Decoupling highly structured polynomial systems

Daniel J. Bates[a], Andrew Newell[b], Matthew Niemerg[a]

[a]*Department of Mathematics, Colorado State University (bates,niemerg@math.colostate.edu, www.math.colostate.edu/~bates,~niemerg).*
Partially supported by NSF grant DMS-1025564.
[b]*Department of Marine, Earth, and Atmospheric Sciences, North Carolina State University (ajnewell@ncsu.edu,www4.ncsu.edu/~ajnewell).*
Partially supported by NSF grant DMS-1025544.

Abstract

An efficient technique for solving polynomial systems with a particular structure is presented. This structure is very specific but arises naturally when computing the critical points of a symmetric polynomial energy function. This novel numerical solution method is based on homotopy continuation and may apply more broadly than this specific setting. An illustrative example from magnetism is presented, along with some timing comparisons and open problems.

Keywords: polynomial system, homotopy continuation, decoupling
2000 MSC: 65H10, 65H20

1. Introduction

Let $f:\mathbb{C}^{Nk}\to\mathbb{C}^{Nk}$ be a polynomial system of the form

$$f(z_1,z_2,\ldots,z_N)=\begin{cases} f_1(z_1,z_2,\ldots,z_N)\\ f_2(z_1,z_2,\ldots,z_N)\\ \vdots\\ f_N(z_1,z_2,\ldots,z_N)\end{cases}$$

$$=\begin{cases} g_1(z_1) & & & & + & h_1(z_1,z_2,\ldots,z_N)\\ & g_2(z_2) & & & + & h_2(z_1,z_2,\ldots,z_N)\\ & & \ddots & & & \vdots\\ & & & g_N(z_N) & + & h_N(z_1,z_2,\ldots,z_N),\end{cases} \tag{1}$$

where each f_i, g_i, and h_i is a k-tuple of polynomials and the z_i are non-overlapping k-tuples of variables for $i = 1, \ldots, N$. Notice that, for each i, g_i depends only on variables z_i while h_i may depend on all of the variables. We further require the polynomials g_i to all have the same monomial structure. As a very simple example, suppose that the set of equations breaks into 1-tuples and, for each i, g_i is a single polynomial given by $g_i(x) = a_i x_i^3 + b_i$, with the coefficients a_i and b_i being complex numbers. Then, the only difference between $g_i(x_i)$ and another block $g_j(x_j)$ is in their coefficients and the labels on their variables.

The isolated solutions of

$$f(z_1, z_2, \cdots, z_N) = 0$$

in $\mathbb{C}^{Nk}$ (including all real solutions isolated over the complex numbers) may be approximated to arbitrarily high accuracy with numerical algebraic geometry, based on homotopy continuation. In this article, we present a novel method, *decoupling*, that will provide at least some of these solutions (sometimes all) much more efficiently and in a more scalable manner.

While the structure required in (1) may seem restrictive, it is precisely the structure attained when computing the critical points of a symmetric polynomial energy function by solving the system of first partial derivatives. We encountered this structure when working on a particular set of magnetism problems with multiple dipoles. The terms $g_i(z_i)$ come from the energy that dipole i would have in the absence of the other dipoles, while the terms $h_i(z_1, z_2, \cdots, z_N)$ come from interactions between dipoles. The scenario is not restricted to magnetism and should arise whenever there is a system with both long-range interactions and *internal energies*. In fact, polynomial systems of this sort could come from any number of applications or constructions, so this method is not restricted to magnetism or perhaps even to physics.

A sketch of the decoupling method follows. We construct a system g_* that has the same monomial structure as the g_i but with random, complex values for the coefficients. We solve this system using homotopy continuation. We use these solutions as starting points for solving each of the g_i blocks independently, using efficient parameter homotopies. Since the blocks are independent (there is no overlap in variables), we can construct solutions of

the system $g(z_1, z_2, \ldots, z_N) = 0$, where

$$g(z_1, z_2, \ldots, z_N) = \begin{cases} g_1(z_1) \\ g_2(z_2) \\ \vdots \\ g_N(z_N) \end{cases}, \tag{2}$$

from combinations of solutions from each of the g_i. Finally, we move from the solutions of g to those of the full system f by homotoping in the interactions between the blocks.

In §2, we provide the necessary background on our computational engine, homotopy continuation, particularly in the polynomial system setting. We follow a formal statement of the method in §3 with an illustration in §4. In §5 we provide some computational results. Finally, we discuss two open problems in §6:

Open problems

1. Under what conditions will decoupling necessarily find *all* solutions of $f = 0$?
2. Given a non-structured polynomial system, can we add or remove terms to get a system that we can solve using decoupling, then use homotopy on the result to recover the solutions of the original system?

2. Homotopy continuation

Numerical continuation is a well-known and widely-used tool for approximating solutions of systems of equations. The book Allgower and Georg (2003) is a nice reference for these methods, in a general setting. If we restrict our attention to polynomial equations over complex space, we have a variety of useful guarantees that make feasible the computation of *all* isolated solutions. The field dedicated to the development and implementation of homotopy continuation for polynomial systems is called *numerical algebraic geometry*. General references include Bates et al. (2013c); Sommese and Wampler (2005); Li (1997).

In this section, we introduce only the most basic ideas of homotopy continuation and parameter homotopies, leaving most details to the references. The section concludes with a few remarks about software available for these computations.

2.1. Basic homotopy continuation for polynomial systems

To find the solutions of a polynomial system $f(z) = 0$, homotopy continuation begins with a choice of *start system* $g(z)$ that is similar to $f(z)$ but easily solved. We then deform from $g(z)$ to $f(z)$ via a homotopy function,

$$h(z,t) = tg(z) + (1-t)f(z),$$

as t goes from 1 to 0. With probability one, for each t, $h(z,t)$ is a polynomial system having the same number of solutions K as $h(t,1) = g(z) = 0$. Thus, as t varies, we have K solution paths to follow, which we can follow via numerical predictor-corrector methods. With probability one, paths never cross, and they will either converge to a solution of $f(z) = 0$ or diverge to infinity as t reaches 0. This is the essence of homotopy continuation with virtually no details. Refer to the references above for any details.

There is one point worth considering in this article, the choice of start system $g(z)$. There is a simple, canonical choice called the *Bézout* or *total degree* start system. Function g_i is chosen to be of the form $z_i^{d_i} - 1$, where d_i is the degree of polynomial f_i. This start system is guaranteed to have $\prod_{i=1}^{n} d_i$ isolated, nonsingular solutions for a polynomial system with n equations and variables. Although this start system is trivially built and solved, it also results in the most solution paths of virtually any start system.

Another recent, efficient choice of algorithm is *regeneration* (Hauenstein et al., 2011; Bates et al., 2013b). This equation-by-equation solver does not quite fit the format above. However, the core of the method is still homotopy continuation and regeneration is frequently the most efficient homotopy-based method. In the examples below, we compare our method to both a total degree start system and regeneration.

2.2. Parameter homotopies

The ideal start system for a polynomial system is the system itself, but with the coefficients changed to random complex numbers. The problem with this start system is that it is just as difficult to solve as the original system! However, when such a system is used repeatedly to solve polynomial systems that differ only in their coefficients, we have the very effective *parameter homotopy*.

To use the parameter homotopy method to solve a sequence of polynomial systems, differing only in their coefficients, we begin by choosing a start system with random complex coefficients and solving it using any standard

homotopy method, *e.g.*, a total degree homotopy or regeneration. With probability one, this system has the maximum number of solutions for any polynomial system with the same monomial structure. Thus, all solutions of all polynomial systems of our parameterized family may be solved by moving from the random system to the target systems one at a time. The number of paths saved by using parameter homotopies can be striking, sometimes dropping the number of paths by several orders of magnitude.

It is essential that the first system has random complex coefficients. While it might be tempting to use the target polynomial system, there is no guarantee that it will have the maximum number of solutions. Therefore, we would almost surely miss solutions in subsequent runs.

2.3. Software

There are several free, open source software packages for computations in numerical algebraic geometry. In this paper, we focus on Bertini (Bates et al., 2006). However, PHCpack (Verschelde, 1999) or HOM4PS-2.0 (Lee et al., 2008) could have been used equally well. Similarly, we could have used Paramotopy (Bates et al., 2013a) for our parameter homotopies. Regardless of the choice of software, the relative timings would be about the same.

3. Algorithm

Given a polynomial system of the form (1), the process of computing some or all of the isolated solutions of $f = 0$ using decoupling has four steps.

This algorithm clearly terminates as it consists of a finite number of homotopies, each with a finite number of paths.

Steps 1 and 2 involve homotopies, but Step 3 involves no computation. This is just a reorganization of the data collected in the N Step 2 runs.

3.1. Complexity analysis

In homotopy continuation, the fundamental unit to count is the number of paths tracked. Granted, the quality of paths may vary from problem to problem or algorithm to algorithm, but this is difficult to quantify.

The number of paths to track with a Bézout homotopy is clear and easily computed, just the product of the degrees of the polynomials. Regeneration is more complicated to count as polynomial order and other factors affect both the number of paths and the total run time.

Algorithm 1 Decoupling

Input: Polynomial system $f : \mathbb{C}^{Nk} \to \mathbb{C}^{Nk}$, broken into k-tuples g_i and h_i for $i = 1, \ldots, N$, as in (1). Let $z_1, \ldots, z_N$ be N k-tuples of variables.
Output: Solutions of $f = 0$.

1. Solve $g_*(z) = 0$, a $k \times k$ polynomial system with the same monomial structure as each $g_i(z_i)$ but random, complex coefficients. Use any standard homotopy continuation technique for this.
2. For each $i = 1, \ldots, N$, use a parameter homotopy from $g_*(z)$ to find all isolated solutions of $g_i(z_i) = 0$ in $\mathbb{C}^k$.
3. Combinatorially concatenate all solutions from all blocks to form $N-$*tuples* of solutions $(s_1, \ldots, s_N)$ for $g(z_1, z_2, \ldots, z_N) = 0$, where g is defined in (2). These are the start solutions S for the final step.
4. Solve the original problem $f(z_1, z_2, \ldots, z_N)$ by way of a parameter homotopy from $g(z_1, z_2, \ldots, z_N)$ to $f(z_1, z_2, \ldots, z_N)$, starting from all the solutions in S.

For decoupling, the number of paths in Steps 1 and 2 are virtually negligible compared to the Bézout number for the full system, at least for $k > 1$. Indeed,

$$(N+1)\prod_{i=1}^{k} d_i << \prod_{i=1}^{k} d_i^N$$

is certainly true for nontrivial nonlinear systems, and the left hand side is an upper bound on the number of paths tracked with decoupling. Furthermore, each of these systems consists of only k equations and variables, so the numerical linear algebra underlying homotopy continuation will be much faster than that for the full system. Step 3 is computationally trivial.

Thus, the main complexity consideration for decoupling is the number of paths in Step 4, which is difficult to predict *a priori*. If the number of solutions of each block $g_i(z_i) = 0$ is the Bézout number of that block, then decoupling is a huge waste of time as Step 4 will involve tracking as many paths as a Bézout homotopy for the full system. However, if the number of solutions of each block $g_i(z_i) = 0$ is significantly less than the Bézout number for the block, we expect a savings in the total run time for the problem.

Of course, it should be noted that Bézout homotopies and regeneration will find all isolated solutions of the system (at least if the perturbed version

of regeneration Bates et al. (2013b) is used). With decoupling, there is currently no such guarantee, as described in §6. However, in some settings, decoupling can find *all* the solutions even though each block does not have the total degree number of solutions. This is shown in the following example.

4. Illustrative example from magnetism

We illustrate the algorithm (described formally in §3) with an application in magnetism: a micromagnetic model for interacting single-domain ferromagnets as used in Newell (2009). A single-domain magnet has a moment of fixed magnitude that can only rotate. The moment of each magnet is represented by a vector $\boldsymbol{\mu}_i = V M_\mathrm{s} \mathbf{m}_i$, where V is the volume of the magnet, M_s the saturation magnetization (a property of the material), and $\mathbf{m}_i$ a unit vector. The magnetic energy of a system of magnets is

$$E = E_h + E_d + E_a$$

where E_h is the energy of magnetostatic coupling with the external field; E_d is the magnetostatic self-energy (or "demagnetizing energy") of the system; and E_a is the internal (anisotropy) energy.

If N magnets are in an external magnetic field $\mathbf{H}$ (measured in amperes per meter, or $\mathrm{A\,m^{-1}}$), the energy of coupling with the field is

$$E_h = -\mu_0 M_\mathrm{s} \mathbf{H} \cdot \sum_{i=1}^{N} V_i \mathbf{m}_i,$$

where μ_0 is the permeability of free space ($4\pi \times 10^{-7}\,\mathrm{H\,m^{-1}}$).

The demagnetizing energy is

$$E_d = \frac{\mu_0}{2} M_\mathrm{s}^2 \sum_{i=1, j=1, j\neq i}^{N} V_i \mathbf{m}_i \mathbf{N}_{ij} \mathbf{m}_j,$$

where $\mathbf{N}_{ij}$, called the demagnetizing tensor, depends only on the geometry of the system (the sizes, shapes, orientations and positions of the magnets).

The internal energy of each magnet is the sum of magnetostatic, magnetocrystalline and magnetoelastic factors. To lowest order, the energy of each magnet can be approximated by a quadratic expansion in the direction cosines:

$$E_a = \sum_{i=1}^{N} V_i \mathbf{m}_i \mathbf{K}_i \mathbf{m}_i,$$

where each matrix $\mathbf{K}_i$ depends on the shape of the magnet and the physical properties of the material.

The equilibrium states for this system satisfy $\partial E/\partial \mathbf{m}_i = 0$ and the constraints $|\mathbf{m}_i| = 1$ for all i. In translating these equations into a polynomial system, we assume that all the magnets and their moments are on the same plane, though this model naturally extends to 3 dimensions. Since the spatial coordinates do not appear directly, we can also represent unit vector components $m_{\mathrm{x},i}$ and $m_{\mathrm{z},i}$ as x_i and z_i. The constraints are enforced using Lagrange multipliers w_i. For simplicity we also absorb the factors V_i, μ_0 and M_s into the coefficients. For N magnets there are $3N$ equations and $3N$ variables (w_i, x_i, and z_i for each i). The polynomial expressions for magnet i derived from $\partial E/\partial \mathbf{m}_i$ are:

$$f_i = \begin{cases} (K_{\mathrm{xx},i} - w_i)\, x_i + K_{\mathrm{xz},i} z_i - H_\mathrm{x} + \sum_{j=1,j\neq i}^{N} (N_{\mathrm{xx},ij} x_j + N_{\mathrm{xz},ij} z_j) \\ K_{\mathrm{zx},i} x_i + (K_{\mathrm{zz},i} - w_i)\, z_i - H_\mathrm{z} + \sum_{j=1,j\neq i}^{N} (N_{\mathrm{zx},ij} x_j + N_{\mathrm{zz},ij} z_j) \\ x_i^2 + z_i^2 - 1. \end{cases}$$

In these polynomials, the variables are in 3-tuples $\lambda_i \equiv (x_i, z_i, w_i)$. The isolated-magnet terms are

$$g_i(x, z, w) = \begin{cases} (K_{\mathrm{xx},i} - w)\, x + K_{\mathrm{xz},i} z - H_\mathrm{x} \\ K_{\mathrm{zx},i} x + (K_{\mathrm{zz},i} - w)\, z - H_\mathrm{z} \\ x^2 + z^2 - 1 \end{cases},$$

while the interaction terms are

$$h_i(\lambda_1, \ldots, \lambda_N) = \begin{cases} \sum_{j=1,j\neq i}^{N} (N_{\mathrm{xx},ij} x_j + N_{\mathrm{xz},ij} z_j) \\ \sum_{j=1,j\neq i}^{N} (N_{\mathrm{zx},ij} x_j + N_{\mathrm{zz},ij} z_j) \\ 0 \end{cases}.$$

(The third polynomial is identically zero.)

We first solve $g_* = 0$, a system with the same monomial support as each of the g_i but with all coefficients set to randomly-chosen complex numbers. This is Step 1 of the Algorithm. Any standard homotopy continuation method could be used to solve this simple 3 equation, 3 variable system. Denote the set of isolated solutions of $g_* = 0$ by S_*.

For each $i = 1, \ldots, N$, solve $g_i = 0$ via parameter homotopy from g_* and solutions S_*. This is Step 2. This produces solution sets S_i. Notice that there will need to be a trivial relabeling of variables to carry out these parameter homotopies.

The set of all solutions of the system $g_1 = \ldots = g_N = 0$ is then simply the product $S = S_1 \times S_2 \times \cdots \times S_N$ (Step 3). The original system can then be solved via Step 4, with a homotopy from $g_1 = \ldots = g_N = 0$, starting with the points in S.

In this example, half of the paths diverge to infinity in each Step 2 run. When combinatorially building the solutions, we attain a start system with the same number of solutions as the system of interest. None of the paths tracked from these starting points diverge to infinity, and we capture all the solutions. We cannot *guarantee* that we will always find all the complex solutions for a system of this kind. However, due to the highly structured nature of this system, a block system setup works in this case. See Open Problem #1 in §6 for more on this important point.

5. Computational results

There is a variation on the decoupling algorithm not described above. Rather than moving directly to each $g_i(z_i)$ and to the final polynomial system, one could instead solve a randomized system (meaning random complex numbers) in place of each $g_i(z_i)$, then move from this randomized system to the final system. While this generality is perhaps favorable in capturing more solutions, the results described below show that there is a cost in doing so.

The illustrative example of the previous section was run for varying N on a single 2.67 GHz Xeon processor running the CentOS operating system. Table 1 provides the average of 10 runs per entry for $N = 2, \ldots, 8$, using a Bézout homotopy, regeneration, the basic decoupling algorithm, and the previously described variant. Timings were stopped at 24 hours.

6. Open problems

6.1. Conditions on finding all solutions

A fundamental issue with the proposed decoupling algorithm is whether it will produce *all* isolated solutions of the polynomial system. In the illustrative example, it does so. However, an example below shows that (some variation of) decoupling fails to produce all solutions.

In one situation, we know that decoupling will produce all isolated solutions. It is true that if each block has the total degree number of solutions,

N	Bézout	Regeneration	Basic Decoupling	Decoupling Variant
2	0.5	0.6	**0.4**	0.2
3	11.0	2.4	**0.8**	0.6
4	268.6	13.2	**1.6**	1.9
5	3757.2	89.0	**3.7**	8.6
6	59131.5	519.9	**18.9**	53.5
7	$>$ 24 hours	3470.7	**105.2**	344.3
8	$>$ 24 hours	18109.6	**903.6**	2269.6

Table 1: Average run time of 10 runs for four ways of computing the solutions of the illustrative example – Bézout, regeneration, basic decoupling, and the decoupling variant – for various values of N. Timings in seconds except where stated otherwise.

combinatorially building up a start system and turning on and off various interaction terms will work. Doing this amounts to nothing more than creating a different start system with the total degree number of solutions, but not the default start system created by Bertini.

At present, the most we can say in general is that decoupling *may* produce *some* of the isolated solutions of the system. Of course, for very large polynomial systems, this could be helpful if the Bézout number of the full system is too large to handle. We hope to later find other criteria that indicate when we can find *all* solutions.

6.2. Extension to non-structured systems

Based on our experiences with decoupling, we were led to consider a slightly more general method. Indeed, suppose we could add or remove terms from some polynomial system to cause it to break into a set of several smaller, decoupled systems. We would require that each block depend only on a number of variables equal to the number of equations in the block and that each variable appear in exactly one block. While we could dictate that each block have the same monomial structure (as with basic decoupling), we relax that requirement in the examples below.

The following examples shows that this extension of decoupling may fail.

Example 6.1 Suppose

$$\begin{aligned} f_1 &= p_1x^3y + p_2y^2x^2 + p_3y^2x + p_4x^2y - 1 + p_{12}wz + p_{13}wz^2y \\ f_2 &= p_5x^2y + p_6y^2x - 4 + p_{14}w^2y + p_{15}wxz \\ f_3 &= p_7wz + p_8z^3 + p_9w - 3 + p_{16}yz + p_{17}x^2 \\ f_4 &= p_{10}wz - 1 + p_{18}xz + p_{19}y^2x \end{aligned}$$

is the system we are interested in solving.

If we naively choose our two subsystems to be

$$\begin{aligned} f_1 &= p_1x^3y + p_2y^2x^2 + p3y^2x + p4x^2y - 1 \\ f_2 &= p_5x^2y + p_6y^2x - 4 \\ &\text{and} \\ f_3 &= p_7wz + p_8z^3 + p_9w - 3 \\ f_4 &= p_{10}wz + p_{11}w^3 - 1 \end{aligned}$$

we will not capture all the solutions with decoupling.

The first subsystem has Bézout count 12 but only 5 solutions. The second subsystem has Bézout count 9 and exactly that many solutions. When combinatorially building the start system when we bring in the coupled terms between the subsystems, we only have $5(9) = 45$ starting solutions. Out of these 45 starting solutions, 30 of the paths remain finite, and the other paths diverge. Unfortunately, the system that we are attempting to solve has 57 solutions.

While this example shows that this more general idea need not succeed, some version of it has succeeded in the past. The authors of Hao et al. (2013) aimed to solve a system of partial differential equations related to tumor growth. They found it useful to first solve the system with an extra viscosity term, then let the coefficient of that term go to zero. This allowed them to solve polynomial systems that they were not otherwise able to solve. The open question here is when this sort of *surgery on monomials* is guaranteed to yield all isolated solutions.

References

Allgower, E., Georg, K., 2003. Introduction to numerical continuation methods. volume 45 of *SIAM Classics in Applied Math*.

Bates, D., Brake, D., Niemerg, M., 2013a. Paramotopy: Parameter homotopies in parallel. Submitted.

Bates, D., Davis, B., Eklund, D., Hanson, E., Peterson, C., 2013b. Perturbed regeneration for finding all isolated solutions of polynomial systems. Submitted.

Bates, D., Hauenstein, J., Sommese, A., Wampler, C., 2013c. Numerically solving polynomial systems with Bertini. SIAM.

Bates, D., Hauenstein, J.D., Sommese, A., Wampler, C., 2006. Bertini: Software for numerical algebraic geometry. Software available at www.bertini.nd.edu.

Hao, W., Hu, B., Sommese, A., 2013. Cell cycle control and bifurcation for a free boundary problem modeling tissue growth. Submitted.

Hauenstein, J., Sommese, A., Wampler, C., 2011. Regeneration homotopies for solving systems of polynomials. Mathematics of Computation 80, 345–377.

Lee, T., Li, T., , Tsai, C., 2008. Hom4ps–2.0, a software package for solving polynomial systems by the polyhedral homotopy continuation method. Computing 83, 109–133.

Li, T., 1997. Numerical solution of multivariate polynomial systems by homotopy continuation methods. Acta Numerica 6, 399–436.

Newell, A.J., 2009. Transition to superparamagnetism in chains of magnetosome crystals. Geochem. Geophys. Geosyst. 10. doi:10.1029/2009GC002538.

Sommese, A., Wampler, C., 2005. Numerical solution of polynomial systems arising in engineering and science. World Scientific.

Verschelde, J., 1999. Algorithm 795: Phcpack: A general-purpose solver for polynomial systems by homotopy continuation. ACM Trans. Math. Softw. 25, 251–276. Software and example systems available at www.math.uic.edu/∼jan.