Recent Results in Numerical Algebraic Geometry

Andrew Sommese

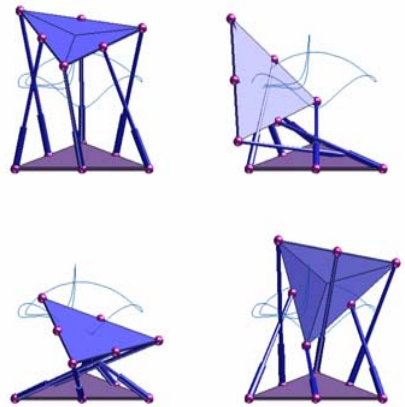
University of Notre Dame

Numerical Algebraic Geometry

- **Goal:** To develop and implement numerical algorithms, based on algebraic geometry:
  - To solve problems from engineering and science that currently do not have a solution
  - To do exploratory computations in algebraic geometry
- **Technical Challenge:** To combine high performance numerics with algebraic geometry
- **Applications:**
  - Robotics and Mechanism Theory
  - Computation of algebraic-geometric invariants
  - Solution of discretizations of nonlinear differential equations
  - Chemical Reactions including combustion

graphics on right from Sommese-Wampler Book

Robotics/Mechanism Theory



Combustion

$$
\begin{aligned}
O_2 &\rightleftharpoons 2O & k_1 X_{O_2} &= X_O^2 \\
H_2 &\rightleftharpoons 2H & k_2 X_{H_2} &= X_H^2 \\
N_2 &\rightleftharpoons 2N & k_3 X_{N_2} &= X_N^2 \\
CO_2 &\rightleftharpoons O + CO & k_4 X_{CO_2} &= X_O X_{CO} \\
OH &\rightleftharpoons O + H & k_5 X_{OH} &= X_O X_H \\
H_2O &\rightleftharpoons O + 2H & k_6 X_{H_2O} &= X_O X_H^2 \\
NO &\rightleftharpoons O + N & k_7 X_{NO} &= X_O X_N .
\end{aligned}
$$

There are four conservation equations:

$$
\begin{aligned}
T_H &= X_H + 2X_{H_2} + X_{OH} + 2X_{H_2O} \\
T_C &= X_{CO} + X_{CO_2} \\
T_O &= X_O + X_{CO} + 2X_{O_2} + 2X_{CO_2} + X_{OH} + X_{H_2O} + X_{NO} \\
T_N &= X_N + 2X_{N_2} + X_{NO}
\end{aligned}
$$

Overview of Talk

- **Solving Polynomial Systems**
  - **Homotopy Continuation to Compute Isolated Solutions.**
    Alt's Problem: a case study.
  - **Positive Dimensional Solution Sets**
- **Numerical issues and Bertini**
  - **Deflation and Endgames to deal with multiple components**
  - **The need for adaptive precision**
- **Regeneration: a new approach to finding isolated solutions**
- **Two applications**
  - **Overconstrained Mechanisms and Fiber Products**
  - **Geometric Genus of Curve Components**

Solving Polynomial Systems

- Find all solutions of a polynomial system $\mathbb{F}^{\mathbb{Q}}$:

$$\begin{bmatrix} f_1(z_1,\ldots,z_N) \\ \vdots \\ f_n(z_1,\ldots,z_N) \end{bmatrix} = 0$$

Computing Isolated Solutions

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

$$\begin{bmatrix} f_1(z_1,\ldots,z_N) \\ \vdots \\ f_n(z_1,\ldots,z_N) \end{bmatrix} = 0$$

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.

Continuation's Core Computations

- Given a system f(z) = 0 of N polynomials in N unknowns, continuation computes a finite set S of solutions such that:
  - any isolated root of f(z) = 0 is contained in S;
  - any isolated root "occurs" a number of times equal to its multiplicity as a solution of f(z) = 0;
  - S is often larger than the set of isolated solutions.
- Continuation allows us to vary parameters.

Path Tracking

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

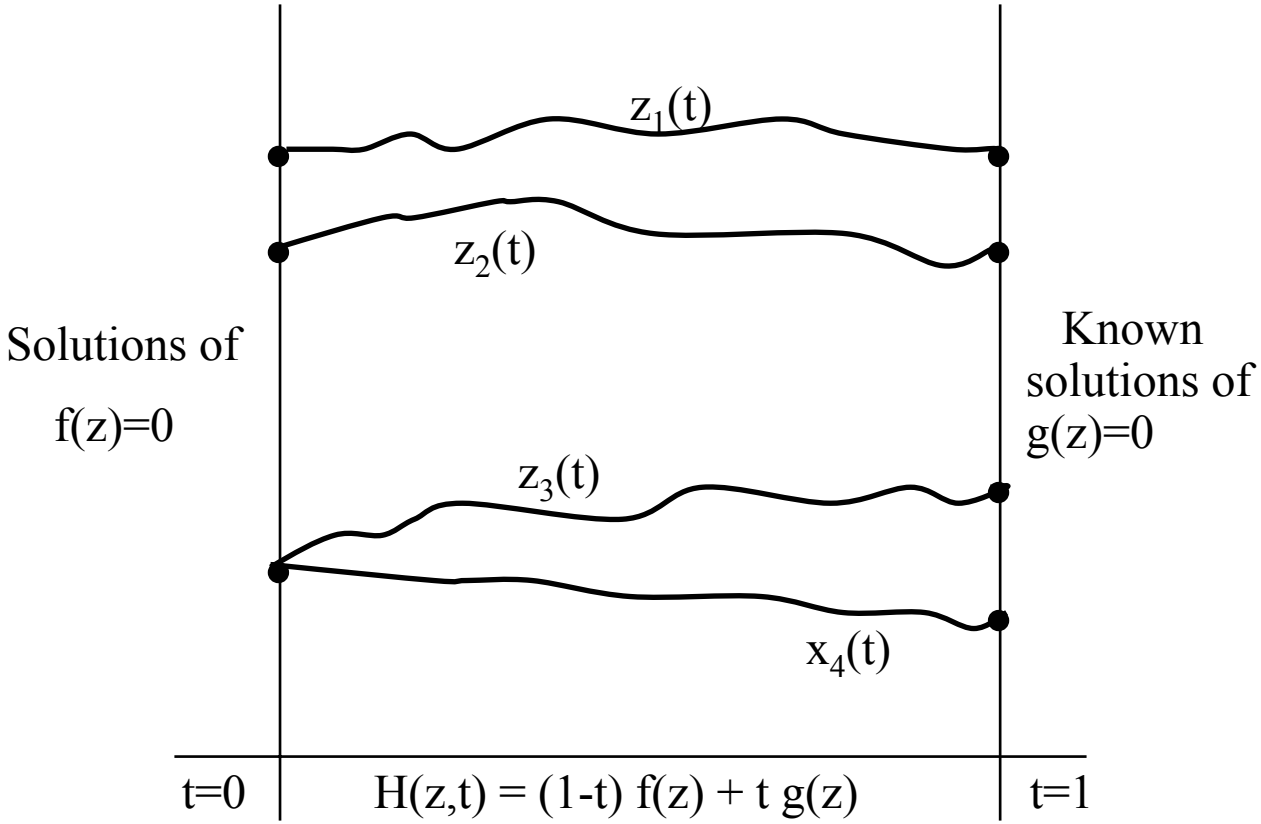
$$H(z,t) = (1-t)f(z) + tg(z).$$

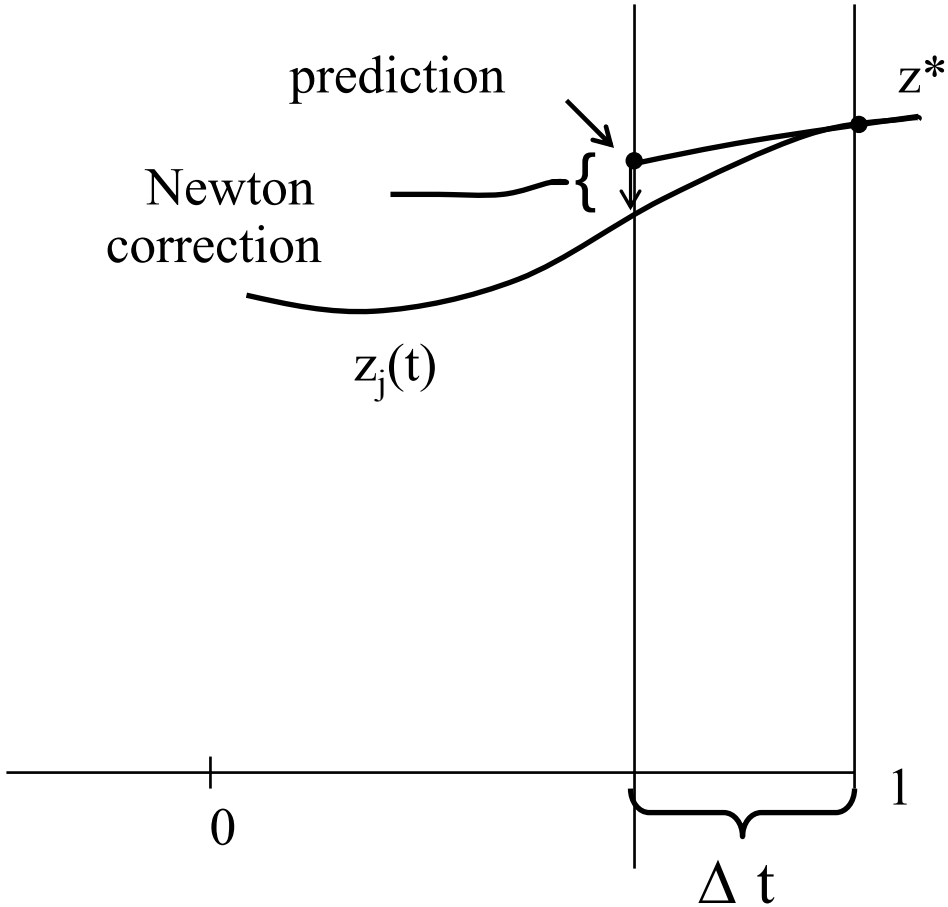
Hopefully, H(z,t) defines "paths" z(t) as t runs from 1 to 0. They start at known solutions of g(z) = 0 and end at the solutions of f(z) at t = 0.

- The paths satisfy the Davidenko equation

$$0 = \frac{dH(z(t),t)}{dt} = \sum_{i=1}^{N} \frac{\partial H}{\partial z_i} \frac{dz_i}{dt} + \frac{\partial H}{\partial t}$$

- To compute the paths: use ODE methods to predict and Newton's method to correct.


$z_1(t)$
$z_2(t)$
Solutions of
f(z)=0
Known
solutions of
g(z)=0
$z_3(t)$
$x_4(t)$
t=0
H(z,t) = (1-t) f(z) + t g(z)
t=1


prediction
z*
Newton
correction
$z_j(t)$
0
1
Δ t

A Guiding Principle Then

- Algorithms must be structured – when possible – to avoid paths leading to singular solutions: find a way to never follow the paths in the first place.
- Special Homotopies to take advantage of sparseness

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.

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.
  - 2006: about 10 paths a second on an single processor desktop CPU; 1000's of paths/second on moderately sized clusters.

Positive Dimensional Solution Sets

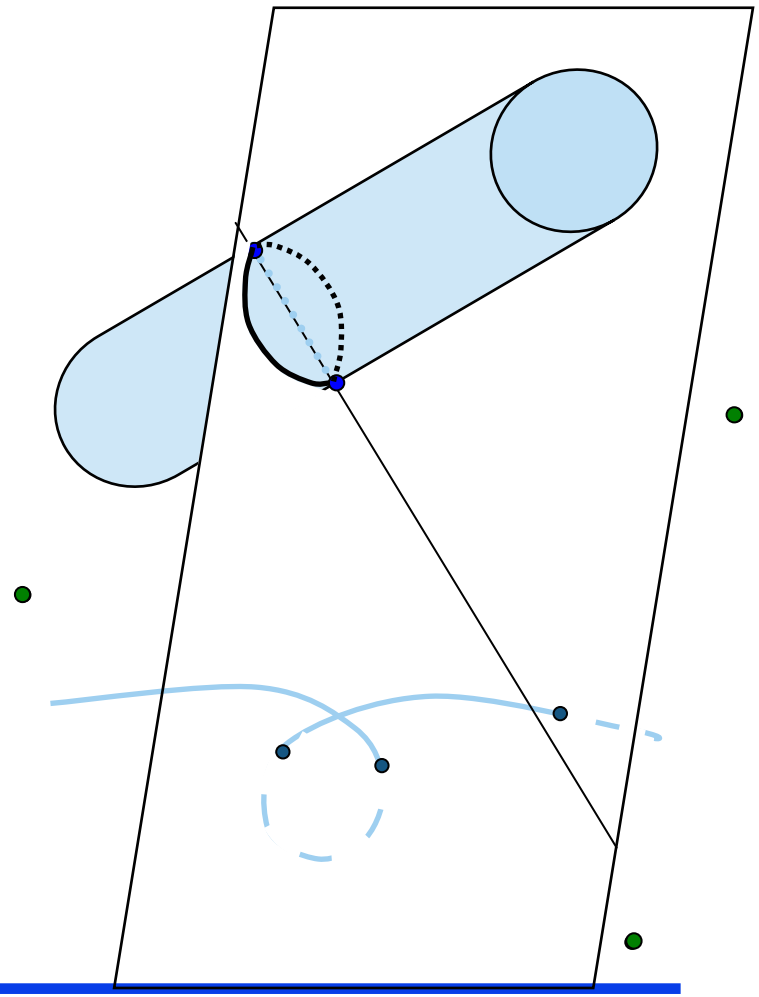
We now turn to finding the positive dimensional solution sets of a system

$$\begin{bmatrix} f_1(z_1,\ldots,z_N) \\ \vdots \\ f_n(z_1,\ldots,z_N) \end{bmatrix} = 0$$

Representing positive dimensional components

- 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



Numerical issues and Bertini

- Endgames and Deflation to deal with multiplicity greater than one components

- The need for adaptive precision

Numerical issues posed by multiple components

Consider a toy homotopy

$$H(z_1, z_2, t) = \begin{bmatrix} z_1^2 \\ z_2 - t \end{bmatrix} = 0$$

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

How do we deal with this?

Deflation for isolated singularities

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

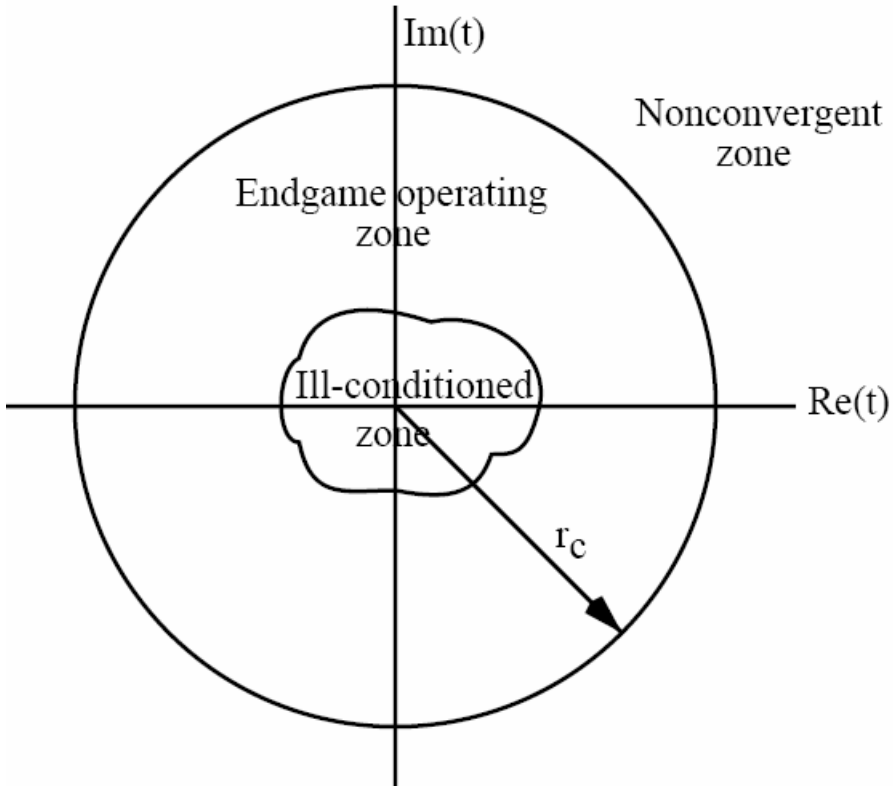
$$\begin{bmatrix} f(x) \\ Jf(z) \cdot \xi \\ A \cdot \xi + b \end{bmatrix} = 0$$

- 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.

Endgames (Morgan, Wampler, and S.)

- Example: $(z - 1)^2 - t = 0$

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 $s = 0$.



Bertini and the need for adaptive precision

- High precision was not practical in the 90's!
  - Highly nontrivial to design and dependent on hardware
  - Hardware too slow

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 is 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 = 27.999999999999. Evaluating p(a) to 15 digits, we find that

  p(a) = -0.057845595340 77.

- Even with 17 digit accuracy, the approximate root a is a = 27.999999999999905 and we still only have p(a) = -0.004953315573729 3.

Wilkinson's Theorem Numerical Linear Algebra

- Solving Az = 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 to the set defined by det(A) = 0.

Using Higher Precision

- 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.

Using Higher Precision

- 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.

This is computationally very expensive!

grxedn +85 elw,	97 elw	<9 elw	45; elw	589 elw	845 elw	4357 elw
5177:	651949	681789	681;5<	831663	:6133<	4571734

Wdedn 4= D yhudjh wlp h/ lq vhfrqgv xvlqj E huwlql/ ri 43 uxqv ri wkh F khe|vkhy sro|qrp ldo ri ghjuhh 43 z lwk @ ;>iru gl huhqw dhyhov ri {hg suhflvlrq1

Iurp E dwhv/ K dxhqvwhlq/ Vrp p hvh/ Z dp schu= D gdswlyh p xowlsuhflvlrq sdwk wudfnlqj1

Bertini

- Bertini uses data types modeled on the geometry and is designed to dynamically adjust the precision to achieve a solution with a prespecified error.
  - Bertini is being developed by Daniel Bates, Jon Hauenstein, Charles Wampler, and myself.

Issues

- You need to stay on the parameter space your problem is on.
- You need rules to decide when to change precision and by how much to change it.

Near-singular conditions actually arise. For the best known polynomial system to solve Alt's problem:

- Out of 143,360 paths:
  - 1184 paths (0.826%) used higher precision and then dropped back to double precision before starting the endgame
  - 680 paths (0.474%) used at least 96-bit precision and then dropped back to double precision before starting the endgame

Equation-by-Equation Methods

- Potential to solve systems with relatively few solutions that are completely outside of the beyond the pale of standard continuation methods
  - Intersection Method by Sommese, Verschelde, and Wampler
  - Regeneration Method by Hauenstein, Sommese, and Wampler

Basic Idea

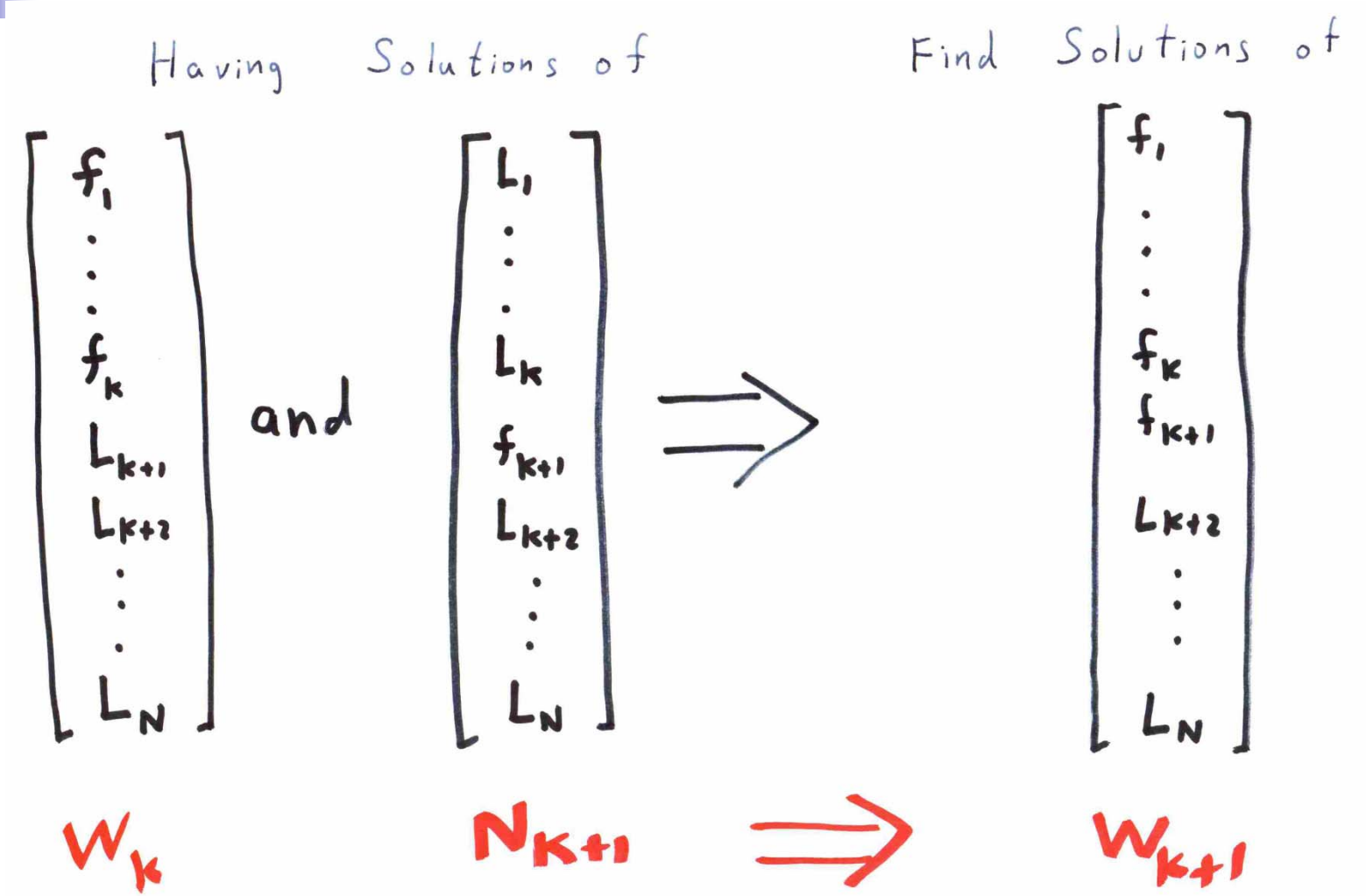
It is a natural and ancient approach to work *equation by equation.*

First Solve $f_1(x) = 0$;

then solve $\begin{bmatrix} f_1(z) \\ f_2(z) \end{bmatrix} = 0$;

then solve $\begin{bmatrix} f_1(z) \\ f_2(z) \\ f_3(z) \end{bmatrix} = 0$;**. . . and so on.**

Numerical Equation by Equation



Diagonal Approach of S., V., & W.

a) Restrict to the $\mathbb{C}^{k+1}$ defined by $L_{k+2}, \ldots, L_N$.

b) Double the variables to $(u, v) \in \mathbb{C}^{k+1} \times \mathbb{C}^{k+1}$

c) Use homotopy continuation starting with given data

$$\begin{bmatrix} f_1(u) \\ \vdots \\ f_k(u) \\ f_{k+1}(v) \\ L_1(u) \\ \vdots \\ L_k(u) \\ L_{k+1}(v) \end{bmatrix}$$

$W_k \times N_{k+1}$

going to

$$\begin{bmatrix} f_1(u) \\ \vdots \\ f_k(u) \\ f_{k+1}(v) \\ u_1 - v_1 \\ \vdots \\ u_k - v_k \\ u_{k+1} - v_{k+1} \end{bmatrix}$$

$\Delta(W_{k+1})$

The Intrinsic Diagonal Homotopy

Let $\phi : \mathbb{C}^{k+1} \longrightarrow \mathbb{C}^{2k+2}$ be a parameterization of the kernel of $L_1(u), \ldots, L_k(u), L_{k+1}(v)$

$w \longrightarrow (u, v)$

and $\Psi : w \longrightarrow (u, v) = (w, w)$ be a parameterization of the kernel of $u_1 - v_1, \ldots, u_{k+1} - v_{k+1}$

Let $\pi_1 : \mathbb{C}^{2k+2} \longrightarrow \mathbb{C}^k$

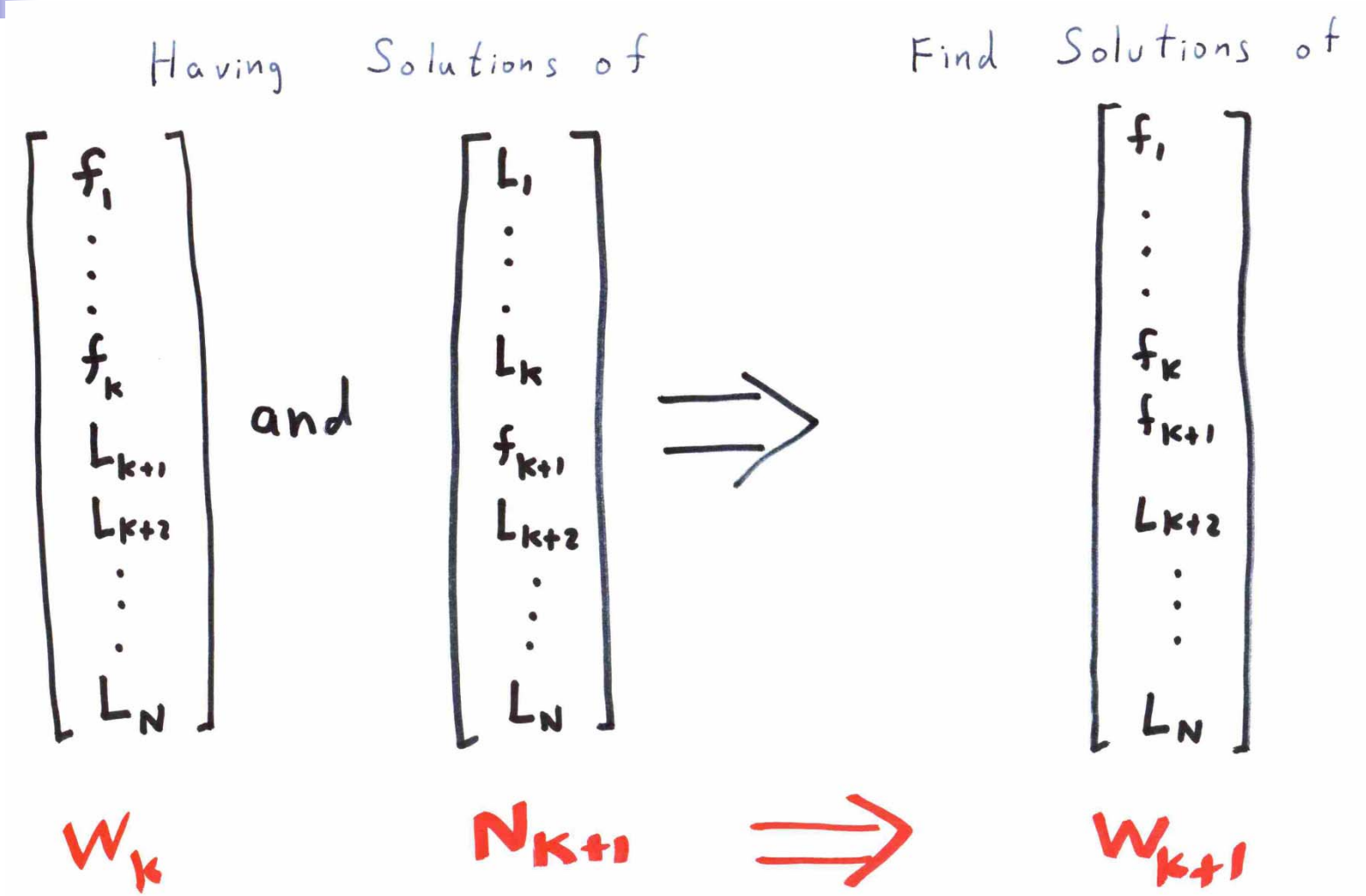
$(u, v) \longrightarrow u$

$\pi_2 : \mathbb{C}^{2k+2} \longrightarrow \mathbb{C}^k$

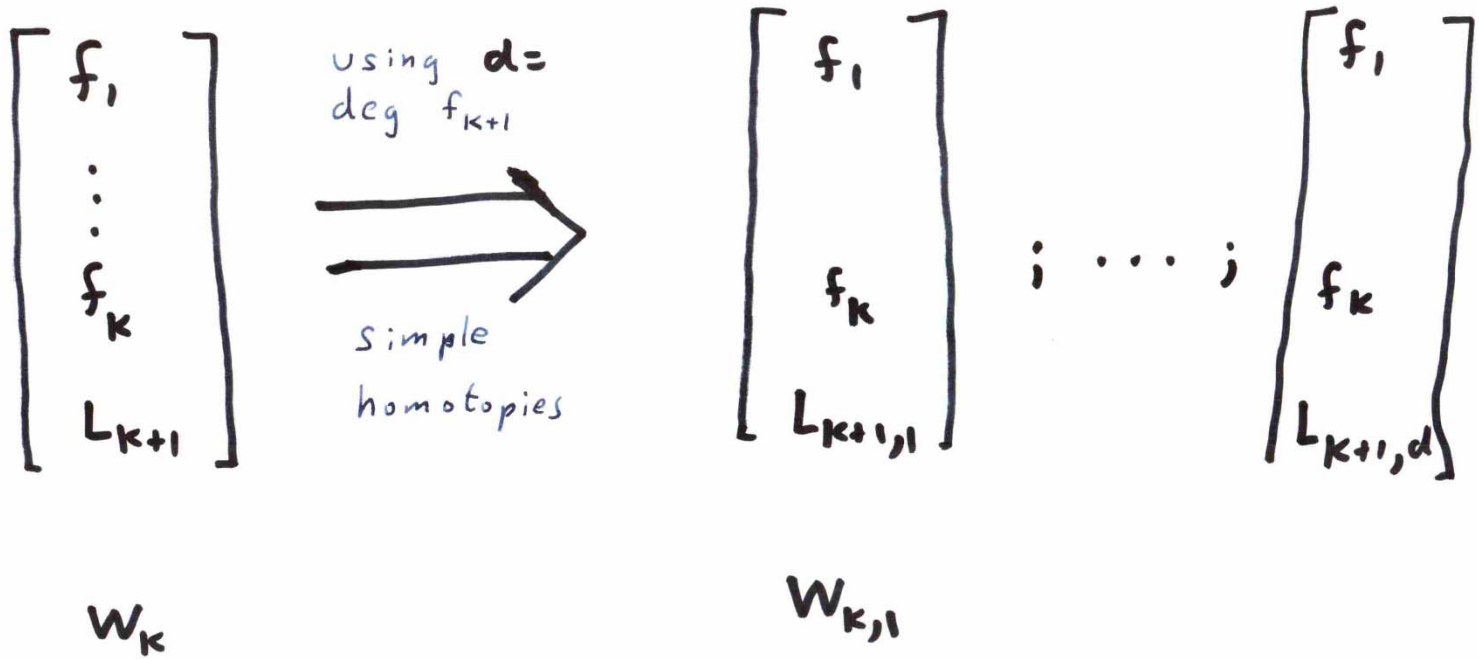
$(u, v) \longrightarrow v$

$$H(w,t) = \begin{bmatrix} f_1 \circ \pi_1 \\ \vdots \\ f_k \circ \pi_1 \\ f_{k+1} \circ \pi_2 \end{bmatrix} ((1-t)\Psi(w) + \gamma t \Phi(w)) = 0$$

Numerical Equation by Equation



Regeneration: Jon Hauenstein, S., &W.



where

$L_{k+1,1}, \ldots, L_{k+1,d}$ are random linear equations

Now use the homotopy

$t : 1 \longrightarrow 0$

$\overset{d}{\underset{i=1}{\cup}} W_{k,i} \Longrightarrow W_{k+1}$

$$H(z,t) = \begin{bmatrix} f_1(z) \\ \vdots \\ f_k(z) \\ (1-t)f_{k+1}(z) + \gamma t \Pi_{i=1}^{\deg f_{k+1}} L_{k+1,i}(z) \end{bmatrix}$$

Both methods generate a witness set for $f_1,..,f_{k+1}$

$$H(z,t) = \begin{bmatrix} f_1(z) \\ \vdots \\ f_k(z) \\ (1-t)f_{k+1}(z) + \gamma t \Pi_{i=1}^{\deg f_{k+1}} L_{k+1,i}(z) \end{bmatrix}$$

$$H(w,t) = \begin{bmatrix} f_1 \circ \pi_1 \\ \vdots \\ f_k \circ \pi_1 \\ f_{k+1} \circ \pi_2 \end{bmatrix} ((1-t)\Psi(w) + \gamma t \Phi(w))$$

Alt's System

$$\left[(\hat{a}-\bar{\delta}_j)x\right]\gamma_j+\left[(a-\delta_j)\hat{x}\right]\hat{\gamma}_j+\delta_j(\hat{a}-\hat{x})+\bar{\delta}_j(a-x)-\delta_j\bar{\delta}_j=0$$

$$\left[(\hat{b}-\bar{\delta}_j)y\right]\gamma_j+\left[(b-\delta_j)\hat{y}\right]\hat{\gamma}_j+\delta_j(\hat{b}-\hat{y})+\bar{\delta}_j(b-y)-\delta_j\bar{\delta}_j=0$$

$$\gamma_j+\hat{\gamma}_j+\gamma_j\hat{\gamma}_j=0$$

in the 24 variables $a, b, x, y, \hat{a}, \hat{b}, \hat{x}, \hat{y}$ and $\gamma_j, \hat{\gamma}_j$ 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.

Freudenstein-Roth system

Using Cramers rule and substitution we have what is essentially the Freudenstein-Roth system consisting of 8 equations of degree 7. In 1991, this was impractical to solve:

$7^8 = 5{,}764{,}801$ solutions.

Timings: Alt's Problem

Timings are by Jon Hauenstein

W lp lqjv duh rq rxu fdxvwhu U xqjh xvlqj rqh R swhurq 583 dv khdg qrgh dqg 47 R swhurq 583 surfhvvruv dv z runhu qrghv1

X vlqj wkh rqh0krp rjhqhxv 45 ixqfwlrq irup xodwlrq z lwk ghjuhh 4/37;/8:9/ lw wrrn

```
6817 krxuv +hvw1, xvlqj vwdqgdug sdwk wudfnlqj>dqg

417 krxuv xvlqj uhjhqhudwlrq +45</739 sdwkv,1

916 krxuv xvlqj wkh gldjrqdo dssurdfk +45</777 sdwkv,1
```

Further Timings: Alt's Problem

X vlqj wkh wz r0krp rjhqhxv 45 ixqfwlrq irup xodwlrq z lwk dq rughu wz r v|p p hwu|/z klfk lv wkh p rvwh flhqw irup xodwlrq iru wkh vwdqgdug krp rwrs| dssurdfk/

41586 krxuv xvlqj vwdqgdug sdwk wudfnlqj +476/693,>dqg

3165 krxuv xvlqj uhjhqhudwlrq +:6/883 sdwkv,1

More dramatic with the simpler system

X vlqj wkh vlp sdu fodvvlfdo|vwhp ri hljkw ghjuhh vhyhq htxdwlrqv +8/:97/;34 sdwkv,/lw wrrn

6:1< krxuv xvlqj vwdqgdug sdwk wudfnlqj +hvwlp dwhg e| xvlqj 43/333 udqgrp sdwkv,>dqg

51:8 krxuv xvlqj uhjhqhudwlrq +4:6968 sdwkv,1

Conclusions from Equation-by-Equation Runs

- Equation-by-equation methods may be much faster than using standard continuation

Conclusions from Equation-by-Equation Runs

- Equation-by-equation methods may be much faster than using standard continuation
- On a badly formulated system, equation-by-equation may dramatically beat standard continuation

Conclusions from Equation-by-Equation Runs

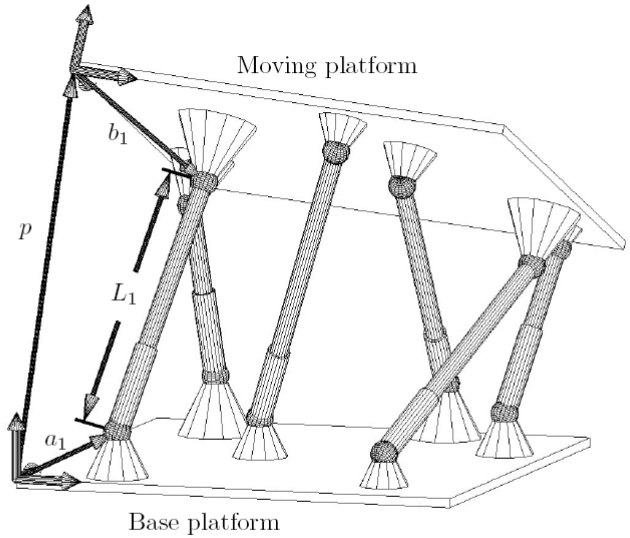
- Equation-by-equation methods may be much faster than using standard continuation
- On a badly formulated system, equation-by-equation may dramatically beat standard continuation
- Equation-by-equation methods used on badly formulated systems may be comparable to standard continuation used on optimally formulated systems

Diagonal versus Regeneration

- Regeneration can easily take advantage of special product structures of the equations.
- Diagonal homotopies allow us to intersect solution components of the same system, while regeneration does not.
- Regeneration is roughly twice as fast an equation-by-equation method as the diagonal method.

Overconstrained Mechanisms

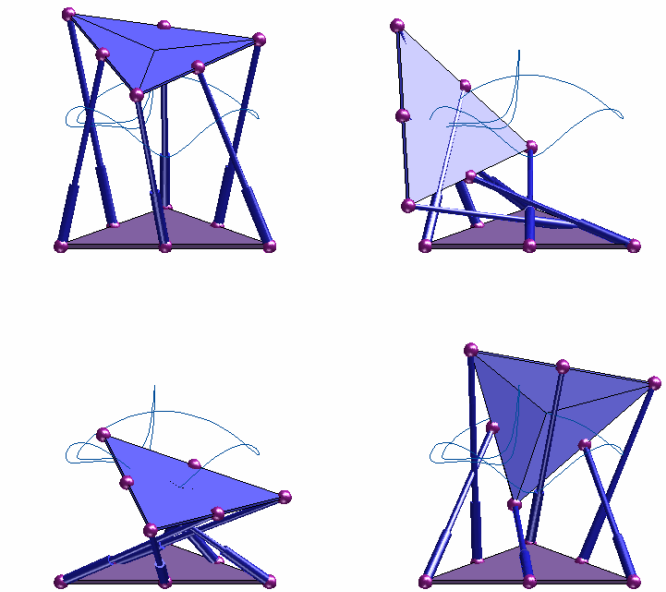
General platform robot



General Stewart-Gough platform robot: from Sommese & Wampler: *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*, World Scientific (2005).

Leg Lengths = Not Fixed

Special platform robot



Leg Lengths = Fixed

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
  - A.J. Sommese and C.W. Wampler, Exceptional sets and fiber products, to appear Foundations of Computational Mathematics.
- Equation-by-equation approach

Computing the geometric genus of a curve

- Bates, Chris Peterson, Sommese, Wampler
- Key observation: the geometric genus of a curve depends only on monodromy around images of singularities, which is easy to compute numerically.

Numerical approach by using Hurwitz's formula

- Given an irreducible curve X that is a component of V(f) of a polynomial system,

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

- X is represented by a generic hyperplane L = 0 plus W, the set of degree X points where L meets X.
- L = 0 is the fiber of a general projection p from C^N to C.

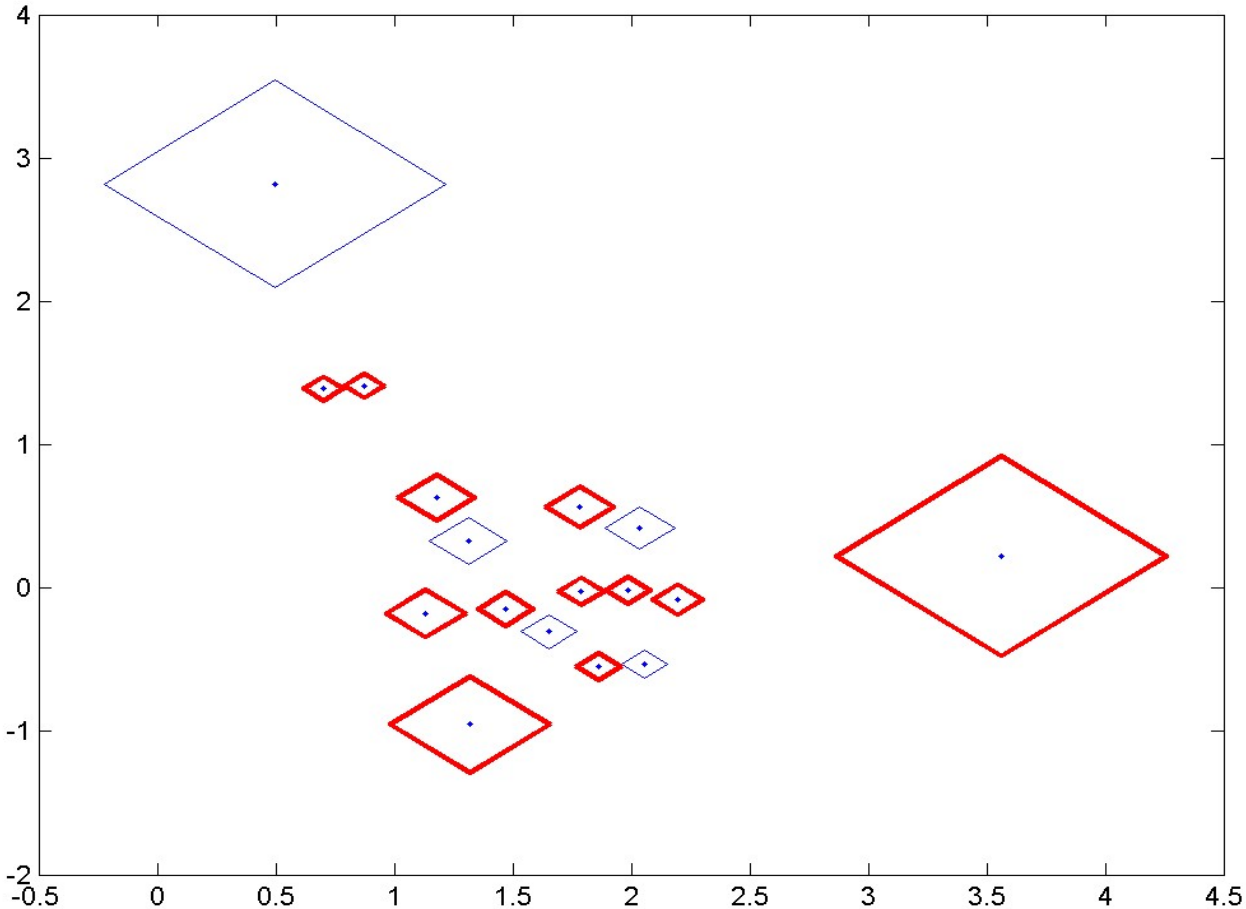
Use the Hurwitz formula

- $g = -\deg X + 1 + \rho/2$

A general planar four-bar coupler curve C

- C is a (3,3) curve in $P^1 \times P^1$. Arithmetic genus at most 4 with at least 3 singularities: so geometric genus is at most 1.
- Treating it as a degree 6 curve in P^2. We compute 30 potential branch points counting multiplicities:
  - 2 potential branch points with multiplicity 6 each make a local contribution 0 to ρ;
  - 3 potential branch points with multiplicity 2 each make a contribution of 0 to ρ;
  - 12 potential branch points contribute one each to ρ.

The geometric genus is 1



Algebraic vs. Geometric Approaches

	Pros	Cons
Algebraic/Symbolic	Highly developed	Parallelizes very poorly
	Can give exact answers	Very unstable dependence on inputs
	Not restricted to characteristic zero	Memory requirements grow rapidly with number of variables
Geometric/ Numerical Methods	Parallelizes almost perfectly	New--less developed than algebraic/symbolic approach
	Relatively stable dependence on inputs	
	Depending on # of digits used, gives answers reasonable for that accuracy	Certifiably exact answers are computationally very expensive
		Restricted to real and complex numbers

Summary

- Many Problems in Engineering and Science are naturally phrased as problems about algebraic sets and maps.
- Numerical analysis (using continuation) gives a geometric 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.