$.
- For each edge e between v_i and v_j, we record the two complex numbers γ_1 and γ_2 and we store the known partial correspondences $C_e \subset \pi^{-1}(F_{p_i}) \times \pi^{-1}(F_{p_j})$ between known points Q_i and Q_j.

- At each iteration, we pick an edge and direction, track the corresponding homotopy starting with yet unmatched points, and update known points and correspondences between them.
- We may obtain the initial "knowledge" as a *seed pair* (p_0, x_0) by picking $x_0 \in \mathbb{C}^n$ at random and choosing p_0 to be a generic solution of the linear system $F_p(x_0) = 0$.

We list basic operations that result in transition between one state of our algorithm captured by G, Q_i for $v_i \in V(G)$, and C_e for $e \in E(G)$ to another.

- For an edge $e = v_i \xleftrightarrow{(\gamma_1,\gamma_2)} v_j$, consider the homotopy

$$H^{(e)} = (1-t)\gamma_1 F_{p_i} + t\gamma_2 F_{p_j}$$

  where $(\gamma_1, \gamma_2) \in \mathbb{C}^2$ is the label of e.

  - Take start points S_i to be a subset of the set of known points Q_i that do not have an established correspondence with points in Q_j.
  - Track S_i along $H^{(e)}$ for $t \in [0, 1]$ to get $S_j \subset \pi^{-1}(F_{p_j})$.
  - Extend the known points for v_j, that is $Q_j := Q_j \cup S_j$ and record the newly established correspondences.
- Add a new vertex corresponding to F_p for a generic $p \in B \setminus D$.
- Add a new edge $e = v_i \xleftrightarrow{(\gamma_1,\gamma_2)} v_j$ between two existing vertices decorated with generic $\gamma_1, \gamma_2 \in \mathbb{C}$.

2.4 Graph of homotopies: examples

We demonstrate the idea of graphs of homotopies by giving two examples.

Example 2.4 Figure 1 shows a graph G with 2 vertices and 3 edges embedded in the base space B with paths partially lifted to the solution variety, which is a covering space with 3 sheets. The two fibers $\{x_1, x_2, x_3\}$ and $\{y_1, y_2, y_3\}$ are connected by 3 partial correspondences induced by the liftings of three egde-paths.

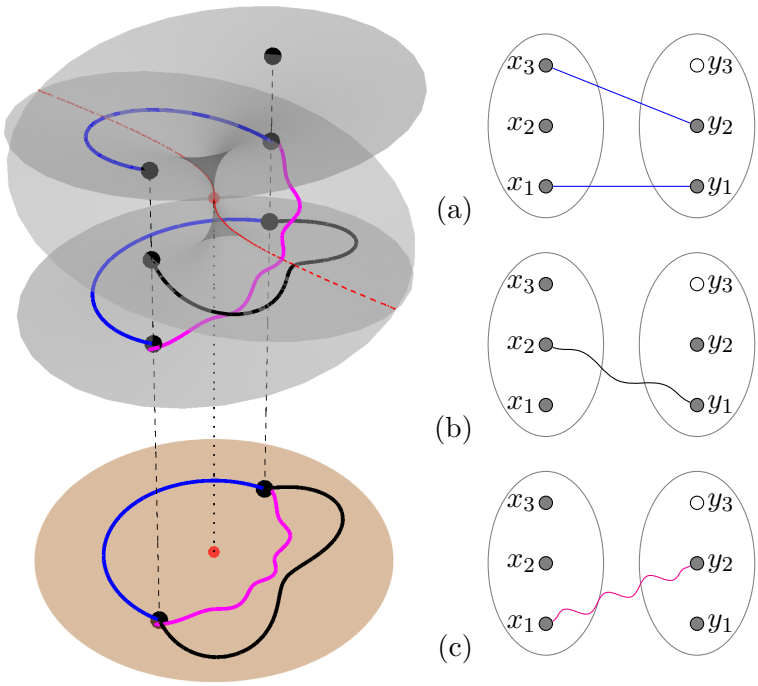

Figure 1: Selected liftings of 3 edges connecting the fibers of 2 vertices and induced correspondences.

Note that several aspects in this illustration are fictional. There is only one branch point in the actual complex base space B that we would like the reader to imagine. The visible self-intersections of the solution variety V are an artifact of drawing the picture in the real space. Also, in practice we use homotopy paths as simple as possible, however, here the paths are involved for the purpose of distinguishing them in print.

An algorithm that we envision may hypothetically take the following steps:

(1) *seed* the first fiber with x_1;

(2) use a lifting of edge e_a to get y_1 from x_1;

(3) use a lifting of edge e_b to get x_2 from y_1;

(4) use a lifting of edge e_c to get y_2 from x_1;

(5) use a lifting of edge e_a to get x_3 from y_2.

Note that it is *not* necessary to complete the correspondences (a), (b), and (c). Doing so would require tracking 9 continuation paths, while the hypothetical run above uses only 4 paths to find a fiber.

Example 2.5 Figure 2 illustrates two partial correspondences associated to two edges e_a and e_b, both connecting two vertices v_1 and v_2 in $V(G)$. Each vertex v_i stores the array of known points Q_i, which are depicted in solid. Both

correspondences in the picture are subsets of a perfect matching, a one-to-one correspondence established by a homotopy associated to the edge.

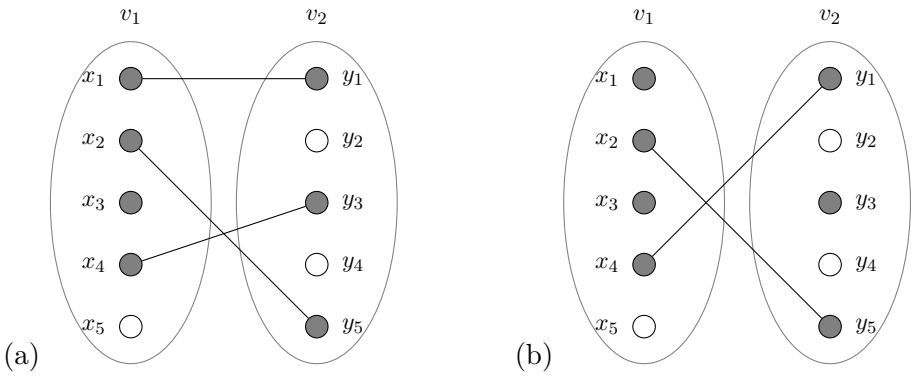

Figure 2: Two partial correspondences induced by edges e_a and e_b for the fibers of the covering map of degree $d = 5$ in Example 2.5.

Note that taking the set of start points $S_1 = \{x_3\}$ and following the homotopy $H^{(e_a)}$ from left to right is *guaranteed* to discover a new point in the second fiber. On the other hand, it is impossible to obtain new knowledge by tracking $H^{(e_a)}$ from right to left. Homotopy $H^{(e_b)}$ has a *potential* to discover new points if tracked in either direction. We can choose $S_1 = \{x_1, x_3\}$ as the start points for one direction and $S_2 = \{y_3\}$ for the other. In this scenario, following the homotopy from left to right is guaranteed to produce at least one new point, while going the other way may either deliver a new point or just augment the correspondences between the already known points. If the correspondences in (a) and (b) are completed to one-to-one correspondences of the fibers, taking the homotopy induced by the edge e_a from left to right followed by the homotopy induced by edge e_b from right to left would produce a permutation. However, the group generated by this permutation has to stabilize $\{x_2\}$, therefore, it would not act transitively on the fiber of v_1. One could also imagine a completion such that the given edges would not be sufficient to discover x_5 and y_4.

In our algorithm, we record and use correspondences; however, they are viewed as a secondary kind of knowledge. In particular, in §3.2.4 we develop heuristics driven by edge potential functions which look to maximize the number of newly discovered solutions, in other words, to extend the primary knowledge in some greedy way.

3 Algorithms and strategies

The operations listed in Section 2.3 give a great deal of freedom in the discovery of solutions. However, not all strategies for applying these operations are equally efficient. We distinguish between *static* strategies, where the graph is

fixed throughout the discovery process and *dynamic* strategies, where vertices and edges may be added.

3.1 A naive dynamic strategy

An immediate strategy for populating the fiber $\pi^{-1}(F_{p_1})$ starting from a single element is to make random monodromy loops and track the single solution around to new elements in $\pi^{-1}(F_{p_1})$. If the induced permutation on $\pi^{-1}(F_{p_1})$ is uniformly distributed, then so is the obtained solution in $\pi^{-1}(F_{p_1})$. If $\pi^{-1}(F_{p_1})$ is large, then it is likely a different solution. However, it is possible that the found solution was already known. We repeat the process, tracking a single solution at a time, until all d elements of $\pi^{-1}(F_{p_1})$ have been found. Finding the expected number of iterations to discover the entire fiber is equivalent to solving the *coupon collector's problem*. The number equals $d\ell(d)$ where $\ell(d) := \frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{d}$. The values of $\ell(d)$ can be regarded as lower and upper sums for two integrals of the function $x \mapsto x^{-1}$, leading to the bounds $\ln(d+1) \leq \ell(d) \leq \ln(d) + 1$. An alternative strategy is to take all known points along the edge simultaneously. This has the advantage that two homotopy paths will not lead to the discovery of the same solution, and therefore chances of discovering new points of $\pi^{-1}(F_{p_1})$ are better.

We remark that the existing implementations of numerical irreducible decomposition in Bertini [2], PHCpack [32], and `NumericalAlgebraicGeometry` for Macaulay2 [20] that use monodromy are driven by a version of the naive dynamic strategy.

3.2 Static graph strategies

It turns out to be an advantage to reuse the edges of the graph. In a static strategy the graph is fixed and we discover solutions according to the following algorithm.

Algorithm 3.1 (Static graph strategy) *Let the base space be given by a map $\varphi : p \mapsto F_p$.*
$(j, Q_j) =$ `monodromySolve`$(G, Q', \texttt{stop})$
Input:

- *A graph G with vertices decorated with p_i's and edges decorated with pairs $(\gamma_1, \gamma_2) \in \mathbb{C}^2$.*
- *Subsets $Q'_i \subset \pi^{-1}(\varphi(p_i))$ for $i \in 1, \ldots, |V(G)|$, not all empty.*
- *A stopping criterion* `stop`*.*

Output: *A vertex j in G and a subset Q_j of the fiber $\pi^{-1}(F_{p_j})$ with the property that Q_j cannot be extended by tracking homotopy paths represented by G.*

$Q_i := Q'_i$ for $i \in 1, \ldots, |V(G)|$.
while *there exists an edge $e = (j, k)$ in G such that Q_j has points not yet tracked with $H^{(e)}$* ***do***
Choose such an edge $e = (j, k)$.

Let $S \subset Q_j$ be a nonempty subset of the set of points not yet tracked with $H^{(e)}$.
Track the points S with $H^{(e)}$ to obtain elements $T \subset \pi^{-1}(\varphi(p_k)) \setminus Q_k$.
Let $Q_k := Q_k \cup T$.
***if** the criterion* `stop` *is satisfied (e.g., $|Q_k|$ equals a known solution count)*
then
return (k, Q_k)
end if
end while
Choose some vertex j and return (j, Q_j).

The algorithm can be specialized in several ways. We may

- choose the graph G,
- specify a stopping criterion `stop`,
- choose a strategy for picking the edge $e = (j, k)$.

We address the first choice in §3.2.1 by listing several graph layouts that can be used. Stopping criteria are discussed in §3.2.2 and §3.2.3, while strategies for selecting an edge are discussed in §3.2.4.

Remark 3.2 We notice that if the stopping criterion is never satisfied, the number of paths being tracked by Algorithm 3.1 is at most $d|E(G)|$, where d is the number of solutions of a generic system.

3.2.1 Two static graph layouts

We present two graph layouts to be used for the static strategy (Figure 3).

`flower(s,t)` The graph consists of a *central node* v_0 and s additional vertices (number of petals), each connected to v_0 by t edges.

`completeGraph(s,t)` The graph has s vertices. Every pair of vertices is connected by t edges.

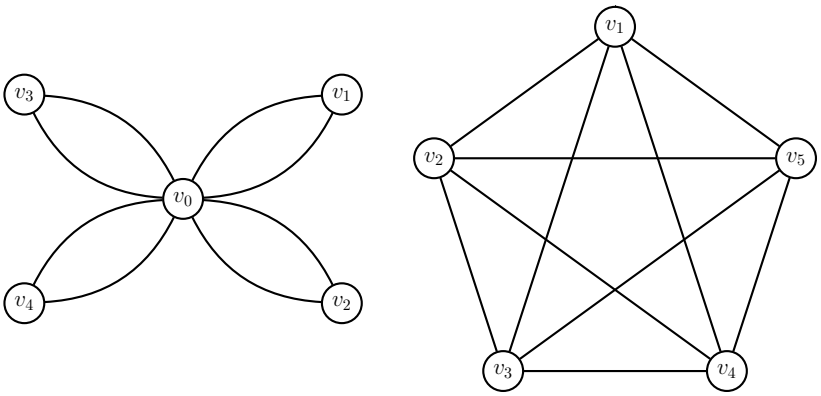

Figure 3: Graphs for the `flower(4,2)` strategy and `completeGraph(5,1)`.

3.2.2 Stopping criterion if a solution count is known

Suppose the cardinality of the fiber $\pi^{-1}(F_p)$ for a generic value of p is known. Then a natural stopping criterion for our algorithm is to terminate when the set of known solutions Q_i at any node i reaches that cardinality. In particular, for a generic sparse system with fixed monomial support we can rely on this stopping criterion due to the BKK bound [3] that can be obtained by a *mixed volume* computation.

3.2.3 Stopping criterion if no solution count is known

For a static strategy one natural stopping criterion is *saturation* of the known solution correspondences along all edges. In this case, the algorithm simply can't derive any additional information. It also makes sense to consider a heuristic stopping criterion based on *stabilization*. The algorithm terminates when no new points are discovered in a fixed number of iterations. This possibly avoids saturating correspondences unnecessarily. In particular, this could be useful if a static strategy algorithm is a part of the dynamic strategy of §3.3.

Remark 3.3 In certain cases it is possible to provide a stopping criterion using the *trace test* [29, 22]. This is particularly useful when there is an equation in the family $F_p(x) = 0$ that describes a generic hypersurface in the parameter space, e.g., an affine linear equation with indeterminate coefficients. In full generality, one could restrict the parameter space to a generic line and, hence, restrict the solution variety to a curve. Now, thinking of $F_p(x) = 0$ as a system of equations bihomogeneous in p and x, one can use the multihomogeneous trace test [22, 14].

We note that the multihomogeneous trace test complexity depends on the degree of the solution variety, which may be significantly higher than the degree d of the covering map, where the latter is the measure of complexity for the main problem in our paper. For instance, the system (7) corresponding to the reaction network in Figure 4 has 4 solutions, but an additional set of 11 points is necessary to execute the trace test. See `example-traceCRN.m2` at [9].

3.2.4 Edge selection strategy

We propose two methods for selecting the edge e in Algorithm 3.1. The default is to select an edge and direction at random. A more sophisticated method is to select an edge and a direction based on the potential of that selection to deliver new information. Let $e = v_i \xleftarrow{(\gamma_1,\gamma_2)} v_j$ be an edge considered in the direction from v_i to v_j.

`potentialLowerBound` equals the minimal number of new points guaranteed to be discovered by following a chosen homotopy using the maximal batch of starting points S_i. That is, it equals the difference between the numbers of known unmatched points $(|Q_i| - |C_e|) - (|Q_j| - |C_e|) = |Q_i| - |Q_j|$ if this difference is positive and 0 otherwise.

`potentialE` equals the expected number of new points obtained by tracking one unmatched point along e. This is the ratio $\frac{d-|Q_j|}{d-|C_e|}$ of undiscovered points among all unmatched points if $|Q_i| - |C_e| > 0$ and 0 otherwise.

Note that `potentialE` assumes we know the cardinality of the fiber, while `potentialLowerBound` does not depend on that piece of information.

There is a lot of freedom in choosing potentials in our algorithmic framework. The two above potentials are natural "greedy" choices that are easy to describe and implement. It is evident from our experiments (Tables 2 and 3) that they may order edges differently resulting in varying performance.

3.3 An incremental dynamic graph strategy

Consider a dynamic strategy that amounts to augmenting the graph once one of the above "static" criteria terminates Algorithm 3.1 for the current graph. One simple way to design a dynamic stopping criterion, we may call it *dynamic stabilization*, is to decide how augmentation is done and fix the number of augmentation steps that the algorithm is allowed to make without increasing the solution count. A dynamic strategy which is simple to implement is one that starts with a small graph G and augments it if necessary.

Algorithm 3.4 (Dynamic graph strategy) *Let us make the same assumptions as in Algorithm 3.1.*
$(j, Q_j) =$ `dynamicMonodromySolve`$(G, x_1,$ `stop`, `augment`$)$
Input:

- *A graph G as in Algorithm 3.1.*
- *One seed solution $x_1 \in \pi^{-1}(\varphi(p_1))$.*
- *A stopping criterion* `stop`.
- *An augmenting procedure* `augment`.

Output: *A vertex j in G and a subset Q_j of the fiber $\pi^{-1}(F_{p_j})$.*

$Q_1 := \{x_1\}$ *and* $Q_i = \emptyset$ *for* $i \in 2, \ldots, |V(G)|$.
loop
  $(j, Q_j) =$ `monodromySolve`$(G, Q,$ `stop`$)$ {*here Q_i are modified in-place and passed to the next iteration*}
  if `stop` *(i.e., stopping criterion is satisfied)* ***then***
    return (j, Q_j)
  end if
  $G :=$ `augment`(G)
end loop

We emphasize that the criteria described in this subsection and some of § 3.2.3 are *heuristic* and there is a lot of freedom in designing such. In §6.2 we successfully experiment using a static stabilization criterion with some examples, for which the solution count is generally not known.

4 Statistical analysis

The directed cycles in the graph G starting and ending at a vertex v_1 give elements of the fundamental group $\pi_1(B \setminus D)$, which correspond to the elements of the monodromy subgroup $M(G)$ of the monodromy group $M(\pi_1(B \setminus D))$. The latter is a subgroup of S_d, where $d = |\pi^{-1}(F_{p_1})|$. For example, if $G = \texttt{completeGraph}(2, j+1)$, then the j cycles produced by edges e_1 and e_2, e_2 and e_3,..., e_j and e_{j+1} suffice to generate $M(G)$. The minimal number j of cycles necessary to generate $M(G)$ in the general case is $\beta_1(G)$, the first Betti number of G as a topological space.

For the purpose of simplifying statistical analysis, we assume that picking a random decorated graph G with $j = \beta_1(G)$ induces uniformly and independently distributed permutations $\sigma_1, \ldots, \sigma_j \in S_d$, where S_d is the symmetric group acting on the fiber $\pi^{-1}(F_{p_1})$. It seems it would be hard in practice to achieve uniformity even when the mondromy is symmetric.

4.1 The probability of a transitive action

Suppose the number of solutions d is known and $\texttt{stop}(d)$ denotes the corresponding stopping criterion. Our aim is to analyze the probability of Algorithm 3.1 producing the full solution set or, equivalently, the probability of

$$\texttt{dynamicMonodromySolve}(G, x_1, \texttt{stop(d)}, \texttt{augment})$$

stopping after at most j iterations, assuming that $\beta_1(G) = j$ at the j-th iteration. This equals the probability of $\langle \sigma_1, \ldots, \sigma_j \rangle$ acting transitively i.e. $\Pr[X_d \leq j]$ where X_d is the random variable

$$X_d = \inf\{i \in \mathbb{N} \mid \langle \sigma_1, \ldots, \sigma_i \rangle \text{ is transitive}\}.$$

When $d > 1$ we have $\Pr[X_d = 0] = 0$, while $\Pr[X_d = 1]$ is proportional to the number of d-cycles in the monodromy group. When the monodromy group is full symmetric, we can compute and give asymptotic estimates for the distribution of X_d. The following theorem is a generalization of a result by Dixon, regarding the case $j = 2$. The proof we give in §4.2 follows the strategy of [8].

Theorem 4.1 *For $j \geq 2$, $\Pr[X_d \leq j] = 1 - d^{1-j} + R_j(d)$, where the error term R_j satisfies $|R_j(d)| = O(d^{-j})$.*

Remark 4.2 As a corollary, one can deduce that the expected value of X_d is asymptotically finite and $\mathrm{E}[X_d] \to 2$ as $d \to \infty$. The numerical approximations in Table 1 show that $\mathrm{E}[X_d] \leq 2.1033$ for all d. Moreover, the proof in §4.2 implies that $|R_j(d)| < C\left(\frac{d}{2}\right)^{-j}$ with the constant C not depending on j. Therefore, $\Pr[X_d > j]$ decays exponentially with j.

Under the idealistic assumption that new cycles in the graph lead to independently and uniformly distrubuted permutations of the fiber Q_1, the expected Betti number needed for completion in Algorithm 3.4 is at most 2.1033. If we assume that `augment` increases the Betti number by one by adding at most a fixed number of edges, then *the expected number of tracked paths is linear in d.*

Remark 4.3 We point out that Babai [1] proved Dixon's conjecture stating that the subgroup of S_d generated by two random permutations is S_d or A_d with probability $1 - d^{-1} + O(d^{-2})$. This shows that other subgroups are rare. However, it is easy to construct families with a transitive monodromy group that is neither full symmetric nor alternating. Take, for example, $x_1^2 - c_1 = x_2^2 - c_2 = 0$ with irreducible solution variety and 4 solutions for generic choices of c_1 and c_2. Tracking two solutions with the same x_1 coordinate, as c_1 and c_2 vary, the moving points on the tracked paths will continue to have equal projections to x_1. The monodromy group is $\mathbb{Z}_2 \times \mathbb{Z}_2$.

We reiterate that generators for the monodromy group are seldomly known *a priori*. Computing them is likely to be prohibitively expensive, and the probability distribution with which our algorithm picks elements of the monodromy group is unknown, as it is prohibitively hard to analyze.

4.2 Proof of a generalization of Dixon's theorem

Fix any integer $j \geq 2$. We wish to prove Theorem 4.1 by estimating the quantity

$$t_d = \Pr\left[\langle \sigma_1, \sigma_2, \ldots, \sigma_j \rangle \text{ is transitive}\right],$$

where $\sigma_1, \ldots, \sigma_j$ are independent and uniformly distributed on S_d. Suppose we partition the set $\{1, 2, \ldots, d\}$ in such a way that there are k_i classes of size i for each $1 \leq i \leq d$. All such partitioning schemes are indexed by the set

$$K_d = \left\{ \vec{k} \in \mathbb{N}^d \mid \sum i\, k_i = d \right\}.$$

The number of partitions corresponding to each $\vec{k} \in K_d$ is $d!/(\prod_{i=1}^d (i!)^{k_i} \cdot k_i!)$. For each $\vec{k} \in K_d$, the partition given by the orbits of $\langle \sigma_1, \ldots, \sigma_j \rangle$ is $\vec{k}$-indexed precisely when this group acts transitively on all classes of some partition associated to $\vec{k}$. The number of tuples in S_i^j with coordinates generating a group acting transitively on $\{1, \ldots, i\}$ is $t_i\,(i!)^j$. Thus, we may count the set $\underbrace{S_d \times \cdots \times S_d}_{j}$ as

$$\begin{aligned}
(d!)^j &= \sum_{\vec{k} \in K_d} \frac{d!}{\prod_{i=1}^d (i!)^{k_i} \cdot k_i!} \cdot \prod_{i=1}^d \left(t_i\,(i!)^j\right)^{k_i} \\
&= d! \cdot \sum_{\vec{k} \in K_d} \prod_{i=1}^d \frac{\left(t_i\,(i!)^{j-1}\right)^{k_i}}{k_i!}.
\end{aligned}$$

Let $\widehat{F}$ denote the generating function of the sequence $F(d) = (d!)^{j-1}$. Note the formal identity

$$\exp\left(\sum_{i=0}^{\infty} y_i x^i\right) = \sum_{d=0}^{\infty} x^d \sum_{\vec{k} \in K_d} \prod_{i=1}^d \frac{y_i^{k_i}}{k_i!},$$

d	$j=2$	$j=3$	$j=4$	$\mathrm{E}[X_d]$
1	1	1	1	0
2	0.75	0.875	0.9375	2
3	0.72222222	0.89814815	0.96450617	2.10000000
4	0.73958333	0.93012153	0.98262080	2.10329381
5	0.76833333	0.95334722	0.99115752	2.08926525
10	0.88180398	0.98954768	0.99898972	2.02976996
20	0.94674288	0.99747856	0.99987487	2.00591026
30	0.96536852	0.99888488	0.99996295	2.00245160

Table 1: Numerical approximations of t_d — the probability of the j random permutations acting transitively on a fiber of size d for $j = 2, 3, 4$. After computing these values for larger j a numerical approximation of $\mathrm{E}[X_d]$ is extracted.

which follows by letting $g(x)$ denote the right hand side as a formal power series in x, $f(x) = \sum y_i x^i$, and noting the equivalent form $f'g = g'$ with $f(0) = y_0$. We have

$$\begin{aligned}
\sum_{d=1}^{\infty} d \cdot (d!)^{j-1}\, x^{d-1} &= \frac{d}{dx}\, \widehat{F}(x) \\
&= \frac{d}{dx}\, \exp\left(\sum_{i=1}^{\infty} t_i\, (i!)^{j-1} x^i \right) \\
&= \Big(\sum_{d=0}^{\infty} (d!)^{j-1}\, x^d \Big) \cdot \sum_{i=1}^{\infty} i \cdot t_i\, (i!)^{j-1} x^{i-1} \\
&= \sum_{d'=1}^{\infty} x^{d'-1} \Big(\sum_{i=1}^{d'} i \cdot t_i \left(i! \cdot (d'-i)! \right)^{j-1} \Big)
\end{aligned}$$

where the first equation follows by formal differentiation of the power series $\widehat{F}$, the second from the two identities above with properly substituted values for y_i, the third by applying the chain rule and the definition of $\widehat{F}$, and the fourth by rearranging terms by index substitution $d' = i + d$. Upon equating coefficients of x^{d-1} for $d = 1, \ldots$ we obtain

$$d = \sum_{i=1}^{d} \binom{d}{i}^{1-j} i\, t_i. \tag{1}$$

Remark 4.4 Equation (1) gives a list of linear equations in the probabilities $t_1, t_2, \ldots$ allowing us to successively determine these values by backward substitution. In Table 1 we list some solutions for $j = 2, 3, 4$.

To complete the proof of Theorem 4.1, we introduce, as in Dixon's proof, the auxiliary quantities

$$r_d = d\,(1 - t_d) \quad \text{and} \quad c_d = \sum_{i=1}^{d-1} \binom{d}{i}^{1-j} i.$$

Noting that $\binom{d}{i}^{1-j} i + \binom{d}{d-i}^{1-j}(d-i) = \frac{d}{2}\left(\binom{d}{i}^{1-j} + \binom{d}{d-i}^{1-j}\right)$, we have

$$\begin{aligned}\frac{c_d}{d} = \frac{1}{2}\cdot\sum_{i=1}^{d-1}\binom{d}{i}^{1-j} &= d^{1-j} + \left[\binom{d}{2}^{1-j} + \frac{1}{2}\sum_{i=3}^{d-3}\binom{d}{i}^{1-j}\right] \\ &\le d^{1-j} + \left[\binom{d}{2}^{1-j} + \frac{1}{2}(d-5)\binom{d}{3}^{1-j}\right] \qquad (2)\end{aligned}$$

From $j \ge 2$, it follows that the bracketed expression in (2) is $O(d^{-j})$. Using (1), $t_i = 1 - \frac{r_i}{i} \le 1 - \frac{r_i}{d}$ and the definition of c_d, we may bound r_d:

$$\begin{aligned} r_d = d(1-t_d) = -dt_d + d &= -dt_d + \sum_{i=1}^{d}\binom{d}{i}^{1-j} i\cdot t_i \qquad (3)\\ = \sum_{i=1}^{d-1}\binom{d}{i}^{1-j} i\cdot t_i &\le \sum_{i=1}^{d-1}\binom{d}{i}^{1-j} i\cdot(1-\frac{r_i}{d}) \\ = \sum_{i=1}^{d-1}\binom{d}{i}^{1-j} i - \frac{1}{d}\sum_{i=1}^{d-1} r_i\binom{d}{i}^{1-j} &= c_d - \frac{1}{d}\sum_{i=1}^{d-1} r_i\binom{d}{i}^{1-j} \le c_d. \qquad (4)\end{aligned}$$

To bound the error term $R_j(d) := t_d - (1-d^{1-j})$, we consider first the case where its sign is positive. Expanding $t_d = 1 - \frac{r_d}{d}$ using (3) above and $i\,t_i = i - r_i$,

$$\begin{aligned} t_d - (1-d^{1-j}) &= 1 - \sum_{i=1}^{d-1}\binom{d}{i}^{1-j}\frac{it_i}{d} - 1 + d^{1-j} = d^{1-j} - \sum_{i=1}^{d-1}\binom{d}{i}^{1-j}\frac{it_i}{d} \\ &= d^{1-j} - \sum_{i=1}^{d-1}\binom{d}{i}^{1-j}\frac{i}{d} + \sum_{i=1}^{d-1}\binom{d}{i}^{1-j}\frac{r_i}{d} \\ &= \underbrace{d^{1-j} - d^{1-j}\left(\frac{1}{d} + \frac{d-1}{d}\right)}_{=0} - \sum_{i=2}^{d-2}\binom{d}{i}^{1-j}\frac{i}{d} + \sum_{i=1}^{d-1}\binom{d}{i}^{1-j}\frac{r_i}{d}, \end{aligned}$$

and we may focus on the last summation to get, for $d \ge 2$,

$$\begin{aligned} t_d - (1-d^{1-j}) &\le d^{1-j}\frac{r_{d-1}}{d-1} + \sum_{i=2}^{d-2}\binom{d}{i}^{1-j}\frac{r_i}{d} && \text{(note } r_1 = 0\text{)} \\ &\le d^{1-j}\frac{c_{d-1}}{d-1} + \sum_{i=2}^{d-2}\binom{d}{2}^{1-j}\frac{c_i}{d} && \text{(by (4))} \\ &\le d^{1-j}\frac{c_{d-1}}{d-1} + \left(\frac{d^2}{4}\right)^{1-j}\sum_{i=2}^{d-2}\frac{c_i}{i} \\ &= O(d^{2-2j}) + O(d^{(2-2j)+(2-j)}) \\ &= O(d^{-j}) && \text{(since } j \ge 2.\text{)} \end{aligned}$$

The case where $R_j(d) \leq 0$ may be handled similarly, using $t_d = 1 - \frac{r_d}{d}$, $r_d \leq c_d$ and what we know about the content of the bracket in (2).

$$-t_d + (1 - d^{1-j}) = -(1 - \frac{r_d}{d}) + (1 - d^{1-j}) = \frac{r_d}{d} - d^{1-j} \leq \frac{c_d}{d} - d^{1-j}$$
$$\leq \left[\binom{d}{2}^{1-j} + \frac{1}{2}(d-5)\binom{d}{3}^{1-j} \right] = O(d^{-j}).$$

5 Implementation

5.1 MonodromySolver: a Macaulay2 package

We implemented the package `MonodromySolver` in Macaulay2 [11] using the functionality of the package `NumericalAlgebraicGeometry` [20]. The source code and examples used in the experiments in the next section are available at [9].

The main function `monodromySolve` realizes Algorithms 3.1 and 3.4, see the documentation for details and many options. The tracking of homotopy paths in our experiments is performed with the native routines implemented in the kernel of Macaulay2, however, `NumericalAlgebraicGeometry` provides an ability to outsource this core task to an alternative tracker (PHCpack or Bertini). Main auxiliary functions—`createSeedPair`, `sparseSystemFamily`, `sparseMonodromySolve`, and `solveSystemFamily`—are there to streamline user's experience. The last two are blackbox routines that don't assume any knowledge of the framework described in this paper.

The overhead of managing the data structures is supposed to be negligible compared to the cost of tracking paths. However, since our implementation uses the interpreted language of Macaulay2 for other tasks, this overhead could be sizable (up to 10% for large examples in §6). Nevertheless, most of our experiments are focused on measuring the *number of tracked paths* as a proxy for computational complexity.

Remark 5.1 This paper's discussion focuses on linear parametric systems with a nonempty dominant component. However, the implementation works for other cases where our framework can be applied.

For instance, if the system is linear in parameters but has no dominant component, there may still be a unique "component of interest" with a straightforward way to produce a seed pair. This is so, for instance, in the problem of finding the degree of the variety $SO(n)$, which we use in Table 7. The point x is restricted to $SO(n)$, the special orthogonal group, which is irreducible as a variety. This results in a unique "component of interest" in the solution variety, the one that projects onto $SO(n)$, see [5] for details.

In a yet more general case of a system that is nonlinear in parameters, it is still possible to use our software. We outline the theoretical issues one would need to consider in §7.2.

5.2 Solution count

The BKK bound, computed via mixed volume, is used as a solution count in the examples of sparse systems in §6.1.1 and §6.1.2. In the latter we compute mixed volume via a closed formula that involves permanents, while the former relies on general algorithms implemented in several software packages. Our current implementation uses PHCpack [32], which incorporates the routines of MixedVol further developed in Hom4PS-2 [19]. Other alternatives are pss5 [25] and Gfanlib [18]. While some implementations are randomized, the latter uses symbolic perturbations to achieve exactness. The computation of the mixed volume is *not* a bottleneck in our algorithm. The time spent in that preprocessing stage is negligible compared to the rest of the computation.

5.3 Certification

The reader should realize that the numerical homotopy continuation we use is driven partly by heuristics. As a post-processing step, we can certify (i.e. formally prove) the completeness and correctness of the solution set to a polynomial system computed with our main method. This is possible in the scenario when

- the parameteric system is square,
- all solutions are regular (the Jacobian of the system is invertible), and
- the solution count is known.

We can use Smale's α-theory [4, §8] to certify an approximation to a regular solution of a square system. In a Macaulay2 package `AlphaTest`, we implement a numerical version of an *α-test* after finding an approximate solution to certify that our solution is an *approximate zero* in a rigorous sense.

One of the main functions of `AlphaTest` is `certifySolutions`, which determines whether the given solution is an approximate zero of the given polynomial system. It also produces an upper bound on the distance from the approximation to the exact solution to which it is associated.

See `paper-examples/example-NashCertify.m2` at [9], which is an example of an α-test application to the solutions of a problem described in §6.1.2.

In the implementation of certification, all arithmetic and linear algebra operations are done over the field of Gaussian rationals, $\mathbb{Q}[\mathbf{i}]/(\mathbf{i}^2+1)$. To use this certification method we first convert the coefficients of the system to Gaussian rationals, then perform certification numerically.

See [16] for a standalone software package `alphaCertified` and detailed implementation notes.

6 Experiments

In this section we first report on experiments with our implementation and various examples in §6.1 and §6.2. We then investigate the completion rate of Algorithm 3.1 in §6.3. Finally we compare against other software in §6.4.

6.1 Sparse polynomial systems

The example families in this subsection have the property that the support of the equations is fixed while coefficients can vary freely, as long as they are generic. We run the static graph strategy Algorithm 3.1 on these examples. Our timings do not include the α-test, which was only applied in §6.1.2.

6.1.1 Cyclic roots

The cyclic n-roots polynomial system is

$$\begin{cases} i = 1, 2, 3, 4, \ldots, n-1 : \displaystyle\sum_{j=0}^{n-1} \prod_{k=j}^{j+i-1} x_{k \bmod n} = 0 \\ x_0 x_1 x_2 \cdots x_{n-1} - 1 = 0. \end{cases} \tag{3}$$

This system is commonly used to benchmark polynomial system solvers. We will study the modified system with randomized coefficients and seek solutions in $(\mathbb{C} \setminus \{0\})^n$. Therefore the solution count can be computed as the mixed volume of the Newton polytopes of the left hand sides, providing a natural stopping criterion discussed in §3.2.2. This bound is 924 for cyclic-7.

Tables 2 and 3 contain averages of experimental data from running twenty trials of Algorithm 3.1 on cyclic-7. The main measurement reported is the average number of paths tracked, as the unit of work for our algorithm is tracking a single homotopy path. The experiments were performed with 10 different graph layouts and 3 edge selection strategies.

(#vertices-1, edge multiplicity)	(3,2)	(4,2)	(5,2)	(3,3)	(4,3)
$\lvert E(G)\rvert$	6	8	10	9	12
$\beta_1(G)$	3	4	5	6	8
$\lvert E(G)\rvert \cdot 924$	5544	7392	9240	8316	11088
completion rate	100%	100%	100%	100%	100%
Random Edge	5119	6341	7544	6100	7067
`potentialLowerBound`	5252	6738	8086	6242	7886
`potentialE`	4551	5626	6355	4698	5674

Table 2: Cyclic-7 experimental results for the `flower` strategy.

With respect to number of paths tracked, we see that it is an advantage to keep the Betti number high and edge number low. The same experiment with the naive dynamic strategy of §3.1 needs on average 7521 paths. This shows that already on examples of this size ($d \approx 1000$ solutions) our new method outperforms the naive one. Since the former is argued to be roughly linear in §4 while the latter is roughly $d \ln d$ according to §3.1, the advantage grows with d.

Remark 6.1 Computing the expected success rates ($> 99\%$) using Remark 4.4, we conclude that the resulting permutations do not conform to the model of picking uniformly from S_{924}. The completion rate depends on the choice of

(#vertices, edge multiplicity)	(2,3)	(2,4)	(2,5)	(3,2)	(4,1)
$\|E(G)\|$	3	4	5	6	6
$\beta_1(G)$	2	3	4	4	3
$\|E(G)\| \cdot 924$	2772	3698	4620	5544	5544
completion rate	65%	80%	90%	100%	100%
Random Edge	2728	3296	3947	4805	5165
`potentialLowerBound`	2727	3394	3821	4688	5140
`potentialE`	2692	2964	2957	3886	4380

Table 3: Cyclic-7 experimental results for the `completeGraph` strategy.

strategy (compare Table 2 to Table 3). Nevertheless, both in theory and in practice, the completion rate does converge to 100% rapidly as the Betti number grows.

6.1.2 Nash equilibria

Semi-mixed multihomogeneous systems arise when one is looking for all totally mixed Nash equilibria (TMNE) in game theory. A specialization of mixed volume using matrix permanents gives a concise formula for a root count for systems arising from TMNE problems [10]. We provide an overview of how such systems are constructed based on [10]. Suppose there are N players with m options each. For player $i \in \{1, \ldots, N\}$ using option $j \in \{1, \ldots, m\}$ we have the equation $P_j^{(i)} = 0$, where

$$P_j^{(i)} = \sum_{\substack{k_1,\ldots,k_{i-1},\\ k_{i+1},\ldots,k_N}} a^{(i)}_{k_1,\ldots,k_{i-1},j,k_{i+1},\ldots,k_N} p_{k_1}^{(1)} p_{k_2}^{(2)} \cdots p_{k_{i-1}}^{(i-1)} p_{k_{i+1}}^{(i+1)} \cdots p_{k_N}^{(N)}. \tag{4}$$

The parameters $a^{(i)}_{k_1,k_2,\ldots,k_N}$ are the payoff rates for player i when players $1, \ldots, i-1, i+1, \ldots, N$ are using options $k_1, \ldots, k_{i-1}, k_{i+1}, \ldots, k_N$, respectively. Here the unknowns are $p_{k_j}^{(i)}$, representing the probability that player i will use option $k_j \in \{1, \ldots, m\}$. There is one constraint on the probabilities for each player $i \in \{1, \ldots, N\}$, namely the condition that

$$p_1^{(i)} + p_2^{(i)} + \cdots + p_m^{(i)} = 1. \tag{5}$$

The system (4) consists of $N \cdot m$ equations in $N \cdot m$ unknowns. Using condition (5) reduces the number of unknowns to $N(m-1)$. Lastly, we eliminate the $P_j^{(i)}$ by constructing

$$P_1^{(i)} = P_2^{(i)},\ P_1^{(i)} = P_3^{(i)},\ \ldots\ , P_1^{(i)} = P_m^{(i)}, \quad \text{for each } i \in \{1, \ldots, N\}. \tag{6}$$

The final system is a square system of $N(m-1)$ equations in $N(m-1)$ unknowns.

For one of our examples (`paper-examples/example-Nash.m2` at [9]), we chose the generic system of this form for $N = 3$ players with $m = 3$ options for each. The result is a system of six equations in six unknowns and 81 parameters with 10 solutions. We also use this example to demonstrate that these solutions can be certified using `AlphaTest` (§5.3).

6.2 Chemical reaction networks

A family of interesting examples arises from chemical reaction network theory. A chemical reaction network considered under the laws of mass-action kinetics leads to a dynamical polynomial system, the solutions of which represent all the equilibria for the given reaction network [12, 24]. These polynomial systems are not generically sparse and we cannot easily compute their root count. In our experiments, we used the stabilization stopping criterion, terminating the algorithm after a fixed number of iterations that did not deliver new points. The default is 10 fruitless iterations.

Figure 4 gives an example of a small chemical reaction network.

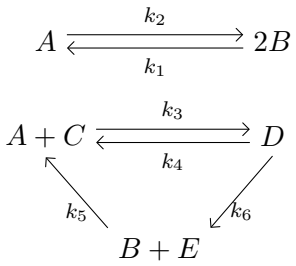

Figure 4: Chemical reaction network example.

Applying the laws of mass-action kinetics, we obtain the polynomial system (7) consisting of the steady-state and conservation equations of the reaction network. Here the k_i's represent the reaction rates, x_i's represent species concentrations, and the c_i's are parameters.

$$\begin{aligned}
\dot{x_A} &= k_1x_B^2 - k_2x_A - k_3x_Ax_C + k_4x_D + k_5x_Bx_E \\
\dot{x_B} &= 2k_1x_A - 2k_2x_B^2 + k_4x_D - k_5x_Bx_E \\
\dot{x_C} &= -k_3x_Ax_C + k_4x_D + k_5x_Bx_E \\
\dot{x_D} &= k_3x_Ax_C - (k_4 + k_6)x_D \\
\dot{x_E} &= -k_5x_Bx_E + k_6x_D \\
0 &= 2x_A + x_B - x_C + x_D - c_1 \\
0 &= -2x_A - x_B + 2x_C + x_E - c_2
\end{aligned} \tag{7}$$

Typically, systems resulting from chemical reaction networks will be overdetermined. With the current implementation one needs to either square the system or use a homotopy tracker that supports following a homotopy in a space of overdetermined systems.

Although we may obtain large systems, they typically have very low root counts compared to the sparse case. The polynomial system (7) has four solutions. A larger example is the *wnt signaling pathway* from Systems Biology [12] consisting of 19 polynomial equations with 9 solutions. All 9 solutions are obtained in less than a second with Algorithm 3.1.

6.3 Completion rate

We investigate the completion rate of Algorithm 3.1 for the Katsura family parametrized by n with fixed support and generically chosen coefficients. Tables 4 and 5 contain the percentage of successes from 500 runs with distinct random seeds. In Table 6, we show the computed expected values using Remark 4.4.

		(#vertices-1, edge multiplicity)				
n	BKK Bound	(3,2) $\beta_1 = 3$	(4,2) $\beta_1 = 4$	(5,2) $\beta_1 = 5$	(3,3) $\beta_1 = 6$	(4,3) $\beta_1 = 8$
5	12	96.4%	99.4%	99.6%	100%	99.8%
6	30	98.6%	100%	99.8%	100%	99.6%
7	54	97.6%	98.8%	99.4%	99.4%	98.4%
8	126	99.2%	99.8%	99.6%	99.8%	99.8%
9	240	98.8%	99.6%	98.4%	98.4%	98.6%
10	504	98.6%	98.8%	99.2%	99.4%	98.8%

Table 4: Katsura-$(n-1)$ for the `flower` strategy.

		(#vertices, edge multiplicity)				
n	BKK Bound	(2,3) $\beta_1 = 2$	(2,4) $\beta_1 = 3$	(2,5) $\beta_1 = 4$	(3,2) $\beta_1 = 4$	(4,1) $\beta_1 = 3$
5	12	65.6%	88.2%	95%	99.2%	98%
6	30	77.4%	95.2%	99%	99.8%	99.6%
7	54	74.4%	96.2%	99.2%	99.6%	99.8%
8	126	81.8%	97%	99.2%	100%	99.8%
9	240	85.2%	97.6%	99.4%	99%	98.2%
10	504	89.2%	98.2%	99.2%	99.4%	99%

Table 5: Katsura-$(n-1)$ for the `completeGraph` strategy.

For $\beta_1 \geq 3$ the observed success rates approach the expected values of Table 6. We note that the `flower` strategy is again closest to the estimates. We do not expect the numbers produced in experiments to match the numbers in Table 1, since the assumptions made for that statistical analysis are quite idealistic; however, both the analysis and experiments show that the probability of success approaches 100% rapidly as the number of solutions grows and the first Betti number increases.

d	$\beta_1 = 2$	$\beta_1 = 3$	$\beta_1 \geq 4$
12	90.5%	99.3%	100.0%
30	96.5%	99.9%	100.0%
54	98.1%	100.0%	100.0%
126	99.2%	100.0%	100.0%
240	99.6%	100.0%	100.0%
504	99.8%	100.0%	100.0%

Table 6: Rounded expected probability of success assuming uniform distribuition of permutations and full monodromy group.

6.4 Timings and comparison with other solvers

All timings appearing in this section are done on *one* thread and on the *same* machine. Remarks 3.2 and 4.2 show that we should expect the number of tracked paths in Algorithms 3.1 and 3.4 to be linear (with a small constant!) in the number of solutions of the system. In this section we highlight the practicality of our approach in two ways.

Firstly, the monodromy method extends our computational ability dramatically for systems where the solution count turns out to be significantly smaller than the count corresponding to a more general family, for example, BKK count for sparse systems. This means that the existing blackbox methods, whose complexity relies on a larger count, are likely to spend significantly more time in computation compared to our approach.

In Table 7, we collect timings on several challenging examples mentioned in recent literature where smaller solution counts are known, thus providing us with rigorous test cases for our heuristic stopping criterion. The first system in the table is that of the wnt signaling pathway reaction network mentioned in §6.2. The others come from the problem of computing the degree of $SO(n)$, the special orthogonal group, as a variety [5].

Below is a list of comments on the setup:

- For our implementation we chose small graphs with $\beta_1 \leq 4$ and the random edge selection strategy. The stopping criterion is "stabilization" as discussed in §3.2.3.

- While the blackbox solver of PHCpack ultimately performs polyhedral homotopy continuation, Bertini relies by default on an equation-by-equation technique dubbed *regeneration* (see [2]). The latter may be faster than the former in certain cases, which this series of examples show.

Secondly, when the solution count is given by the BKK bound our method is a viable alternative to polyhedral homotopy solvers, since the number of paths we track is linear in the number of solutions. The timings on a few large benchmark problems of our current implementation and several other software packages are in Table 8. Our goal in the rest of this section is to show that our running times are in the same ballpark as polyhedral homotopies.

Below is a list of comments on the setup:

problem	wnt	$SO(4)$	$SO(5)$	$SO(6)$	$SO(7)$
count	9	40	384	4768	111616
MonodromySolver	0.52	4	23	528	42791
Bertini	42	81	10605	out of memory	
PHCpack	862	103	> one day		

Table 7: Examples with solution count smaller than BKK bound (timings in seconds).

- For our implementation we chose two small graphs and default (random edge selection) strategy.
- For PHCpack there is a way to launch a mixed volume computation with the option of creating a system with the same support and random coefficients together with its solutions. This is the option we are using; the blackbox computation takes a little longer.
- HOM4PS2 [19] is not open source unlike all other software mentioned here. (We use HOM4PS2 stock examples for all systems and call its blackbox polyhedral homotopies solver.) HOM4PS2 may use just-in-time compilation of straight-line programs used for evaluation, which speeds up computations considerably. (PHCpack does not use this technique; neither does our software, but our preliminary experiments in Macaulay2 show a potential for a 10- to 20-fold speed up over our currently reported timings.)

problem	cyclic-10	cyclic-11	noon-10
BKK bound	35940	184756	59029
`completeGraph`(2,3)	610	7747	failed
	(107820 paths)	(540155 paths)	(59001 solutions)
`completeGraph`(2,4)	740	8450	935
	(129910 paths)	(737432 paths)	(236051 paths)
PHCpack	538	4256	751
HOM4PS2	62	410	120

Table 8: Software timings on large examples (in seconds).

Remark 6.2 For large examples, assuming the probabilistic model leading to Theorem 4.1 and Remark 4.2, the probability of success should be extremely close to 100% even for a random graph with $\beta_1 = 2$. The run of noon-10, which is an example of neural network model from [27], demostrates an unlikely but possible failure for $\beta_1 = 2$ followed by success at $\beta_1 = 3$.

On the examples in Table 8, we also ran the blackbox solvers of Bertini and NumericalAlgebraicGeometry [20], which uses the total-degree homotopy. Both were able to finish noon-10 with timings similar to the table, but all other problems took longer than a day. This is expected, as the BKK bound of noon-10 is only slightly sharper than the Bézout bound.

7 Generalizations

The goal of this article and our implementation is to demonstrate the power of the main algorithm and analyze it in the simplest terms possible. This section collects possible modifications, extensions, and ideas for the future that we purposefully excluded from the main scope of this paper. One topic for future development is the stopping criterion using the trace test as discussed in Remark 3.3. Some of the modifications of our algorithm are relatively straightforward (e.g., §7.4); however, their careful implementation is not. Accompanied with careful analysis, this should be carried out elsewhere.

7.1 Numerical failures

One advantage of our approach is that it can tolerate numerical failures of the underlying homotopy tracker. Such failures are bound to happen as the size of problems increases and the probability of a path going too close to the discriminant gets higher. The canonical remedy is increasing computational precision either uniformly or adaptively. However, this tends to result in much higher computational costs.

When the tracking of a homotopy path in Algorithm 3.1 fails, one could simply record that the starting point of that path corresponds to a failure. The algorithm then avoids retracking this path, while its target solution still has high chances to be reached by some path along a different edge. In fact, we already implemented this simple failure resistant mechanism and it successfully tolerates a few failures that arise in some runs for large test examples in §6.4. What has yet to be done is the statistical analysis in the presense of failures, under some suitable assumptions, and modifications of heuristic strategies, e.g., edge potential functions may penalize failures.

7.2 Nonlinear families

Let us reiterate that we choose to restrict ourselves to families of polynomial systems F_p that are affine linear in parameters for the sake of simplicity of exposition. Also, as pointed out in Remark 2.3, if a family is nonlinear in parameters, then the ability to replace systems with their scalar multiples is lost, together with an inexpensive way to produce new homotopies between two nodes in our graph. If a family is nonlinear, the ingredients in the algorithm that should change are the following.

- There could be only one edge between two nodes, since γ_1 and γ_2 have to be equal to 1 (see the discussion above).
- The initial seed, a parameter-solution pair (p_0, x_0), can not be produced in the same simple manner. One can still pick a solution x_0 at random, but the system $F_p(x_0) = 0$ will not be linear in parameters p. One has to find generic points on *all* solution components of this system and use them as seeds in order to eventually capture all solutions of a generic system.

- Since the dominant component of the solution variety (cf. Remark 2.1) is not necessarily irreducible anymore, the relevance of our statistical analysis based on modelling the monodromy group as a symmetric group is further reduced.

7.3 Arbitrary parameter varieties

We assumed that the parameter space is a complex affine space. In fact, one can drop this assumption and consider the family given by a *rational* map ϕ from an irreducible variety of parameters P to the space of polynomial systems. Our algorithm would still work as long as given two points in P one has a way to construct a parametrization of a curve containing these points.

Note that even in the case of the whole affine space, one doesn't need to stick linear segment homotopies. Should some family of explicitly constructible curves present a faster way to generate the monodromy group, the algorithm could associate edges in the graph to segments of those curves.

7.4 Monodromy/Galois group computation

The classical article in enumerative geometry [13] points out that a Galois group can be defined as the monodromy group of a covering map. Various interesting theoretical questions related to Galois groups of incidence varieties (called solution varieties here) can be attacked by a monodromy group computation, see [21, 15].

Suppose our algorithm is set to stop when the correspondences are saturated. That is, when for every edge $e = v_i \leftrightarrow v_j$ there is a bijection of full solution sets for the nodes v_i and v_j. Assuming there are no failures discussed in §7.1, one can construct a subgroup of the monodromy group induced by the fundamental group of the graph using complete correspondences between the points of the fibers along every edge by simply collecting the permutations corresponding to the simple cycles. This modification gives a heuristic algorithm to compute Galois groups. In certain scenarios (e.g., [21]) the Galois group is the largest one possible, the symmetric group, which provides an obvious verification procedure. In general, we know no good way (apart from computing the discriminant locus and, therefore, the fundamental group; which is an option only for toy problems) to design a stopping criterion that guarantees that the subgroup is actually the entire Galois group.

7.5 Parallelization

Homotopy continuation is *embarrassingly parallel*, since homotopy continuation paths can be tracked independently. A naive parallelization strategy for our approach is to pick an edge and direction according to §3.2.4, one point in the source vertex that has no correspondence along that edge, and task an idling processor with tracking the path emanating from that point. This would likely suffice to achieve linear speedups asymptotically as the number of processors is fixed while the number of solutions grows.

However, firstly, in various stages of computation the number of eligible paths may drop below the number of available idling processors. This phenomena may happen not only close to the beginning—it starts with one point—and the end of computation, but also in the intermediate stages. Secondly, the tasks as described are not completely independent, since the outcome of a task affects the procedure of selection of the next task. The efficiency of a naive parallelization may also decrease due to the fact that there is a non-zero probability for two paths with the same vertex as a target to produce the same point.

An asynchronous approach of [23] demonstrates the nontriviality of the problem of parallelization in a similar setting. One should look carefully at how to modify potentials in §3.2.4 in case several tracking tasks that are "in progress". This may be even less trivial if numerical failures discussed in §7.1 are taken into account.

7.6 Hybrid approaches

Our algorithm has yet another flexible feature. Suppose one has a method to generate partial solution sets for a given value of parameters. Then that method can be combined with our method in several ways. One way is through preprocessing by initializing Algorithm 3.1 with partial solution sets Q_i' computed with that method. An interesting combination to explore would be using polyhedral homotopies to populate just one set of known points, e.g. Q_1'. One could discard paths that are poorly conditioned, thus producing a solution set that is potentially incomplete but is computed very fast. Another way would be to combine any solver that produces random points in the fibers *in parallel* with the procedure that discovers new points in Algorithm 3.1. This has a potential to produce a faster and more robust blackbox solver by employing the best qualities of all methods involved.

References

[1] László Babai. The probability of generating the symmetric group. *Journal of Combinatorial Theory, Series A*, 52(1):148–153, 1989.

[2] Daniel J. Bates, Jonathan D. Hauenstein, Andrew J. Sommese, and Charles W. Wampler. *Numerically solving polynomial systems with Bertini*, volume 25. SIAM, 2013.

[3] D. N. Bernstein. The number of roots of a system of equations. *Funkcional. Anal. i Priložen.*, 9(3):1–4, 1975.

[4] Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale. *Complexity and real computation*. Springer-Verlag, New York, 1998.

[5] Madeline Brandt, DJ Bruce, Taylor Brysiewicz, Robert Krone, and Elina Robeva. The degree of SO(n). *arXiv preprint arXiv:1701.03200*, 2017.

[6] Justin Chen and Joe Kileel. NumericalImplicitization for Macaulay2. Preprint, 2016.

[7] Abraham Martin del Campo and Jose Israel Rodriguez. Critical points via monodromy and local methods. *arXiv preprint arXiv:1503.01662*, 2015.

[8] John D. Dixon. The probability of generating the symmetric group. *Mathematische Zeitschrift*, 110(3):199–205, 1969.

[9] Timothy Duff, Cvetelina Hill, Anders Jensen, Kisun Lee, Anton Leykin, and Jeffrey Sommars. MonodromySolver: a Macaulay2 package for solving polynomial systems via homotopy continuation and monodromy. Available at http://people.math.gatech.edu/∼aleykin3/MonodromySolver.

[10] Ioannis Z. Emiris and Raimundas Vidunas. Root counts of semi-mixed systems, and an application to counting nash equilibria. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 154–161, New York, NY, USA, 2014. ACM.

[11] Daniel R. Grayson and Michael E. Stillman. Macaulay2, a software system for research in algebraic geometry. Available at http://www.math.uiuc.edu/Macaulay2/.

[12] E. Gross, H. A. Harrington, Z. Rosen, and B. Sturmfels. Algebraic Systems Biology: A Case Study for the Wnt Pathway. *Bulletin of Mathematical Biology*, 78(1):21–51, 2016.

[13] Joe Harris. Galois groups of enumerative problems. *Duke Math. J.*, 46(4):685–724, 1979.

[14] Jonathan D Hauenstein and Jose Israel Rodriguez. Numerical irreducible decomposition of multiprojective varieties. *arXiv preprint arXiv:1507.07069*, 2015.

[15] Jonathan D. Hauenstein, Jose Israel Rodriguez, and Frank Sottile. Numerical computation of galois groups. *arXiv preprint arXiv:1605.07806*, 2016.

[16] Jonathan D. Hauenstein and Frank Sottile. Algorithm 921: alphaCertified: Certifying solutions to polynomial systems. *ACM Trans. Math. Softw.*, 38(4):28:1–28:20, August 2012.

[17] Birkett Huber and Bernd Sturmfels. A polyhedral method for solving sparse polynomial systems. *Math. Comp.*, 64(212):1541–1555, 1995.

[18] Anders Nedergaard Jensen. An implementation of exact mixed volume computation. In *Mathematical Software - ICMS 2016 - 5th International Conference, Berlin, Germany, July 11-14, 2016, Proceedings*, pages 198–205, 2016.

[19] Tsung-Lin Lee, Tien-Yien Li, and Chih-Hsiung Tsai. HOM4PS-2.0: a software package for solving polynomial systems by the polyhedral homotopy continuation method. *Computing*, 83(2-3):109–133, 2008.

[20] Anton Leykin. Numerical algebraic geometry. *The Journal of Software for Algebra and Geometry*, 3:5–10, 2011.

[21] Anton Leykin and Frank Sottile. Galois groups of Schubert problems via homotopy computation. *Math. Comp.*, 78(267):1749–1765, 2009.

[22] Anton Leykin and Frank Sottile. Trace test. *arXiv preprint arXiv:1608.00540*, 2016.

[23] Anton Leykin and Jan Verschelde. Decomposing solution sets of polynomial systems: a new parallel monodromy breakup algorithm. *International Journal of Computational Science and Engineering*, 4(2):94–101, 2009.

[24] A. L. MacLean, Z. Rosen, H. M. Byrne, and H. A. Harrington. Parameter-free methods distinguish Wnt pathway models and guide design of experiments. *Proceedings of the National Academy of Science*, 112:2652–2657, March 2015.

[25] Gregorio Malajovich. Computing mixed volume and all mixed cells in quermassintegral time. *Found. Comput. Math.*, 2016.

[26] Alexander Morgan. *Solving polynomial systems using continuation for engineering and scientific problems*. Prentice Hall Inc., Englewood Cliffs, NJ, 1987.

[27] VW Noonburg. A neural network modeled by an adaptive lotka-volterra system. *SIAM Journal on Applied Mathematics*, 49(6):1779–1792, 1989.

[28] A. J. Sommese, J. Verschelde, and C. W. Wampler. *Using Monodromy to Decompose Solution Sets of Polynomial Systems into Irreducible Components*, pages 297–315. Springer Netherlands, Dordrecht, 2001.

[29] 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.

[30] A.J. Sommese, J. Verschelde, and C.W. Wampler. Introduction to numerical algebraic geometry. In A. Dickenstein and I. Emiris, editors, *Solving polynomial equations*, pages 301–338. Springer-Verlag, 2005.

[31] Andrew J. Sommese and Charles W. Wampler, II. *The numerical solution of systems of polynomials*. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2005.

[32] J. Verschelde. Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Trans. Math. Softw.*, 25(2):251–276, 1999. Available at http://www.math.uic.edu/∼jan.

[33] Jan Verschelde, Pierre Verlinden, and Ronald Cools. Homotopies exploiting newton polytopes for solving sparse polynomial systems. *SIAM J. Numer. Anal.*, 31(3):915–930, June 1994.