Solving Polynomial Systems by Homotopy Continuation

Andrew Sommese
University of Notre Dame

www.nd.edu/~sommese
Reference on the area up to 2005:

Survey covering other topics
Overview

- Solving Polynomial Systems
- Computing Isolated Solutions
  - Homotopy Continuation
  - Case Study: Alt’s nine-point path synthesis problem for planar four-bars
- Positive Dimensional Solution Sets
  - How to represent them
  - Decomposing them into irreducible components
- Numerical issues posed by multiplicity greater than one components
  - Deflation and Endgames
  - Bertini and the need for adaptive precision
- A Motivating Problem and an Approach to It
  - Fiber Products
  - A positive dimensional approach to finding isolated solutions equation-by-equation
Solving Polynomial Systems

- Find all solutions of a polynomial system on $\mathbb{C}^N$:

\[
\begin{bmatrix}
  f_1(x_1, \ldots, x_N) \\
  \vdots \\
  f_n(x_1, \ldots, x_N)
\end{bmatrix} = 0
\]
Why?

- To solve problems from engineering and science.
Characteristics of Engineering Systems

- systems are sparse: they often have symmetries and have much smaller solution sets than would be expected.
Characteristics of Engineering Systems

- systems are sparse: they often have symmetries and have much smaller solution sets than would be expected.
- systems depend on parameters: typically they need to be solved many times for different values of the parameters.
Characteristics of Engineering Systems

- systems are sparse: they often have symmetries and have much smaller solution sets than would be expected.
- systems depend on parameters: typically they need to be solved many times for different values of the parameters.
- usually only real solutions are interesting
Characteristics of Engineering Systems

- Systems are sparse: they often have symmetries and have much smaller solution sets than would be expected.
- Systems depend on parameters: typically they need to be solved many times for different values of the parameters.
- Usually only real solutions are interesting.
- Usually only finite solutions are interesting.
Characteristics of Engineering Systems

- systems are sparse: they often have symmetries and have much smaller solution sets than would be expected.
- systems depend on parameters: typically they need to be solved many times for different values of the parameters.
- usually only real solutions are interesting.
- usually only finite solutions are interesting.
- nonsingular isolated solutions were the center of attention.
Computing Isolated Solutions

Find all isolated solutions in $\mathbb{C}^N$ of a system on $n$ polynomials:

$$
\begin{bmatrix}
  f_1(x_1,\ldots,x_N) \\
  \vdots \\
  f_n(x_1,\ldots,x_N)
\end{bmatrix} = 0
$$
Solving a system

- Homotopy continuation is our main tool:
  - Start with known solutions of a known start system and then track those solutions as we deform the start system into the system that we wish to solve.
Path Tracking

This method takes a system $g(x) = 0$, whose solutions we know, and makes use of a homotopy, e.g.,

$$H(x,t) = (1 - t)f(x) + tg(x).$$

Hopefully, $H(x,t)$ defines “nice paths” $x(t)$ as $t$ runs from 1 to 0. They start at known solutions of $g(x) = 0$ and end at the solutions of $f(x)$ at $t = 0$. 

The paths satisfy the Davidenko equation

\[ 0 = \frac{dH(x(t), t)}{dt} = \sum_{i=1}^{N} \frac{\partial H}{\partial x_i} \frac{dx_i}{dt} + \frac{\partial H}{\partial t} \]

To compute the paths: use ODE methods to predict and Newton’s method to correct.
Solutions of 
\[ f(x) = 0 \]

\[ H(x,t) = (1-t) f(x) + t g(x) \]

Known solutions of 
\[ g(x) = 0 \]

\( x_1(t) \)
\( x_2(t) \)
\( x_3(t) \)
\( x_4(t) \)
Newton correction

prediction

\[ x_j(t) \]

\[ x^* \]

\[ \Delta t \]

\[ 0 \]

\[ 1 \]
Algorithms

- middle 80’s: Projective space was beginning to be used, but the methods were a combination of differential topology and numerical analysis with homotopies tracked exclusively through real parameters.
- early 90’s: algebraic geometric methods worked into the theory: great increase in security, efficiency, and speed.
Simple but extremely useful consequence of algebraicity [A. Morgan (GM R. & D.) and S.]

- Instead of the homotopy  \( H(x,t) = (1-t)f(x) + tg(x) \)
use \( H(x,t) = (1-t)f(x) + \gamma tg(x) \)
Genericity

- Morgan + S.: if the parameter space is irreducible, solving the system at a random point simplifies subsequent solves: in practice speedups by factors of 100.
Endgames (Morgan, Wampler, and S.)

- Example: \((x - 1)^2 - t = 0\)

We can uniformize around a solution at \(t = 0\). Letting \(t = s^2\), knowing the solution at \(t = 0.01\), we can track around \(|s| = 0.1\) and use Cauchy’s Integral Theorem to compute \(x\) at \(s = 0\).
Special Homotopies to take advantage of sparseness
Multiprecision

- Not practical in the early 90’s!
  - Highly nontrivial to design and dependent on hardware
  - Hardware too slow
Hardware

- Continuation is computationally intensive. On average:
  - in 1985: 3 minutes/path on largest mainframes.
Hardware

- Continuation is computationally intensive.
  On average:
  - in 1985: 3 minutes/path on largest mainframes.
  - in 1991: over 8 seconds/path, on an IBM 3081;
    2.5 seconds/path on a top-of-the-line IBM 3090.
Continuation is computationally intensive. On average:

- in 1985: 3 minutes/path on largest mainframes.
- in 1991: over 8 seconds/path, on an IBM 3081; 2.5 seconds/path on a top-of-the-line IBM 3090.
- 2006: about 10 paths a second on a single processor desktop CPU; 1000’s of paths/second on moderately sized clusters.
A Guiding Principle then and now

- Algorithms must be structured – when possible – to avoid extra paths and especially those paths leading to singular solutions: find a way to never follow the paths in the first place.
Continuation’s Core Computation

- Given a system $f(x) = 0$ of $n$ polynomials in $n$ unknowns, continuation computes a finite set $S$ of solutions such that:
  - any isolated root of $f(x) = 0$ is contained in $S$;
  - any isolated root “occurs” a number of times equal to its multiplicity as a solution of $f(x) = 0$;
  - $S$ is often larger than the set of isolated solutions.
Case Study: Alt’s Problem

- We follow

A four-bar planar linkage is a planar quadrilateral with a rotational joint at each vertex.

They are useful for converting one type of motion to another.

They occur everywhere.
How Do Mechanical Engineers Find Mechanisms?

- Pick a few points in the plane (called precision points)
- Find a coupler curve going through those points
- If unsuitable, start over.
- Having more choices makes the process faster.
- By counting constants, there will be no coupler curves going through more than nine points.
Nine Point Path-Synthesis Problem

H. Alt, Zeitschrift für angewandte Mathematik und Mechanik, 1923:

- Given nine points in the plane, find the set of all four-bar linkages, whose coupler curves pass through all these points.
First major attack in 1963 by Freudenstein and Roth.
\[ v = y - b \]
\[ \text{ve}^{i\mu_j} = \text{ye}^{i\tau_j} - (b - d_j) \]
\[ = \text{ye}^{i\tau_j} + d_j - b \]
We use complex numbers (as is standard in this area)

Summing over vectors we have 16 equations

\[(y - b)e^{i\mu_j} = ye^{i\theta_j} + \delta_j - b\]

\[(x - a)e^{i\lambda_j} = xe^{i\theta_j} + \delta_j - a\]

plus their 16 conjugates

\[(\bar{y} - \bar{b})e^{-i\mu_j} = \bar{y}e^{-i\theta_j} + \bar{\delta}_j - \bar{b}\]

\[(\bar{x} - \bar{a})e^{-i\lambda_j} = \bar{x}e^{-i\theta_j} + \bar{\delta}_j - \bar{a}\]
This gives 8 sets of 4 equations:

\[(x - a)e^{i\lambda_j} = xe^{i\delta_j} + \delta_j - a\]

\[(y - b)e^{i\mu_j} = ye^{i\delta_j} + \delta_j - b\]

\[(\bar{y} - \bar{b})e^{-i\mu_j} = \bar{y}e^{-i\delta_j} + \bar{\delta}_j - \bar{b}\]

\[(\bar{x} - \bar{a})e^{-i\lambda_j} = \bar{x}e^{-i\delta_j} + \bar{\delta}_j - \bar{a}\]

in the variables \(a, b, x, y, \bar{a}, \bar{b}, \bar{x}, \bar{y}\), and

\(?_j, \mu_j, \theta_j\) for \(j\) from 1 to 8.
Multiplying each side by its complex conjugate and letting $\alpha_j = e^{i \theta_j} - 1$ we get 8 sets of 3 equations

$$[\bar{a} - \bar{d}_j)x_j] + [(a - d_j)x_j] + d_j(a - x) + \bar{d}_j(a - x) - d_j\bar{d}_j = 0$$

$$[\bar{b} - \bar{d}_j)y_j] + [(b - d_j)y_j] + d_j(b - y) + \bar{d}_j(b - y) - d_j\bar{d}_j = 0$$

$$\alpha_j + \beta_j + \gamma_j = 0$$

in the 24 variables $a, b, x, y, \bar{a}, \bar{b}, \bar{x}, \bar{y}$ and $\alpha_j, \beta_j, \gamma_j$ with $j$ from 1 to 8.
Freudenstein and Roth go to real and imaginary parts or replace $\overline{a}, \overline{b}, \overline{x}, \overline{y}, \overline{\gamma}_j$ with new variables $\hat{a}, \hat{b}, \hat{x}, \hat{y}, \hat{\gamma}_j$. We choose the second approach—when we are done we will have to check the reality conditions $\overline{a} = \hat{a}, \overline{x} = \hat{x}$ etc.
\[
\begin{align*}
[(\hat{a} - \hat{d}_j)x]_j & + [(a - d_j)\hat{x}]_j + d_j(\hat{a} - \hat{x}) + \bar{d}_j(a - x) - d_j\bar{d}_j = 0 \\
[(\hat{b} - \hat{d}_j)y]_j & + [(b - d_j)\hat{y}]_j + d_j(\hat{b} - \hat{y}) + \bar{d}_j(b - y) - d_j\bar{d}_j = 0 \\
\hat{?}_j + ?_j + ?_j ?_j & = 0
\end{align*}
\]

in the 24 variables \( a, b, x, y, \hat{a}, \hat{b}, \hat{x}, \hat{y} \) and \(?_j, ?_j \)
with \( j \) from 1 to 8.
Note we have 24 equations of which 16 are degree 3 and 8 are degree 2. This would give a total possible number of solutions, $2^8 3^{16} = 11,019,960,576$—allowing 1 second a path it would take over 300 years to solve this system.
Using Cramer’s rule and substitution we have what is essentially the Freudenstein-Roth system consisting of 8 equations of degree 7. Impractical to solve in early 90’s:

\[ 7^8 = 5,764,801 \text{ solutions.} \]
Newton’s method doesn’t find many solutions: Freudenstein and Roth used a simple form of continuation combined with heuristics.

Tsai and Lu using methods introduced by Li, Sauer, and Yorke found only a small fraction of the solutions. That method requires starting from scratch each time the problem is solved for different parameter values.
We followed a different route by introducing new variables \(n, \hat{n}, m, \hat{m}\) so that \(n = a \hat{x}, \hat{n} = a x, m = b \hat{y}, \hat{m} = b y\). We group the variables into 10 groups \(\{\gamma_j, \hat{\gamma}_j\}, \{x, \hat{x}, a, \hat{a}, n, \hat{n}\}, \{y, \hat{y}, b, \hat{b}, m, \hat{m}\}\) for \(j = 1, \ldots, 8\). Introducing homogeneous coordinates into each group, we use Cramer’s rule to reduce to a system of 12 equations in 12 unknowns: 4 quadrics and 8 quartics. Though the Bézout number is 1,048,576, the 2-homogeneous Bézout number is 286,720, and there is an involution reducing the work to following 143,360 paths. There is also an order 3 symmetry...
Solve by Continuation

All 2-homog. systems

All 9-point systems

“numerical reduction” to test case (done 1 time)

synthesis program (many times)
Summary

- Analytical Reduction
  Initial formulation ............... $\approx 10^{10}$
  Roth & Freudenstein ............ 5,764,801
  Our elimination ................. 1,048,576
  Multi-homogenization ........... 286,720
  Symmetry ......................... 143,360

- Numerical Reduction
  Nondegenerate .................... 4326
  Roberts cognates ............... 1442

- Synthesis program tracks 1442 solution paths.
Intermission
We now turn to finding the positive dimensional solution sets of a system

\[
\begin{bmatrix}
  f_1(x_1, \ldots, x_N) \\
  \vdots \\
  f_n(x_1, \ldots, x_N)
\end{bmatrix} = 0
\]
How to represent positive dimensional components?

- S. + Wampler in ’95:
  - Use the intersection of a component with generic linear space of complementary dimension.
  - By using continuation and deforming the linear space, as many points as are desired can be chosen on a component.
- Use a generic flag of affine linear spaces to get witness point supersets
- This approach has 19th century roots in algebraic geometry
The Numerical Irreducible Decomposition

Carried out in a sequence of articles with Jan Verschelde (University at Illinois at Chicago) and Charles Wampler (General Motors Research and Development)

- Efficient Computation of “Witness Supersets”

- Numerical Irreducible Decomposition
- An efficient algorithm using monodromy

- Intersection of algebraic sets
Symbolic Approach with same classical roots

Two nonnumerical articles in this direction:

The solution set $Z := V(f)$ decomposes as

$$\bigcup_{i=0}^{\dim Z} Z_i$$

where $Z_i$ is pure $i$-dimensional, and

$$Z_i = \bigcup_{j \in \mathcal{I}_i} Z_{i,j}$$

where

1. each $Z_{i,j}$ is irreducible, i.e., for each $i, j$, the Zariski open and dense set of smooth points of $Z_{i,j}$ is connected;

2. $\mathcal{I}_j$ is finite, and no $Z_{i,j}$ is contained in the union of all the remaining $Z_{i',j'}$. 
Witness Point Sets

Given: Polynomial system of \( n \) equations, \( N \) variables

\[
f(x) = \{f_1(x), \ldots, f_n(x)\} = 0, \quad x = (x_1, \ldots, x_N) \in \mathbb{C}^N
\]

Suppose \( V \in \mathbb{C}^N \) is

- a component of the solution set of \( f(x) = 0 \)
- irreducible (its regular points are connected in \( \mathbb{C}^N \))
- degree \( d \) and dimension \( i \)

Then

A witness point set for \( V \) is

- \( d \) distinct points
- that lie on \( V \cap L_{N-i} \)
- where \( L_{N-i} \) is a generic linear set of dimension \( N - i \).
In the *numerical irreducible decomposition* we want for each component $Z_{i,j}$

- a finite set $Z_{i,j}$ of $\deg Z_{i,j}$ points $Z_{i,j} \cap L_{N-i}$, where $L_{N-i}$ is a generic affine linear space $\mathbb{C}^{N-i}$; and

- a “probability one” test for a point $x \in \mathbb{C}^N$ to lie on a given $Z_{i,j}$. 
Basic Steps in the Algorithm

The algorithm breaks into a few theoretically distinct steps, which in actual implementation are intertwined.

1. Find finite sets $\mathcal{W}_i$ with $i$ running from 0 to $\dim Z$, with $\mathcal{W}_i = Z_i \cup J_i$, where $Z_i := \bigcup_{j \in I_i} Z_{i,j}$ and $J_i \subset \bigcup_{k > i} Z_k$.

2. Give a criterion for a point $x$ to be a member of any of the sets $Z_i$, and as a consequence for $x \in \mathcal{W}_i$ to be in $J_i$.

3. Refine the criterion of step 2) to a membership test for the $Z_{i,j}$, and in particular break $Z_i$ into the $Z_{i,j}$. 
Example

From a paper with Verschelde and Wampler

\[ f = \begin{bmatrix}
(y - x^2)(x^2 + y^2 + z^2 - 1)(x - 0.5) \\
(z - x^3)(x^2 + y^2 + z^2 - 1)(y - 0.5) \\
(y - x^2)(z - x^3)(x^2 + y^2 + z^2 - 1)(z - 0.5)
\end{bmatrix}. \]
Because the equations are in a factored form it is easy to identify the irreducible decomposition of $Z = V(f)$, as

\[ Z = Z_2 \cup Z_1 \cup Z_0 = \{Z_{21}\} \cup \{Z_{11} \cup Z_{12} \cup Z_{13} \cup Z_{14}\} \cup \{Z_{01}\} \]

1. $Z_{21}$ is the sphere $x^2 + y^2 + z^2 - 1 = 0$,

2. $Z_{11}$ is the line $(x = 0.5, z = 0.5^3)$,

3. $Z_{12}$ is the line $(x = \sqrt{0.5}, y = 0.5)$,

4. $Z_{13}$ is the line $(x = -\sqrt{0.5}, y = 0.5)$,

5. $Z_{14}$ is the twisted cubic $(y - x^2 = 0, z - x^3 = 0)$,

6. $Z_{01}$ is the point $(x = 0.5, y = 0.5, z = 0.5)$. 
Numerical issues posed by multiple components

Consider a toy homotopy

\[ H(x_1, x_2, t) = \begin{bmatrix} x_1^2 \\ x_1 \\ x_2 - t \end{bmatrix} = 0 \]

Continuation is a problem because the Jacobian with respect to the x variables is singular.

How do we deal with this?
Deflation

The basic idea introduced by Ojika in 1983 is to differentiate the multiplicity away. Leykin, Verschelde, and Zhao gave an algorithm for an isolated point that they showed terminated. Given a system \( f \), replace it with

\[
\begin{bmatrix}
  f(x) \\
  \mathbf{J}f(x) \cdot \mathbf{z} \\
  \mathbf{A} \cdot \mathbf{z} + \mathbf{b}
\end{bmatrix} = 0
\]
Bates, Hauenstein, Sommese, and Wampler:
To make a viable algorithm for multiple components, it is necessary to make decisions on ranks of singular matrices. To do this reliably, endgames are needed.
Bertini and the need for adaptive precision

- Why use Multiprecision?
  - to ensure that the region where an endgame works is not contained in the region where the numerics break down;
Bertini and the need for adaptive precision

- Why use Multiprecision?
  - to ensure that the region where an endgame works is not contained in the region where the numerics break down;
  - to ensure that a polynomial being zero at a point is the same as the polynomial numerically being approximately zero at the point;
Bertini and the need for adaptive precision

- **Why use Multiprecision?**
  - to ensure that the region where an endgame works is not contained in the region where the numerics break down;
  - to ensure that a polynomial being zero at a point is the same as the polynomial numerically being approximately zero at the point;
  - to prevent the linear algebra in continuation from falling apart.
Evaluation

\[ p(z) = z^{10} - 28z^9 + 1 \]

- To 15 digits of accuracy one of the roots of this polynomial is \( a = 27.9999999999999 \). Evaluating \( p(a) \) exactly to 15 digits, we find that \( p(a) = -0.0578455953407660 \).

- Even with 17 digit accuracy, the approximate root is \( a = 27.999999999999999 \) and we still only have \( p(a) = -0.0049533155737293130 \).
Wilkinson’s Theorem in Numerical Linear Algebra

- Solving $Ax = f$, with $A$ an $N$ by $N$ matrix, we must expect to lose $\log_{10}[\text{cond}(A)]$ digits of accuracy. Geometrically, $\text{cond}(A) = \| A \| \cdot \| A^{-1} \|$ is on the order of the inverse of the distance in $P^{N \times N - 1}$ from $A$ to the set defined by $\det(A) = 0$. 
One approach is to simply run paths that fail over at a higher precision, e.g., this is an option in Jan Verschelde’s code, PHC.
- One approach is to simply run paths that fail over at a higher precision, e.g., this is an option in Jan Verschelde’s code, PHC.
- Bertini is designed to dynamically adjust the precision to achieve a solution with a prespecified error. Bertini is being developed by Dan Bates, Jon Hauenstein, Charles Wampler, and myself (with some early work by Chris Monico). First release scheduled for October 1.
Issues

- You need to stay on the parameter space where your problem is: this means you must adjust the coefficients of your equations dynamically.
Issues

- You need to stay on the parameter space where your problem is: this means you must adjust the coefficients of your equations dynamically.
- You need rules to decide when to change precision and by how much to change it.
The theory we use is presented in the article

This is joint work with Charles Wampler. The problem is to find the families of overconstrained mechanisms of specified types.
If the lengths of the six legs are fixed the platform robot is usually rigid.

Husty and Karger made a study of exceptional lengths when the robot will move: one interesting case is when the top joints and the bottom joints are in a configuration of equilateral triangles.
Another Example

Fig. 9.1 Schematic six-revolute serial-link robot
Overconstrained Mechanisms

1. If \( n \geq 7 \) every element of \( X \) is a mechanism, i.e.,
   “moves.”

2. If \( n \leq 6 \) there are more constraints than
   parameters—not every element of \( X \) is a mechanism.
   (a) If \( n \leq 3 \), then there are no mechanisms.
   (b) If \( n = 4 \), Delassus (1922) showed that the three
       known classes of mechanisms were all that were
       possible.
   (c) If \( n = 5, 6 \), a number of mechanisms have been
       found, but the complete list is not known.
To automate the finding of such mechanisms, we need to solve the following problem:

- Given an algebraic map $p$ between irreducible algebraic affine varieties $X$ and $Y$, find the irreducible components of the algebraic subset of $X$ consisting of points $x$ with the dimension of the fiber of $p$ at $x$ greater than the generic fiber dimension of the map $p$. 
An approach

- A method to find the exceptional sets

- An approach to large systems with few solutions
Summary

- Many Problems in Engineering and Science are naturally phrased as problems about algebraic sets and maps.
- Numerical analysis (continuation) gives a method to manipulate algebraic sets and give practical answers.
- Increasing speedup of computers, e.g., the recent jump into multicore processors, continually expands the practical boundary into the purely theoretical region.
Newton failure

The polynomial system of Griewank and Osborne
(Analysis of Newton's method at irregular

\[
\begin{bmatrix}
\frac{29}{16} z^3 - 2 zw \\
\quad w - z^2
\end{bmatrix} = 0
\]

Newton’s Method fails for any point sufficiently near
the origin (other than the origin).