<u>Bertini</u>: A new software package for computations in numerical algebraic geometry

Dan Bates

University of Notre Dame

Andrew Sommese

University of Notre Dame

Charles Wampler

GM Research & Development

Workshop on Approximate Commutative Algebra - Special Semester on Gröbner Bases

RICAM/RISC Linz, Austria (not Vienna!) February 23, 2006

Front matter

Thank you to the organizers!

Front matter

Thank you to the organizers!

WARNING: I am not talking about polynomial GCDs, Grobner bases, nearest systems with certain structure, oil, etc. Also, I assume that we are starting with exact input.

Front matter

Thank you to the organizers!

WARNING: I am not talking about polynomial GCDs, Grobner bases, nearest systems with certain structure, oil, etc. Also, I assume that we are starting with exact input.

Question: Why would you pay attention?

Front matter

Thank you to the organizers!

WARNING: I am not talking about polynomial GCDs, Grobner bases, nearest systems with certain structure, oil, etc. Also, I assume that we are starting with exact input.

Question: Why would you pay attention?

Answer: We also mix algebra, geometry, and analysis, just in a different way (+ "Can he earn his book?" – today's theme!)

Front matter

NOTE: The main point today is software, not theory. More like a tutorial than a talk….

Front matter

NOTE: The main point today is software, not theory. More like a tutorial than a talk….

As a result, this talk covers a lot of topics, none of which are covered very deeply.

Today's goals

I. Describe what Bertini does.

II. Describe what Bertini will do soon.

III. Explain how to use Bertini.

IV. Describe a little about how Bertini works.

V. Describe a few of Bertini's successes.

But first, a little background….

Who is involved

Main team: - Andrew Sommese

- Charles Wampler

- myself

Who is involved

Main team:

- Andrew Sommese
- Charles Wampler
- myself

Some early work by Chris Monico (Texas Tech)

Who is involved

Main team: - Andrew Sommese

- Charles Wampler

- myself

Some early work by Chris Monico (Texas Tech)

Some algorithms developed in collaboration with Chris Peterson (Colorado State) and Gene Allgower (Colorado State)

Original intent of Bertini

Given a polynomial system,

$$\begin{cases} f_1(x_1, x_2, \ldots, x_N) \\ f_2(x_1, x_2, \ldots, x_N) \\ \vdots \\ f_N(x_1, x_2, \ldots, x_N), \end{cases}$$

find all isolated solutions and produce a catalog of the positive-dimensional irreducible components with at least one point on each component (the "numerical irreducible decomposition").

Numerical irreducible decomposition

Let Z denote the solution set of a system. Then there is a decomposition

$$Z = \bigcup_{i=0}^{dim Z} Z_i = \bigcup_{i=0}^{dim Z} \bigcup_{j \in I_i} Z_{ij}$$

where the Z_{ij} are the irreducible components.

Numerical irreducible decomposition

Let Z denote the solution set of a system. Then there is a decomposition

$$Z = \bigcup_{i=0}^{dim Z} Z_i = \bigcup_{i=0}^{dim Z} \bigcup_{j \in I_i} Z_{ij}$$

where the Z_{ij} are the irreducible components.

We want to find a set of points on each irreducible component, so we produce

$$W = \bigcup_{i=0}^{dim Z} W_i = \bigcup_{i=0}^{dim Z} \bigcup_{j \in I_i} W_{ij}$$

where W_{ij} is a set of points on Z_{ij}.

Why new software?

There are currently several software packages for solving polynomial systems numerically:

- PHC (Verschelde, etc.)
- HomLab (Wampler)
- PHoM (Kim, Kojima, etc.)
- Hompack (Watson)
- others?

Why new software?

- Had several ideas and new algorithms that we wanted to implement for proof of concept and testing.

Why new software?

- Had several ideas and new algorithms that we wanted to implement for proof of concept and testing.
- Had a few ideas for increasing efficiency in basic algorithms, too.

Why new software?

- Had several ideas and new algorithms that we wanted to implement for proof of concept and testing.
- Had a few ideas for increasing efficiency in basic algorithms, too.
- New software is a headache, but it was necessary.

New developments

1. Solving two-point boundary value problems – a fun application of homotopy continuation (with Allgower, Sommese, and Wampler)

New developments

1. Solving two-point boundary value problems – a fun application of homotopy continuation (with Allgower, Sommese, and Wampler)
2. Numeric-symbolic methods in algebraic geometry (with Peterson and Sommese)

New developments

1. Solving two-point boundary value problems – a fun application of homotopy continuation (with Allgower, Sommese, and Wampler)
2. Numeric-symbolic methods in algebraic geometry (with Peterson and Sommese)
3. Methods in real algebraic geometry (with Ye Lu, Sommese, and Wampler)

New developments

1. Solving two-point boundary value problems – a fun application of homotopy continuation (with Allgower, Sommese, and Wampler)
2. Numeric-symbolic methods in algebraic geometry (with Peterson and Sommese)
3. Methods in real algebraic geometry (with Ye Lu, Sommese, and Wampler)
4. Moving from a personal tool for experimentation towards public-use software

How to get Bertini

- You can’t (yet).

How to get Bertini

- You can’t (yet). Bertini 1.0 soon released in executable format (probably) after more testing.

How to get Bertini

- You can’t (yet). Bertini 1.0 soon released in executable format (probably) after more testing.

- Available from my website (maybe) and Sommese’s website (definitely).

How to get Bertini

- You can't (yet). Bertini 1.0 soon released in executable format (probably) after more testing.

- Available from my website (maybe) and Sommese's website (definitely).

- Built/tested on Linux (Redhat, debian, SUSE, and Cygwin). Eventually available for Mac and Windows (already on Cygwin for Windows).

How to get Bertini

- You can't (yet). Bertini 1.0 soon released in executable format (probably) after more testing.

- Available from my website (maybe) and Sommese's website (definitely).

- Built/tested on Linux (Redhat, debian, SUSE, and Cygwin). Eventually available for Mac and Windows (already on Cygwin for Windows).

- Uses gcc, GMP/MPFR, flex/bison, maybe other libraries. See website once released.

Today's goals

I. Describe what Bertini does.

II. Describe what Bertini will do soon.

III. Explain how to use Bertini.

IV. Describe a little about how Bertini works.

V. Describe a few of Bertini's successes.

Today's goals

I. **Describe what Bertini does.**

II. Describe what Bertini will do soon.

III. Explain how to use Bertini.

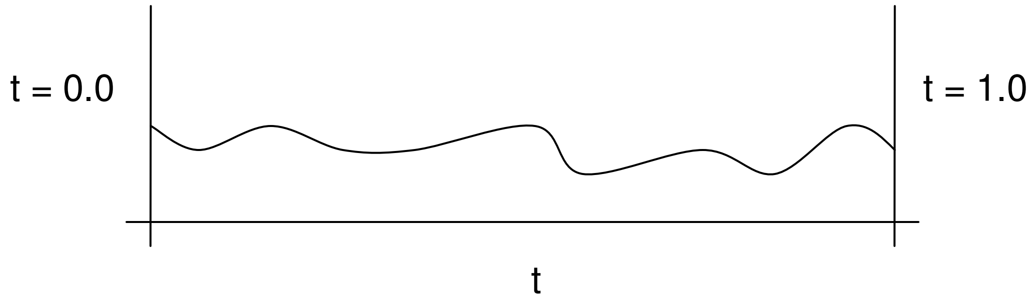
IV. Describe a little about how Bertini works.

V. Describe a few of Bertini's successes.

I. What Bertini does

A. Solving polynomial systems

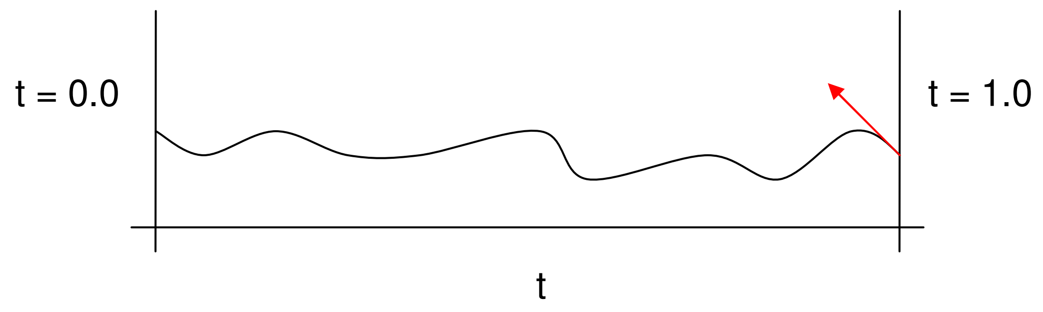
1. Uses predictor/corrector methods (homotopy continuation) to produce all isolated solutions of the given polynomial system.



I. What Bertini does

A. Solving polynomial systems

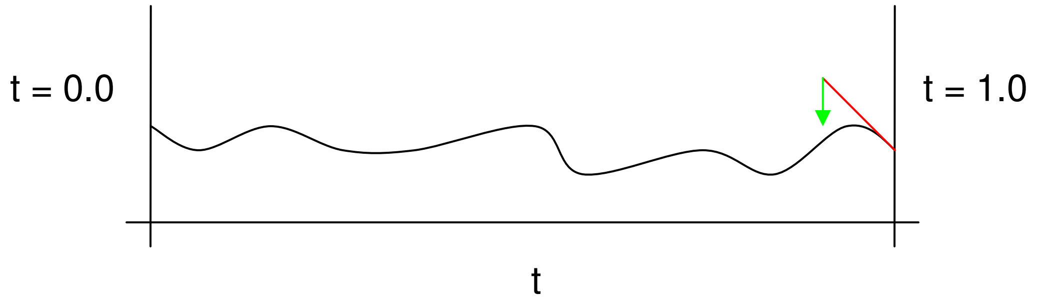
1. Uses predictor/corrector methods (homotopy continuation) to produce all isolated solutions of the given polynomial system.



I. What Bertini does

A. Solving polynomial systems

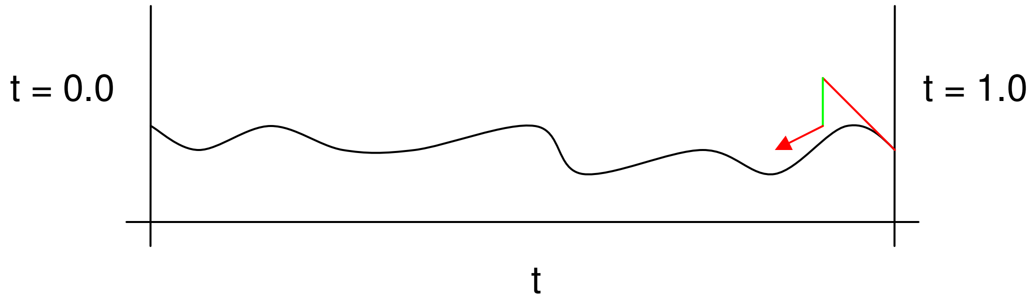
1. Uses predictor/corrector methods (homotopy continuation) to produce all isolated solutions of the given polynomial system.



I. What Bertini does

A. Solving polynomial systems

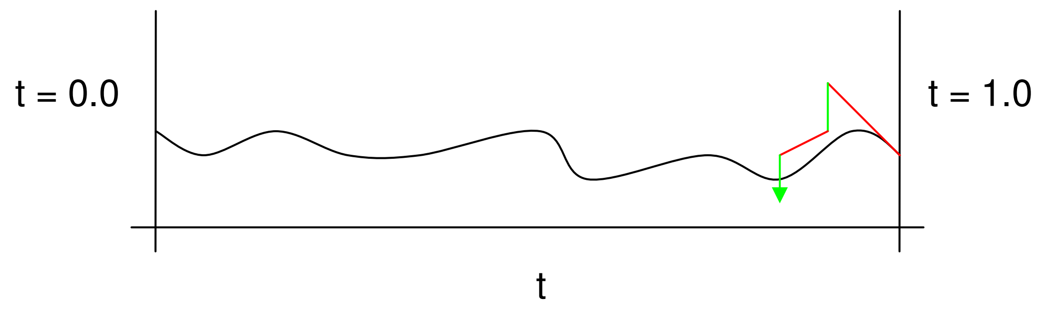
1. Uses predictor/corrector methods (homotopy continuation) to produce all isolated solutions of the given polynomial system.



I. What Bertini does

A. Solving polynomial systems

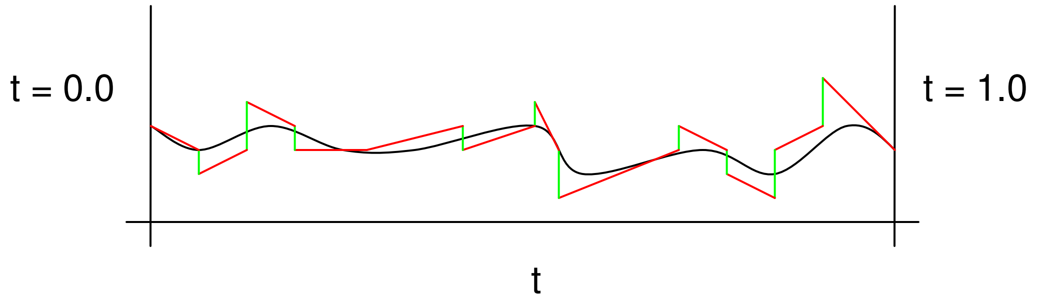
1. Uses predictor/corrector methods (homotopy continuation) to produce all isolated solutions of the given polynomial system.



I. What Bertini does

A. Solving polynomial systems

1. Uses predictor/corrector methods (homotopy continuation) to produce all isolated solutions of the given polynomial system.



I. What Bertini does

A. Solving polynomial systems

2. Automatic m-homogenization and generation of m-homogeneous start systems.

I. What Bertini does

A. Solving polynomial systems

2. Automatic m-homogenization and generation of m-homogeneous start systems.

$$\begin{tikzcd}
\boxed{\begin{array}{l} \frac{29}{16}z_1^3 - 2z_1z_2 \\ z_2 - z_1^2 \end{array}} \arrow[r] & \boxed{\begin{array}{l} \frac{29}{16}z_1^3 - 2z_1z_2H_1 \\ z_2H_1 - z_1^2 \end{array}} \arrow[d] \\
& \boxed{\begin{array}{l} (\frac{29}{16}z_1^3 - 2z_1z_2H_1)(1-s) + sg_1 \\ (z_2H_1 - z_1^2)(1-s) + sg_2 \end{array}}
\end{tikzcd}$$

I. What Bertini does

A. Solving polynomial systems

2. Automatic m-homogenization and generation of m-homogeneous start systems.

$$\begin{tikzcd}
\boxed{\begin{array}{l} \frac{29}{16}z_1^3 - 2z_1z_2 \\ z_2 - z_1^2 \end{array}} \arrow[r] & \boxed{\begin{array}{l} \frac{29}{16}z_1^3 - 2z_1z_2H_1 \\ z_2H_1 - z_1^2 \end{array}} \arrow[dl] \\
\boxed{\begin{array}{l} (\frac{29}{16}z_1^3 - 2z_1z_2H_1)(1-s) + sg_1 \\ (z_2H_1 - z_1^2)(1-s) + sg_2 \end{array}} &
\end{tikzcd}$$

Bertini produces start solutions also.

I. What Bertini does

A. Solving polynomial systems

3. Several endgames (including adaptive precision versions of a couple).

I. What Bertini does

A. Solving polynomial systems

4. Multiple precision, using MPFR.

I. What Bertini does

A. Solving polynomial systems

4. Multiple precision, using MPFR.

NOTE: Extra digits aren’t cheap!

I. What Bertini does

A. Solving polynomial systems

4. Multiple precision, using MPFR.

NOTE: Extra digits aren't cheap!

5. Adaptive multiprecision: [key advance]
Bertini (if set to do so) will change precision only when necessary, i.e., when certain inequalities are violated.

I. What Bertini does

A. Solving polynomial systems

6. Witness point sets for positive-dimensional components:

- Cascade algorithm to get witness supersets

- Perform "junk removal"

- Pure-dimensional decomposition into irreducible components (monodromy + linear traces)

I. What Bertini does

B. Two-point boundary value problems

Input: The (polynomial) nonlinearity (f), and the boundary values, i.e.

$$y'' = f(x, y, y')$$
$$y(a) = \alpha$$
$$y(b) = \beta$$

Output: An approximation of all solutions, given the desired mesh size.

An analytic problem may be solved with algebra.

I. What Bertini does

B. Two-point boundary value problems

Basic idea of the algorithm:

1. Discretize using N mesh points, yielding a polynomial system.
2. Move from N to N+1 using homotopy continuation.
3. Repeat.

I. What Bertini does

B. Two-point boundary value problems

NOTE: Runs in conjunction with Maple. In fact, Maple is where most computation takes place – it calls Bertini for path-tracking.

I. What Bertini does

B. Two-point boundary value problems

NOTE: Runs in conjunction with Maple. In fact, Maple is where most computation takes place – it calls Bertini for path-tracking.

(This is how we deal with new ideas.)

I. What Bertini does

B. Two-point boundary value problems

NOTE: Runs in conjunction with Maple. In fact, Maple is where most computation takes place – it calls Bertini for path-tracking.

(This is how we deal with new ideas.)

For details, see:
Allgower, B., Sommese, & Wampler. Solution of polynomial systems derived from differential equations. *Computing*, 76(1-2): 1-10, 2006.

I. What Bertini does

C. Computing real curves

Input: Polynomial system suspected of having a real curve as a solution component

Output: Description of all real curves, in the form of a set of points on each curve including certain projection-specific critical points. These points convey certain characteristics about the curve….

I. What Bertini does

C. Computing real curves

Input: Polynomial system suspected of having a real curve as a solution component

Output: Description of all real curves, in the form of a set of points on each curve including certain projection-specific critical points. These points convey certain characteristics about the curve….

[not my story to tell]

I. What Bertini does

C. Computing real curves

For details, please refer to:

Y. Lu, Sommese, & Wampler. Finding all real solutions of polynomial systems: I The curve case, *in preparation*.

I. What Bertini does

C. Computing real curves

For details, please refer to:

Y. Lu, Sommese, & Wampler. Finding all real solutions of polynomial systems: I The curve case, *in preparation*.

NOTE: This implementation also works in conjunction with Maple.

I. What Bertini does

D. Multiplicity and regularity of a 0-scheme

Input: Polynomial system with 0-dimensional solution set (or higher-dimensional + slicing)

Output: Multiplicity and other data as with Zeng's talk on Tuesday + a bound on the (Castelnuovo-Mumford) regularity.

I. What Bertini does

D. Multiplicity and regularity of a 0-scheme

Input: Polynomial system with 0-dimensional solution set (or higher-dimensional + slicing)

Output: Multiplicity and other data as with Zeng's talk on Tuesday + a bound on the (Castelnuovo-Mumford) regularity.

Our method is related to Dayton & Zeng's method, although the approach is a little different. Here's a sketch:

I. What Bertini does

D. Multiplicity and regularity of a 0-scheme

For k from 1 until done:

- Form a certain ideal based on k.

I. What Bertini does

D. Multiplicity and regularity of a 0-scheme

For k from 1 until done:

- Form a certain ideal based on k.
- Saturate the ideal (involves computing certain spans of polynomials and intersecting ideals (numerically)).

I. What Bertini does

D. Multiplicity and regularity of a 0-scheme

For k from 1 until done:

- Form a certain ideal based on k.

- Saturate the ideal (involves computing certain spans of polynomials and intersecting ideals (numerically)).

- Compute two numerical ranks – if they agree, you are done, with k bounding reg(I).

I. What Bertini does

D. Multiplicity and regularity of a 0-scheme

From the regularity, the multiplicity is trivial to compute (just some little formula).

I. What Bertini does

D. Multiplicity and regularity of a 0-scheme

From the regularity, the multiplicity is trivial to compute (just some little formula).

For more details, please see:
B., Peterson, & Sommese. A numeric-symbolic algorithm for computing the multiplicity of a component of an algebraic set, *submitted*.

Today's goals

I. Describe what Bertini does.

II. Describe what Bertini will do soon.

III. Explain how to use Bertini.

IV. Describe a little about how Bertini works.

V. Describe a few of Bertini's successes.

II. What Bertini will do

New types of start systems and points

- Automatic total degree start systems for zero-dimensional solving

- Automatic m-homogenization and m-homogeneous start systems for positive-dimensional solving

- Other easy start systems, e.g., linear product

- Polytope-based start systems – does anybody want to share?

II. What Bertini will do

Parallel computation

Andrew’s group has a new cluster, purchased specifically for a parallel version of Bertini.

A new student in the group, Jon Hauenstein, is working on parallelization.

II. What Bertini will do

Advanced algorithms for polynomial systems

(from Andrew’s talk last night)

- Intersection algorithm

- Exceptional fibers algorithm

- Equation by equation algorithm

- New forms of basic path-tracking

II. What Bertini will do

Deflation

Neat Idea: Make a known singular point nonsingular by adding certain derivatives to the system (see paper by Leykin, Verschelde, Zhao).

II. What Bertini will do

Deflation

Neat Idea: Make a known singular point nonsingular by adding certain derivatives to the system (see paper by Leykin, Verschelde, Zhao).

Currently, all derivatives are added at each stage of deflation, so the size of the system increases by a factor of 2^m where m=multiplicity.

II. What Bertini will do

Deflation

Neat Idea: Make a known singular point nonsingular by adding certain derivatives to the system (see paper by Leykin, Verschelde, Zhao).

Currently, all derivatives are added at each stage of deflation, so the size of the system increases by a factor of 2^m where m=multiplicity.

With Lu, Sommese, & Wampler: Considering more efficient methods and an algorithm for tracking along multiple components.

II. What Bertini will do

More real algebraic geometry

There is already an algorithm for real surfaces, just like the curve case. It just needs to be implemented.

II. What Bertini will do

More real algebraic geometry

There is already an algorithm for real surfaces, just like the curve case. It just needs to be implemented.

Related to the concept of a “roadmap.”

II. What Bertini will do

More computational algebraic geometry

Chris Peterson will talk about this a little more.

II. What Bertini will do

More computational algebraic geometry

Chris Peterson will talk about this a little more.

Ideas include numerical syzygy modules, numerical free resolutions, etc.

II. What Bertini will do

More computational algebraic geometry

Chris Peterson will talk about this a little more.

Ideas include numerical syzygy modules, numerical free resolutions, etc.

Key idea: Using different levels of precision, one can detect which singular values are actually 0!

<u>II. What Bertini will do</u>

<u>Interactive version and/or scripting language</u>

We love this idea, but we aren’t there yet.

II. What Bertini will do

Interactive version and/or scripting language

We love this idea, but we aren’t there yet.

We envy where CoCoA is now! CoCoA is where I dream of taking Bertini eventually….

Today’s goals

I. Describe what Bertini does.

II. Describe what Bertini will do soon.

III. Explain how to use Bertini.

IV. Describe a little about how Bertini works.

V. Describe a few of Bertini’s successes.

III. How to use Bertini

Bertini needs three files from the user in order to solve a polynomial system (not as nifty as CoCoA yet!):

III. How to use Bertini

Bertini needs three files from the user in order to solve a polynomial system (not as nifty as CoCoA yet!):

1. “input” contains the target polynomial system or homotopy

III. How to use Bertini

Bertini needs three files from the user in order to solve a polynomial system (not as nifty as CoCoA yet!):

1. “input” contains the target polynomial system or homotopy
2. “config” contains many important settings

III. How to use Bertini

Bertini needs three files from the user in order to solve a polynomial system (not as nifty as CoCoA yet!):

1. “input” contains the target polynomial system or homotopy

2. “config” contains many important settings

3. “start” contains a set of start points, and it is sometimes generated automatically

III. How to use Bertini

Syntax for "input" (specifying target only):

```
mhom 2;
variable_group z1;
variable_group z2;
function f1, f2;
pathvariable t;
f1 = (29/16)*z1^3-2*z1*z2;
f2 = z2-z1^2;
END;
```

III. How to use Bertini

Syntax for “config”:

```
…
0       < machine prec (0), multiprec (1), adaptive multiprec (2).
96      < precision (in bits). (64 -> 19 digits, 96 -> 28, 128 -> 38)
0       < output to screen?  1 for yes, 0 for no.
0       < output level, between -1 (minimal) and 3 (maximal).
3       < # of consecutive successful steps for increasing step size.
3       < maximum number of Newton iterations.
0.1     < maximum step size.
1e-6    < Newton tolerance until endgame.
1e-9    < Newton tolerance for endgame.
1e5     < Newton residual for declaring path at infinity.
0.1     < Beginning of end game range.
0       < final path variable value desired.
10000   < Max number of steps allowed per path.
1       < Endgame number
….
```

III. How to use Bertini

Syntax for "start" (if you need to write it):

For two starting points, (1, -2i) and (3+i, -0.5+i), type:

```
2

1.0 0.0;
0.0 -2.0;

3.0 1.0;
-0.5 1.0;
```

III. How to use Bertini

How to run Bertini:

1. Type “make” to create the executable.

III. How to use Bertini

How to run Bertini:

1. Type “make” to create the executable.
2. Type “./bertini” for zero-dimesional tracking.

III. How to use Bertini

How to run Bertini:

1. Type “make” to create the executable.
2. Type “./bertini” for zero-dimesional tracking.
3. Type “./bertini –c” for positive-dimensional tracking (c is for cascade).

III. How to use Bertini

How to run Bertini:

1. Type “make” to create the executable.
2. Type “./bertini” for zero-dimesional tracking.
3. Type “./bertini –c” for positive-dimensional tracking (c is for cascade).
4. Find the results in “output”, “refined_solutions”, or “cascade_output” (or double check your files in case of an error).

III. How to use Bertini

Output format (for polynomial system solving):

- “output” contains lots of path data (as much as the user requests) and the endpoint for each path.

III. How to use Bertini

Output format (for polynomial system solving):

- “output” contains lots of path data (as much as the user requests) and the endpoint for each path.

- “refined_solutions” gives the vital data for each endpoint and lists points that agree up to a user-defined tolerance together.

III. How to use Bertini

Output format (for polynomial system solving):

- "output" contains lots of path data (as much as the user requests) and the endpoint for each path.

- "refined_solutions" gives the vital data for each endpoint and lists points that agree up to a user-defined tolerance together.

- "cascade_output" gives a catalog of witness points.

III. How to use Bertini

For the multiplicity project, input is similar:

```
VARS x, y, z;
POINT 0.0, 0.0, 1.0;
x^4 + 2.0*x^2*y^2 + y^4 + 3.0*x^2*y*z – y^3*z;
x^6 + 3.0*x^4*y^2+3.0*x^2*y^4 + y^6 – 4.0*x^2*y^2*z^2;
END;
```

III. How to use Bertini

For the multiplicity project, input is similar:

```
VARS x, y, z;
POINT 0.0, 0.0, 1.0;
x^4 + 2.0*x^2*y^2 + y^4 + 3.0*x^2*y*z – y^3*z;
x^6 + 3.0*x^4*y^2+3.0*x^2*y^4 + y^6 – 4.0*x^2*y^2*z^2;
END;
```

Bertini then prints the output directly to the screen, with the bottom line of the form

```
Multiplicity = 14, Regularity = 8
```

III. How to use Bertini

Warning: All syntax is subject to change!

III. How to use Bertini

Warning: All syntax is subject to change!

Bottom line: Read the manual and see the examples when you download it.

<u>Today's goals</u>

I. Describe what Bertini does.

II. Describe what Bertini will do soon.

III. Explain how to use Bertini.

IV. <u>Describe a little about how Bertini works.</u>

V. Describe a few of Bertini's successes.

IV. How Bertini works

Straight-line programs

Example: `f(x,y,z) = 2.0*z^3 - 4.1*x*y`

IV. How Bertini works

Straight-line programs

Example: `f(x,y,z) = 2.0*z^3 - 4.1*x*y`

Store the constants in an array:

0	1	2	3	4	5	6
2.0	3	4.1				
			x	y	z	f

IV. How Bertini works

Straight-line programs

Example: `f(x,y,z) = 2.0*z^3 - 4.1*x*y`

Store the constants in an array:

0	1	2	3	4	5	6
2.0	3	4.1				
			x	y	z	f

Then write evaluation instructions (with lex/yacc):

$7 = 5 \text{ ^ } 1$ $\qquad 10 = 2 * 9$

$8 = 0 * 7$ $\qquad 11 = 8 - 10$

$9 = 3 * 4$ $\qquad 6 = 11$

IV. How Bertini works

Advantages of straight-line programs

- Allows for subfunctions (great for symmetry!).

IV. How Bertini works

Advantages of straight-line programs

- Allows for subfunctions (great for symmetry!).

- Homogenization is easy for SLPs.

IV. How Bertini works

Advantages of straight-line programs

- Allows for subfunctions (great for symmetry!).

- Homogenization is easy for SLPs.

- Efficient (0.1% of total CPU time).

IV. How Bertini works

Advantages of straight-line programs

- Allows for subfunctions (great for symmetry!).

- Homogenization is easy for SLPs.

- Efficient (0.1% of total CPU time).

- Flexible (polynomials can be in factored form or in a format for Horner's method).

IV. How Bertini works

Advantages of straight-line programs

- Allows for subfunctions (great for symmetry!).

- Homogenization is easy for SLPs.

- Efficient (0.1% of total CPU time).

- Flexible (polynomials can be in factored form or in a format for Horner's method).

- Automatic differentiation is simple.

IV. How Bertini works

Multiplicity project

Requires another representation of polynomials – they are represented by vectors with a fixed monomial basis in each degree (à la Kreuzer).

IV. How Bertini works

Multiplicity project

Requires another representation of polynomials – they are represented by vectors with a fixed monomial basis in each degree (à la Kreuzer).

This project involves various special symbolic actions (e.g., polynomial arithmetic, expanding a polynomial to a higher degree) and numeric actions (e.g., computing numerical ranks).

IV. How Bertini works

Adaptive precision

Why bother?

Jacobian matrices become ill-conditioned near singularities.

IV. How Bertini works

Adaptive precision

Why bother?

Jacobian matrices become ill-conditioned near singularities.

Old idea: Increase precision if a path fails.

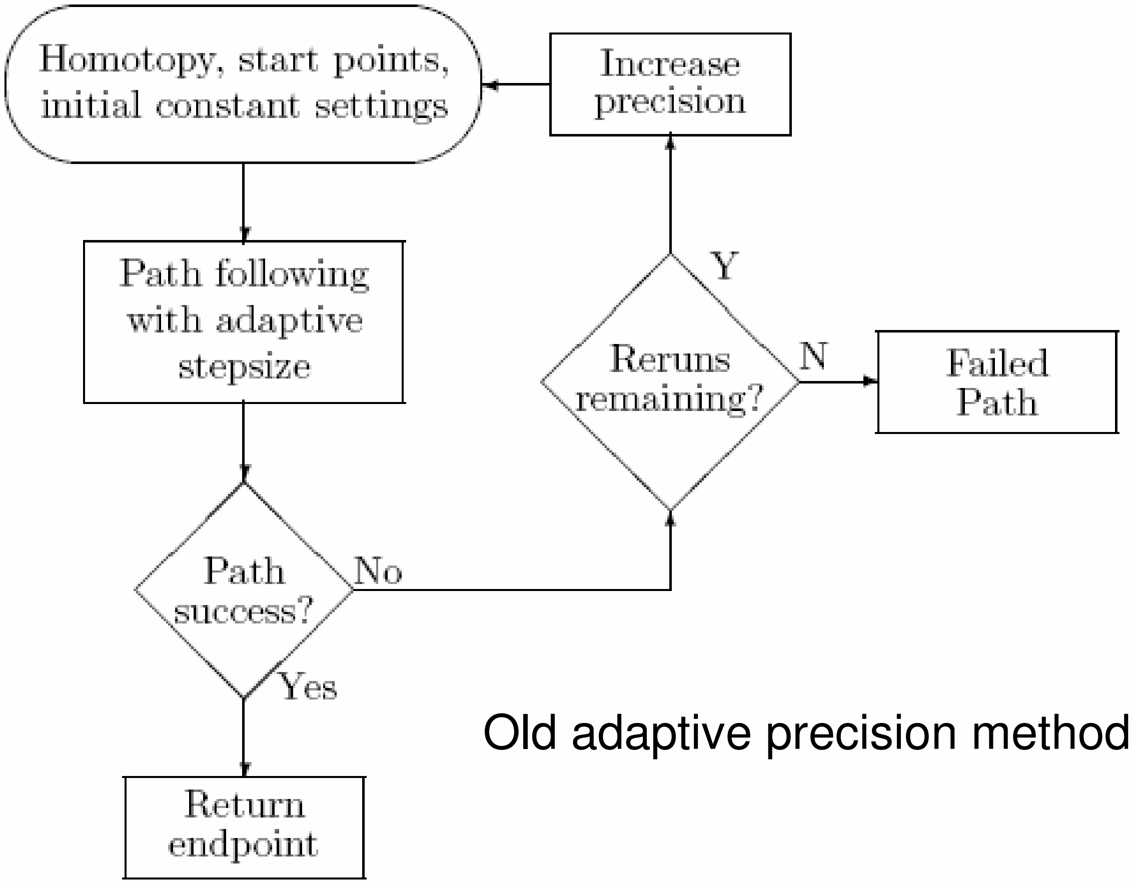
IV. How Bertini works

Adaptive precision

Why bother?

Jacobian matrices become ill-conditioned near singularities.

Old idea: Increase precision if a path fails.


Homotopy, start points, initial constant settings
Increase precision
Path following with adaptive stepsize
Y
Reruns remaining?
N
Failed Path
Path success?
No
Yes
Return endpoint

Old adaptive precision method

IV. How Bertini works

Adaptive precision

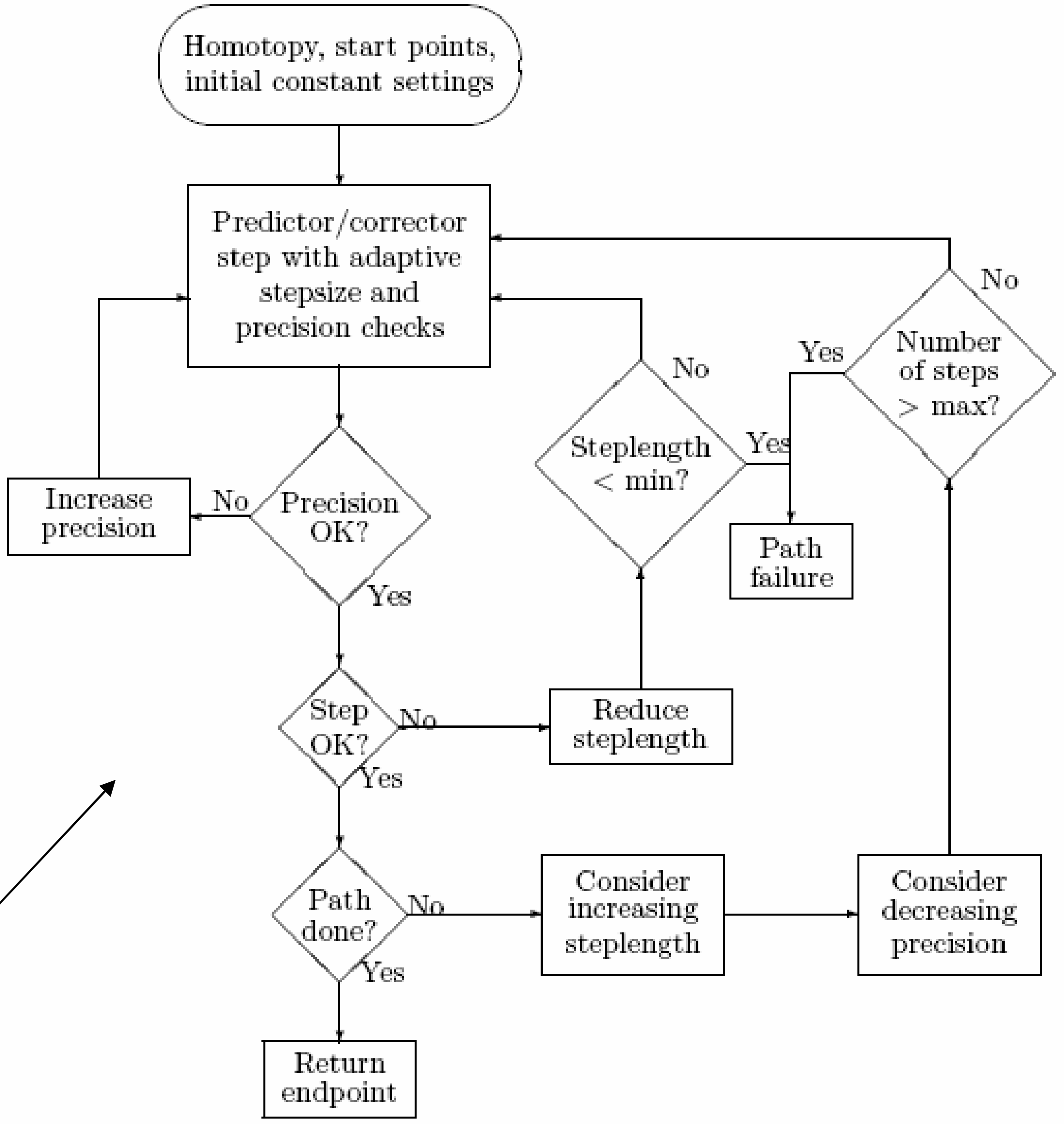
New Idea: Change precision on the fly as needed. Must detect when it is needed. Takes the form of a set of inequalities.

IV. How Bertini works

Adaptive precision

New Idea: Change precision on the fly as needed. Must detect when it is needed. Takes the form of a set of inequalities.

New adaptive precision method



Homotopy, start points, initial constant settings
Predictor/corrector step with adaptive stepsize and precision checks
Increase precision
No
Precision OK?
Yes
Step OK?
No
Yes
Reduce steplength
Steplength < min?
No
Yes
Path failure
Yes
Number of steps > max?
No
Path done?
No
Yes
Consider increasing steplength
Consider decreasing precision
Return endpoint

<u>Today's goals</u>

I. Describe what Bertini does.

II. Describe what Bertini will do soon.

III. Explain how to use Bertini.

IV. Describe a little about how Bertini works.

V. <u>Describe a few of Bertini's successes.</u>

V. Some successes

Solving polynomial systems

- Sym5Alt2 system: Medium-sized (12x12) polynomial system. Found exactly the 78 pairs of solutions out of several thousand paths.

<u>V. Some successes</u>

<u>Solving polynomial systems</u>

- <u>Sym5Alt2 system</u>: Medium-sized (12x12) polynomial system. Found exactly the 78 pairs of solutions out of several thousand paths.

- <u>From the BVP project</u>: Tracked paths of a sparse 100x100 system.

V. Some successes

Solving polynomial systems

- Sym5Alt2 system: Medium-sized (12x12) polynomial system. Found exactly the 78 pairs of solutions out of several thousand paths.

- From the BVP project: Tracked paths of a sparse 100x100 system.

- Wilkinson polynomial: (the product of (x-i) for i from 1 to 20, then perturbed) – Using adaptive precision, we confirmed the roots listed in Wilkinson's book.

V. Some successes

The Bratu two-point BVP

$$y'' = -\lambda e^y, \ \lambda > 0$$
$$y(0) = 0$$
$$y(1) = 0$$

Fact: There are no solutions for λ near 0. Otherwise, there are two.

V. Some successes

The Bratu two-point BVP

$$y'' = -\lambda e^y, \ \lambda > 0$$
$$y(0) = 0$$
$$y(1) = 0$$

Fact: There are no solutions for λ near 0. Otherwise, there are two.

We truncated the Taylor series of e^y and were able to confirm this fact (and similar facts for several other two-point BVPs).

V. Some successes

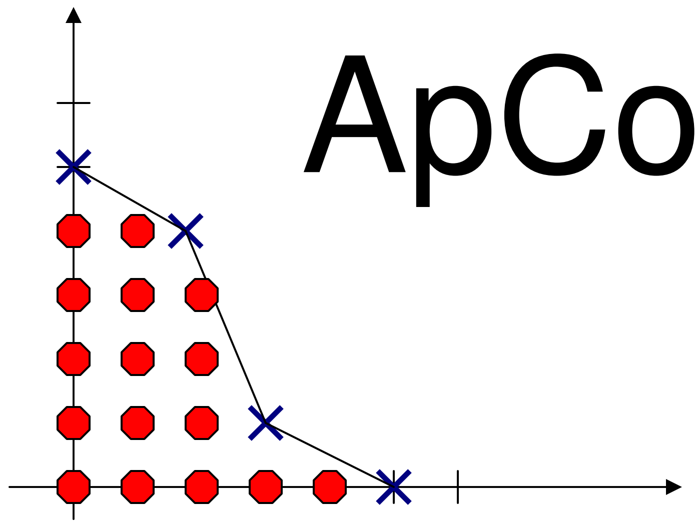
Multiplicity of monomial ideals

x^5

y^5

x^2y^4

x^3y



ApCoA

V. Some successes

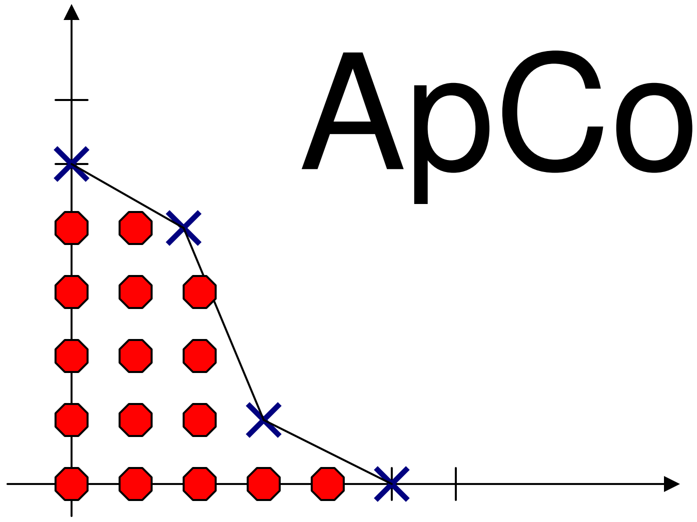
Multiplicity of monomial ideals

x^5

y^5

x^2y^4

x^3y



ApCoA

V. Some successes

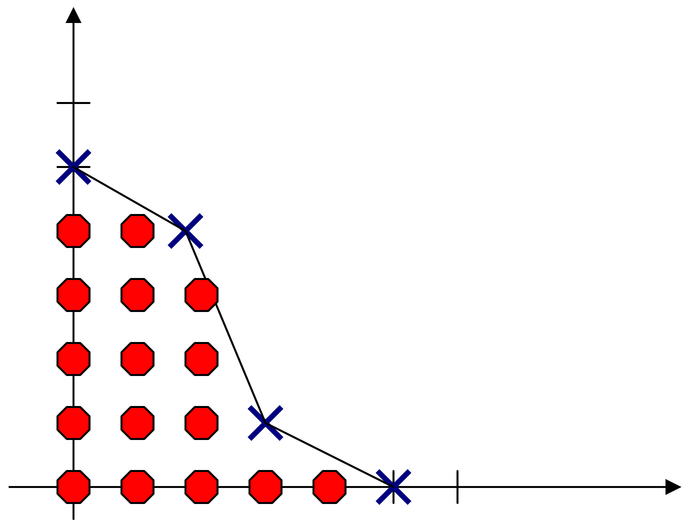
Multiplicity of monomial ideals

x^5

y^5

x^2y^4

x^3y



We get multiplicity = 16, regularity = 6

V. Some successes

Fulton’s multiplicity problem

$x^4 + 2.0*x^2*y^2 + y^4 + 3.0*x^2*y*z - y^3*z$

$x^6 + 3.0*x^4*y^2+3.0*x^2*y^4 + y^6 - 4.0*x^2*y^2*z^2$

V. Some successes

Fulton's multiplicity problem

$x^4 + 2.0{*}x^2{*}y^2 + y^4 + 3.0{*}x^2{*}y{*}z - y^3{*}z$

$x^6 + 3.0{*}x^4{*}y^2 + 3.0{*}x^2{*}y^4 + y^6 - 4.0{*}x^2{*}y^2{*}z^2$

Multiplicity = 14, Regularity = 8

V. Some successes

Fulton's multiplicity problem

$x^4 + 2.0*x^2*y^2 + y^4 + 3.0*x^2*y*z - y^3*z$
$x^6 + 3.0*x^4*y^2+3.0*x^2*y^4 + y^6 - 4.0*x^2*y^2*z^2$

Multiplicity = 14, Regularity = 8

Multiplicity is 14.

V. Some successes

Fulton's multiplicity problem

$x^4 + 2.0*x^2*y^2 + y^4 + 3.0*x^2*y*z - y^3*z$
$x^6 + 3.0*x^4*y^2+3.0*x^2*y^4 + y^6 - 4.0*x^2*y^2*z^2$

Multiplicity = 14, Regularity = 8

Multiplicity is 14.

Also works for perturbed data (not formal!).

References

- B. Dayton and Z. Zeng. Computing the multiplicity structure in solving polynomial systems. *ISSAC* ‘05.

- W. Fulton. Algebraic curves. W.A. Benjamin, New York, 1969.

- A. Leykin, J. Verschelde, and A. Zhao. Evaluation of jacobian matrices for newton’s method with deflation to approximate isolated singular solutions of polynomial systems. *SNC 2005 Proceedings*.

- A. Sommese and C. Wampler. The numerical solution to systems of polynomials arising in engineering and science. World Scientific, Singapore, 2005.

THE END

Thank you for listening!

THE END

Thank you for listening!

Sorry – no Kaltofen this time….