Numerical Algebraic Geometry

Dan Bates
Colorado State University
Goals for this tutorial

- Cover many of the basic **structures**, **methods**, and **assumptions** of numerical algebraic geometry (NAG)
- Provide some training with **Bertini I/O**, with some M2 runs tomorrow
- Address any **concerns** about numerical computation
- Mention a variety of **applications**, both within algebraic geometry and outside mathematics

**Ultimately:**
- Provide enough knowledge and experience to start using NAG methods in your own work
NON-Goals for this tutorial

- Make you into NAG experts
- Describe every alternative algorithm
- Placate your concerns about numerics, the “dark arts of computational algebraic geometry” (according to Frank Sottile)
- Pat anybody’s back or give a fair, balanced history lesson
- Thoroughly introduce every software package
Structure of the tutorial

• (1 hour of lecture + 1.5 hours of “homework”) * 2
• Ask questions!! (Experts: disagree!)

• **Warning**: I am no Macaulay2 expert! I helped Elizabeth Gross, Anton Leykin, and Jose Rodriguez develop Bertini.m2, but I served as the Bertini expert. I like M2 but am still a total newbie!

• I’ll use standalone Bertini mostly….

• **Warning**: I *am* a Bertini expert, but I don’t carry all the nit picky details in my mind. Do ask specific questions, though.

• If you have questions after this week (NAG, Bertini, references, etc.), don’t hesitate to write!
Game plan

1. Setting the ambiance
2. Bertini and friends
3. Basic homotopy continuation
4. Numerical irreducible decomposition
5. Advanced topics
Game plan

1. Setting the ambiance
2. Bertini and friends
3. Basic homotopy continuation
4. Numerical irreducible decomposition
5. Advanced topics
Setting the ambiance

NAG : AG :: NLA : LA
The fundamental goal of NAG is to solve polynomial systems.

Given \( f_1, \ldots, f_n \in \mathbb{C}[z_1, \ldots, z_N] \), we want to find all \( \widehat{z} \in \mathbb{C}^N \) s.t. \( f_i(\widehat{z}) = 0 \ \forall i. \)
Setting the ambiance

\[ \text{NAG : AG :: NLA : LA} \]

The fundamental goal of NAG is to solve polynomial systems.

Given \( f_1, \ldots, f_n \in \mathbb{C}[z_1, \ldots, z_N] \), we want to find all \( \hat{z} \in \mathbb{C}^N \) s.t. \( f_i(\hat{z}) = 0 \ \forall i \).

**Assumptions on \( f = (f_1, \ldots, f_n) \):**

* Always \( \mathbb{C} \) — \( \mathbb{R} \) will come later.
* \( N = n \), for simplicity; \( N \neq n \) will be handled later.
* The coefficients of \( f \) may be approximate, e.g., 3 for \( \pi \), but we aim to solve exactly the system you provide (some exact numbers are OK).
Setting the ambiance

Given \( f_1, \ldots, f_n \in \mathbb{C}[z_1, \ldots, z_N] \), we want to find all \( \hat{z} \in \mathbb{C}^N \) s.t. \( f_i(\hat{z}) = 0 \ \forall i \).

Q: What does this mean??
Setting the ambiance

Given \( f_1, \ldots, f_n \in \mathbb{C}[z_1, \ldots, z_N] \), we want to find all \( \hat{z} \in \mathbb{C}^N \) s.t. \( f_i(\hat{z}) = 0 \ \forall i \).

Q: What does this mean??

Recall: \( Z = \mathcal{V}(f) = \bigcup_{i=0}^{D} Z_i = \bigcup_{i=0}^{D} \bigcup_{j \in \Lambda_i} Z_{i,j} \), where:

- \( D \) is the dimension of \( Z \),
- \( i \) cycles through possible dimensions of irreducible components,
- \( j \) is an index within dimension \( i \), and the
- \( Z_{i,j} \) are the irreducible components.
Setting the ambiance

Example 1:

\[ f = \begin{bmatrix} x(x - 1) \\ x(y - 1) \end{bmatrix} \]
Example 1:

\[
f = \begin{bmatrix}
x(x - 1) \\
x(y - 1)
\end{bmatrix}
\]

We can see that either \(x = 0\) (a line) or \(x = y = 1\) (a point)....
Setting the ambiance

Example 2:

\[ f = \begin{bmatrix} (y - x^2)(x^2 + y^2 + z^2 - 1)(x - 2) \\ (z - x^3)(x^2 + y^2 + z^2 - 1)(y - 2) \\ (z - x^3)(y - x^2)(x^2 + y^2 + z^2 - 1)(z - 2) \end{bmatrix} \]
Example 2:

\[ f = \begin{bmatrix}
(y - x^2)(x^2 + y^2 + z^2 - 1)(x - 2) \\
(z - x^3)(x^2 + y^2 + z^2 - 1)(y - 2) \\
(z - x^3)(y - x^2)(x^2 + y^2 + z^2 - 1)(z - 2)
\end{bmatrix}\]

Here, there are six (mostly less obvious) components:

**Dimension 2:** One surface.

**Dimension 1:** Three lines and one cubic curve.

**Dimension 0:** One point.
Setting the ambiance

Given $f_1, \ldots, f_n \in \mathbb{C}[z_1, \ldots, z_N]$, we want to find all $\tilde{z} \in \mathbb{C}^N$ s.t. $f_i(\tilde{z}) = 0 \ \forall i$.

Q: What does this mean??

For each isolated solution, $\tilde{z}$, we aim to compute a numerical approximation $\tilde{\tilde{z}}$ such that $||\tilde{z} - \tilde{\tilde{z}}|| < \text{FinalTol}$.

For each positive-dimensional irreducible component, $Z_{i,j}$, we aim to find similar numerical approximations to some number of generic points on $Z_{i,j}$. 
Setting the ambiance

About the underlying computations:
* Numerical, inexact
* Probabilistic, involving choices of random numbers
* Theoretically probability one (Zariski open set of good choices)
* Realistically not probability one, since numerics can fail

Mitigating numerical difficulties:
* Adaptive precision
* Recognition and signaling of failures

Fundamental choice: Speed vs. safety?
* Could place fast and loose, but get wrong answers
* Could recognize failures, sacrificing some speed but adding safety
* Could certify (later), sacrificing more speed but adding certainty
About the underlying computations:
  * Numerical, inexact
  * Probabilistic, involving choices of random numbers
  * Theoretically probability one (Zariski open set of good choices)
  * Realistically not probability one, since numerics can fail

Mitigating numerical difficulties:
  * Adaptive precision
  * Recognition and signaling of failures
About the underlying computations:
* Numerical, inexact
* Probabilistic, involving choices of random numbers
* Theoretically probability one (Zariski open set of good choices)
* Realistically not probability one, since numerics can fail

Mitigating numerical difficulties:
* Adaptive precision
* Recognition and signaling of failures

Fundamental choice: Speed vs. safety?
* Could place fast and loose, but get wrong answers
* Could recognize failures, sacrificing some speed but adding safety
* Could certify (later), sacrificing more speed but adding certainty
The tortoise & the hare example

In Banff, Maurice Rojas proposed a challenge — find the number of real isolated solutions of 3 nearly identical polynomial systems

Software package A: 1/2 second on a laptop…wrong answers for all 3.

Software package B: 1 hour on 180 processors…correct answers for all 3.
Game plan

1. Setting the ambiance
2. Bertini and friends
3. Basic homotopy continuation
4. Numerical irreducible decomposition
5. Advanced topics
Bertini and friends

There are many (somewhat) related software packages:

Macaulay2, Singular, CoCoA, GAP, AUTO, MATCONT, etc.

and many derivatives

Paramotopy (Wed, 9), alphaCertified, KhRo, BertiniLab, etc.

but here are the main, currently active NAG software packages (in alphabetical order):

Bertini (Dan Bates, Jon Hauenstein, Andrew Sommese, Charles Wampler)
HOM4PS-2,3 (Tianran Chen, TY Li, Tsung-Lin Lee)
NumericalAlgebraicGeometry, in M2 (Anton Leykin)
PHCpack (Jan Verschelde)

I will focus on Bertini, due to time & expertise (no offense intended!).
Please talk to the others about the other packages.
Bertini and friends

Bertini is free to download and use, and the source code is publicly available. It may be downloaded (source code or binary) at bertini.nd.edu.

2002-2006: Pre-release development.
2006: Initial beta release, during IMA special year.
2006-2013: Various releases.
2013: Version 1.4.

Next: 2.0

Large development team, C++ (not C), GPL (some flavor), modules, GitHub

What should we add/change?
Want to join the development team?
Here’s a nice, simple example of a run....

Bertini and friends
1. Setting the ambiance

2. Bertini and friends

3. Basic homotopy continuation

4. Numerical irreducible decomposition

5. Advanced topics
3. Basic homotopy continuation

A. Homotopies & path-tracking

B. Root counts & start systems

C. Adaptive precision

D. Endgames

E. Sharpening endpoints

F. Advanced topics: Deflation and regeneration
Basic homotopy continuation
A. Homopties & path-tracking

Given polynomial system $f : \mathbb{C}^N \rightarrow \mathbb{C}^N$ (the target system) homotopy
continuation is a 3-step process:

1. Choose and solve a polynomial system $g : \mathbb{C}^N \rightarrow \mathbb{C}^N$ (the start system)
   based on characteristics of $f(z)$ but relatively easy to solve.
Basic homotopy continuation
A. Homotopies & path-tracking

Given polynomial system \( f : \mathbb{C}^N \rightarrow \mathbb{C}^N \) (the target system) homotopy continuation is a 3-step process:

1. Choose and solve a polynomial system \( g : \mathbb{C}^N \rightarrow \mathbb{C}^N \) (the start system) based on characteristics of \( f(z) \) but relatively easy to solve.

2. Form the homotopy \( H : \mathbb{C}^N \times \mathbb{C} \rightarrow \mathbb{C}^N \) given by
   \[
   H(z, t) = f(z) \cdot (1 - t) + g(z) \cdot t
   \]
   so that \( H(z, 1) = g(z) \) and \( H(z, 0) = f(z) \).
Basic homotopy continuation

A. Homopties & path-tracking

Given polynomial system $f : \mathbb{C}^N \rightarrow \mathbb{C}^N$ (the target system) homotopy continuation is a 3-step process:

1. Choose and solve a polynomial system $g : \mathbb{C}^N \rightarrow \mathbb{C}^N$ (the start system) based on characteristics of $f(z)$ but relatively easy to solve.

2. Form the homotopy $H : \mathbb{C}^N \times \mathbb{C} \rightarrow \mathbb{C}^N$ given by

   $$H(z, t) = f(z) \cdot (1 - t) + g(z) \cdot t$$

   so that $H(z, 1) = g(z)$ and $H(z, 0) = f(z)$.

3. Use numerical predictor-corrector methods (see Allgower-Georg) to follow the solutions as $t$ marches from 1 to 0, one solution at a time.

Here’s a schematic…. 
Basic homotopy continuation
A. Homopties & path-tracking

SQUARING

If $N > n$ (more variables), there will be no isolated solutions so there is nothing more to do (for now)....

If $N < n$ (more equations), we can replace $f$ with $N$ linear combinations of the polynomials of $f$ (thanks to a theorem of Bertini). We could pick up spurious solutions or increase multiplicities, but at least “Bertini junk” is easily spotted and removed (automatically by Bertini)....

FACT: The number of paths ending at one isolated solution is an upper bound on the multiplicity of that solution (equal if no squaring).
Basic homotopy continuation

A. Homopties & path-tracking

THE GAMMA TRICK   Recall:

2. Form the homotopy \( H : \mathbb{C}^N \times \mathbb{C} \to \mathbb{C}^N \) given by

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

so that \( H(z, 1) = g(z) \) and \( H(z, 0) = f(z) \).
The Gamma Trick

Recall:

2. Form the homotopy $H : \mathbb{C}^N \times \mathbb{C} \to \mathbb{C}^N$ given by

$$H(z, t) = f(z) \cdot (1 - t) + g(z) \cdot t$$

so that $H(z, 1) = g(z)$ and $H(z, 0) = f(z)$.

In fact, we use the homotopy:

$$H(z, t) = f(z) \cdot (1 - t) + \gamma g(z) \cdot t$$

where $\gamma \in \mathbb{C}$ is chosen at random. Here’s why….
Basic homotopy continuation
A. Homopties & path-tracking

DETECTING DIVERGENCE

Input can be non-homogeneous (use variable_group) or homogeneous (use hom_variable_group).

Either way, Bertini will homogenize (if necessary) and work over a random patch of $\mathbb{P}^N$. Thus, paths of infinite length become paths of finite length.

Even so, paths diverging to infinity often have highly singular endpoints, so it is preferable to avoid them. In Bertini, we kill any path that exceeds the threshold SecurityMaxNorm after $t$ reaches 0.1.
CHECKING FOR PATH-CROSSING

We refer to the error of jumping from one path to another during path tracking as *path-crossing*.

We help to mitigate this sort of error by collecting all path data at \( t=0.1 \) and post-processing to make sure all paths were still separate at that point. If not, we spit out an error message to the screen.

(At \( t=0.1 \), the endgame kicks in with tighter tracking tolerances.)
Basic homotopy continuation
A. Homopties & path-tracking

Any questions about the general homotopy continuation setup before we move on to root counts & start systems?
Basic homotopy continuation
B. Root counts & start systems

Bézout: # finite, isolated solutions $\leq \prod_{i=1}^{N} \deg(f_i)$. 
Basic homotopy continuation
B. Root counts & start systems

**Bézout:** \# finite, isolated solutions \( \leq \prod_{i=1}^{N} \deg(f_i) \).

One choice of start system is the *total degree* or *Bézout start system*:

\[
g = \begin{bmatrix}
    z_1^{d_1} - 1 \\
    \vdots \\
    z_N^{d_N} - 1
\end{bmatrix}.
\]

This has exactly \( \prod_{i=1}^{N} \deg(f_i) \) isolated, nonsingular, finite solutions.
Basic homotopy continuation
B. Root counts & start systems

**Bézout:** # finite, isolated solutions \( \leq \prod_{i=1}^{N} \deg(f_i) \).

One choice of start system is the *total degree or Bézout start system*:

\[
g = \begin{bmatrix}
    z_1^{d_1} - 1 \\
    \\
    \\
    z_N^{d_N} - 1
\end{bmatrix}.
\]

This has exactly \( \prod_{i=1}^{N} \deg(f_i) \) isolated, nonsingular, finite solutions.

This could be overkill — there might be many fewer solutions!

Different root counts lead to different start systems.
Basic homotopy continuation
B. Root counts & start systems

Bézout++: # finite, isolated solutions is also bounded above by some combinatorial formula built from the multidegrees of the polynomials, when the variables are broken into multiple groups.

Depending on choice of variable groups, you might get fewer or more startpoints (typically more).

Example: \[
\begin{bmatrix}
xy - 1 \\
x^2 - 1
\end{bmatrix}
\]

The total degree is 4.
Example:

\[
\begin{bmatrix}
xy - 1 \\
x^2 - 1
\end{bmatrix}
\]

The total degree is 4.
Basic homotopy continuation
B. Root counts & start systems

Example:
\[
\begin{bmatrix}
xy - 1 \\
x^2 - 1
\end{bmatrix}
\]

The total degree is 4.

\[
\begin{array}{c|cc}
& (x) & (y) \\
xy - 1 & 1 & 1 \\
x^2 - 1 & 2 & 0
\end{array}
\]

The 2-homogeneous (or 2-hom) degree is 2, so we can build a start system with 2 nonsingular, isolated solutions and follow only 2 paths to find all solutions.
It is tempting to just try all possible variable groupings…DON’T!

The number of such groupings for n variables grows as the Bell number:

<table>
<thead>
<tr>
<th>N</th>
<th>Bell(N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
</tr>
<tr>
<td>6</td>
<td>203</td>
</tr>
<tr>
<td>7</td>
<td>877</td>
</tr>
<tr>
<td>8</td>
<td>4140</td>
</tr>
<tr>
<td>9</td>
<td>24366</td>
</tr>
<tr>
<td>10</td>
<td>155117</td>
</tr>
<tr>
<td>11</td>
<td>1004715</td>
</tr>
<tr>
<td>12</td>
<td>7466662</td>
</tr>
<tr>
<td>13</td>
<td>57129997</td>
</tr>
<tr>
<td>14</td>
<td>476726964</td>
</tr>
<tr>
<td>15</td>
<td>4080948535</td>
</tr>
<tr>
<td>16</td>
<td>36055088480</td>
</tr>
<tr>
<td>17</td>
<td>328036902735</td>
</tr>
<tr>
<td>18</td>
<td>3035244772418</td>
</tr>
</tbody>
</table>

Table 5.1. Illustration of the growth of Bell(N) as N grows.

example, the cubic $x^2z + xyz + 1$ is quadratic in $(x, y)$ and linear in $z$, so a partition of the variables as \{(x, y), (z)\} could be advantageous. If all three polynomials in a system of three polynomials in $C[x, y, z]$ had this same degree structure, then it would certainly be an advantage to partition the variables in this manner. The situation becomes more complicated when a subdivision reduces degrees in some polynomials of the system but increases them in others.

5.1.1 Multihomogeneous Bézout number

The number of paths in a multihomogeneous homotopy, also called the multihomogeneous Bézout number, of the system, is closely tied to multihomogeneous projective spaces already mentioned in §4.5. To decide which grouping of the variables to use, one may easily compute the multihomogeneous Bézout number of each grouping under consideration, and pick the smallest one. As already mentioned, the total number of possible groupings grows quickly with the number of variables, so Bertini does not automate an exhaustive search. But with a little practice, a user can identify and check the most promising candidates.

Instead of introducing a detailed notation for expressing the multihomogeneous Bézout number, let us demonstrate the idea through examples. Let’s start with the system:

$$xy - 1 = 0, x^2 - 1 = 0.$$ (5.1)

There are two possible groupings, \{(x, y)\} and \{(x), (y)\}, each with its own Bézout number. For each grouping, we make a multidegree table with one column per group and one row per equation, as follows. For the grouping \{(x, y)\}, the table is:

<table>
<thead>
<tr>
<th></th>
<th>$xy - 1$</th>
<th>$x^2 - 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x, y)</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

For brevity, we may sometimes call this variously the Bézout number’, the Bézout count’, or even simply the root count’.
Basic homotopy continuation
B. Root counts & start systems

It is tempting to just try all possible variable groupings…DON’T!

The number of such groupings for n variables grows as the Bell number:

<table>
<thead>
<tr>
<th>N</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bell(N)</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>15</td>
<td>52</td>
<td>203</td>
<td>877</td>
<td>4140</td>
</tr>
<tr>
<td>N</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bell(N)</td>
<td>21,147</td>
<td>115,975</td>
<td>678,570</td>
<td>4,213,597</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bell(N)</td>
<td>4,638,590,332,229,999,353</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Warning: I am entirely ignoring polyhedral homotopies, which can be very efficient in terms of the number of paths to be tracked but is sometimes expensive in terms of precomputation.
Basic homotopy continuation
C. Adaptive precision

For matrix \( A \in \mathbb{C}^{N \times N} \), the singular value decomposition (SVD) of \( A \) is a decomposition \( A = U\Sigma V^* \) with various properties.

For our purposes, the key is that \( \Sigma \) is diagonal with nonnegative real entries called the singular values of \( A \).

Using this, we can define the condition number of \( A \) as:

\[
\kappa(A) = \frac{s_{\text{max}}}{s_{\text{min}}}
\]
Basic homotopy continuation
C. Adaptive precision

For matrix $A \in \mathbb{C}^{N \times N}$, the singular value decomposition (SVD) of $A$ is a decomposition $A = U\Sigma V^*$ with various properties.

For our purposes, the key is that $\Sigma$ is diagonal with nonnegative real entries called the singular values of $A$.

Using this, we can define the condition number of $A$ as:

$$\kappa(A) = \frac{s_{\text{max}}}{s_{\text{min}}}$$

**Wilkinson:** When solving linear system $Ax = b$,

$$\text{ACC} \approx \text{PREC} - \log_{10}(\kappa(A))$$
Basic homotopy continuation

C. Adaptive precision

\[ \text{ACC} \approx \text{PREC} - \log_{10}(\kappa(A)) \]

So, when the condition number gets high, we can increase precision to salvage accuracy. (There are many details....)

This isn’t free!! (There is a new multiprecision LAPACK on the way.)

The key point is that zones of ill-conditioning can cause numerical trouble but AMP reduces the size of these zones significantly. Of course, there is a limit on PREC and failure-causing pathologies can be constructed.

This is OK, so long as we recognize and report such failures!
Basic homotopy continuation
D. Endgames

We cannot avoid \( t=0 \) — that is our target!

Interesting systems often have singular solutions, but we cannot track into \( t=0 \). So, we have endgames.
Basic homotopy continuation
D. Endgames

We cannot avoid $t=0$ — that is our target!

Interesting systems often have singular solutions, but we cannot track into $t=0$. So, we have endgames.

**Two main options:**

1. Power series endgame….

2. Cauchy endgame….
After finding an isolated solution, it is sometimes necessary to sharpen the endpoint, i.e., find more (accurate) digits.

If the solution is nonsingular, this is just a matter of running Newton’s method several more times.

If the solution is singular, we need deflation (next).

Bertini can do this....
Basic homotopy continuation
F. Advanced topics: Deflation and regeneration

Main idea of deflation: Given a singular isolated solution of \( f \), replace \( f \) with a polynomial system having the original solution as a nonsingular solution.

There are many manifestations of deflation (starting with Leykin-Verschelde-Zhao for polynomial systems), but here’s the simplest version:

\[
\begin{bmatrix}
    f(x) \\
    Jf(x) \cdot \lambda \\
    \mathcal{K}(\lambda)
\end{bmatrix}
\]

where:

\[
d = \dim \text{null } Jf(x^*)
\]

\( \mathcal{K} : \mathbb{C}^N \rightarrow \mathbb{C}^d \) is a set of \( d \) general liners.

we replace singular solution \( x^* \) with \( (x^*, \lambda^*) \).

The new solution will have lower multiplicity than the original solution.

If it is still singular, we can repeat this process.

Leykin-Verschelde-Zhao showed that this process will terminate.
Main idea of regeneration: Solve the polynomial system one equation at a time; this is the latest, greatest equation-by-equation method.

This is just a sequence of homotopies of a certain type....

To use this in Bertini, use the config “UseRegeneration: 1”....
Game plan — tomorrow!

1. Setting the ambiance
2. Bertini and friends
3. Basic homotopy continuation
4. Numerical irreducible decomposition
5. Advanced topics