# Solving Algebraic Systems of Equations

*Daniel Lichtblau*

*Wolfram Research, Inc.*

*July 2000*

**Presented at: SCI2000, Orlando**

## Overview

Systems of polynomial equations with finitely many solutions arise in many areas of industrial and applied mathematics. Among these are motion planning, robotics, computer- aided design, and graphics. In this talk I discuss the design and implementation of a numeric solver for algebraic systems. It makes use of hybrid symbolic/numeric functionality present in the *Mathematica* kernel: numeric Gröbner bases and arbitrary-precision numeric eigenvalue computation. I will present some examples that demonstrate how this enhances considerably the capabilities of *Mathematica* in this area.

## The key ingredients

(1) Numeric Groebner basis computation.

This gives equivalent equations from which convenient linear algebra methods may be applied to extract solutions.

(2) Eigenvalue computation. This solves for one variable.

(3) Solve linear equations to obtain solutions for other variables in terms of the one obtained in step (1).

Alternatively, plug the solution for the last variable into the equations and redo the process for each new system in order to solve for the remaining variables. This is effective because we usually spawn much smaller sub- problems.

# Step (1)

First question: What is a Gröbner basis?

A full answer would take too long. In brief, it is a tool from computational commutative algebra that generalizes the notion from matrix algebra of row- reducing a system of linear equations to echelon form. For example, given a system of polynomials one can, in some sense, "triangulate" that system.

Here is a quick example.

```
polys = {2 x^2 - 3 x y + x + y - 4,
    x^3 + 4 x y^2 - y^2 + 3 x - 2 y + 6};
```

```
gb = GroebnerBasis[polys, {x, y}]
```

$$\{346 - 875\,y + 1445\,y^2 - 833\,y^3 + 521\,y^4 - 25\,y^5,\ 483\,642 + 218\,447\,x - 489\,124\,y + 387\,939\,y^2 - 52\,124\,y^3 + 1725\,y^4\}$$

We have a new set of polynomials with the following properties.

(i) They generate the same ideal (in the sense of abstract algebra) as the original polynomials. This means, in essence, that they have identical solution sets.

(ii) They are "triangulated". Specifically we have a univariate polynomial in y, and a polynomial that is linear in x. Thus to solve the system, one merely finds roots in y, then plugs into the second polynomial to get solutions in x. In a sense I will not try to make specific, this is the typical state of affairs for Gröbner bases.

## Why all the fuss?

As a practical matter, computation of Gröbner bases can be extremely time- consuming. Coefficient swell can be a severe bottleneck, and moreover the computation is sensitive to something called the term order. What is term order? Again, being intentionally vague, it is the way that we choose to order powers of variables. In the example, we take powers of $x$ to be "larger" than all powers in $y$ alone. This is called a "lexicographic" term order. Such orders are obviously useful (we can get solutions quite easily, as noted above), but they are typically very expensive to compute. Term orders that are based on total degree of monomials tend to be cheaper for purposes of computing a Gröbner basis, but the result is less convenient for our purposes. Hence the need for steps (2) and (3).

## Step (2)

Suppose we have the Gröbner basis shown above. Here is a seemingly roundabout way to get solutions to the polynomial system. First take the polynomial in $y$. It is

$$346 - 875\,y + 1445\,y^2 - 833\,y^3 + 521\,y^4 - 25\,y^5$$

To this polynomial we can associate a companion matrix.

```
companionMatrix[
  poly_?PolynomialQ] :=
 Module[{vars, coeffs, newcoeffs,
   n, m}, vars = Variables[poly];
  If[! ListQ[vars] ||
    Length[vars] ≠ 1, Return[]];
  coeffs = CoefficientList[poly,
    vars];
  newcoeffs =
   Drop[coeffs / (-Last[coeffs]),
    -1];
  n = Exponent[poly, First[vars]] -
    1;
  m = Prepend[IdentityMatrix[n],
    Table[0, {n}]];
  Transpose[Append[Transpose[m],
    newcoeffs]]]
```

```
MatrixForm[
 mat = companionMatrix[gb[[1]]]
```

$$\begin{pmatrix} 0 & 0 & 0 & 0 & \frac{346}{25} \\ 1 & 0 & 0 & 0 & -35 \\ 0 & 1 & 0 & 0 & \frac{289}{5} \\ 0 & 0 & 1 & 0 & -\frac{833}{25} \\ 0 & 0 & 0 & 1 & \frac{521}{25} \end{pmatrix}$$

## What comes next?

We will now find numeric eigenvalues of this matrix.

```
Sort[Eigenvalues[N[mat]]]
```

```
{0.38675 - 0.47624 i,
 0.38675 + 0.47624 i,
 0.402705 - 1.32171 i,
 0.402705 + 1.32171 i, 19.2611 + 0. i}
```

## So why did we compute those eigenvalues?

So here is the crux of step (2). We can use a Gröbner basis computed with respect to ANY term order, to obtain what is called a "generalized companion matrix" relative to any give variable. Its eigenvalues will be the same as those of the ordinary companion matrix for that variable. So we find a Gröbner basis in step (1) using some order that is less expensive to compute than lexicographic, and in step (2) we set up and extract eigenvalues from a generalized companion matrix.

## Step (3)

Now that we have our solutions in one variable, we can use back- substitution to spawn what are often simpler systems, in order to solve recursively for the other variables.

But other tactics are (typically) faster. One method uses linear algebra to "convert" polynomials from the Gröbner basis into polynomials that are linear in the remaining variables (so back- substitution becomes trivial).

Another method makes use of corresponding eigenvectors to get solutions in the remaining variables. Suffice it to say that they are computationally not much more strenuous than the eigenvalue extraction of step (2).

## Again, why all the fuss?

Obtaining a numerical Gröbner basis is a tricky business. One must use arithmetic that can keep track of precision, and one must have a way to recognize when "small" values are zero.

The gain, though, is in speed. Coefficient swell is not a problem (the number of digits does not increase from its initial value). Also as noted above, we can use cheaper term orders.

Thus for hard problems we have a much better chance of completion in reasonable time than an exact computation.

# A stroll through a simple example

We use the same polynomials as before. The numeric Gröbner basis for a degree- based term order is given below.

```
ngb = GroebnerBasis[N[polys],
   {x, y}, MonomialOrder →
    DegreeReverseLexicographic]
```

$$\{-2. + 0.5\,x + 1.\,x^2 + 0.5\,y - 1.5\,x\,y,\ -28.8 - 12.94\,x + 26.3\,y - 1.38\,x\,y - 20.52\,y^2 + 1.\,y^3,\ 0.8 + 0.84\,x + 0.2\,y - 0.32\,x\,y - 0.28\,y^2 + 1.\,x\,y^2\}$$

We break the Gröbner basis into head terms and "tail" polynomials. This is a main step in finding the endomorphism matrix.

```
fdl =
  Internal‘FromDistributedTermsList
   [
   Internal‘DistributedTermsList[
    ngb, {x, y},
    MonomialOrder →
     DegreeReverseLexicographic],
   List];
```

```
{heads, tails} =
 {Map[First, fdl], Map[Rest, fdl]}
```

$$\left\{\left\{1.\, x^2,\ 1.\, y^3,\ 1.\, x\, y^2\right\},\ \left\{\left\{-1.5\, x\, y,\ 0.5\, x,\ 0.5\, y,\ -2.\right\},\ \left\{-1.38\, x\, y,\ -20.52\, y^2,\ -12.94\, x,\ 26.3\, y,\ -28.8\right\},\ \left\{-0.32\, x\, y,\ -0.28\, y^2,\ 0.84\, x,\ 0.2\, y,\ 0.8\right\}\right\}\right\}$$

Here is the list of "exponent tuples" for the head monomials of the Gröbner basis: `{{2, 0}, {0, 3}, {1, 2}}`

All tuples that lie strictly under the "staircase" they form give basis elements in a representation of the vector space `C[x, y] / (polys)`.

Here is the list of the tuples under the staircase (sounds like the name of a horror flick):
`{{0, 2}, {0, 1}, {2, 0}, {1, 0}, {0, 0}}`

To see how the endomorphism "multiply by $y$" transforms the vector space basis monomial $y^2$, we can do

```
Last[PolynomialReduce[y^3, ngb,
  MonomialOrder →
   DegreeReverseLexicographic]]
```

```
28.8 + 12.94 x -
 26.3 y + 1.38 x y + 20.52 y^2
```

This will become a column in our endomorphism matrix.

Here is the full endomorphism matrix. It is `5 x 5`, because there are `5` basis elements.

$$\begin{pmatrix} 20.52 & 0.28 & 1 & 0 & 0 \\ 1.38 & 0.32 & 0 & 1 & 0 \\ -26.3 & -0.2 & 0 & 0 & 1 \\ 12.94 & -0.84 & 0 & 0 & 0 \\ 28.8 & -0.8 & 0 & 0 & 0 \end{pmatrix}$$

We see that the action on the basis element $y^2$ became column one. This is no surprise because the tuple {0, 2} corresponding to $y^2$ was the first one listed.

The eigenvalues are the values of y for the solutions. There are five such, because the matrix is 5 x 5. Here is one of the values obtained: 19.26108948234646. We plug in this value and next solve for x in the overdetermined system below.

$$\left\{7.63054 - 28.3916\,x + 1.\,x^2,\ 10.724 - 39.5203\,x,\ -99.2249 + 365.666\,x\right\}$$

We arrive at a solution for x. It is 0.271354. If we continue in this manner we will obtain:

$$\{\{x \to 0.271354,\ y \to 19.2611\},\ \{x \to 1.00317 + 0.698106\,I,\ y \to 0.402705 + 1.32171\,I\},\ \{x \to 1.00317 - 0.698106\,I,\ y \to 0.402705 - 1.32171\,I\},\ \{x \to -1.25884 + 0.437825\,I,\ y \to 0.38675 + 0.47624\,I\},\ \{x \to -1.25884 - 0.437825\,I,\ y \to 0.38675 - 0.47624\,I\}\}$$

# A benchmark example from the literature

```
polys = {45 * P + 35 * S - 165 * B - 36,
    35 * P + 40 * Z + 25 * T - 27 * S,
    15 * W + 25 * P * S + 30 * Z - 18 * T -
     165 * B^2,
    -9 * W + 15 * P * T + 20 * Z * S,
    W * P + 2 * Z * T - 11 * B^3,
    99 * W - 11 * S * B + 3 * B^2};
```

```
vars = {P, S, B, Z, T, W};
```

```
Timing[
 sol = NSolve[polys == 0, vars];]
```

```
{3.41 Second, Null}
```

Let us check the solutions. We plug them into the original polynomials and then truncate values less than $\frac{1}{10^{10}}$ to zero.

```
Apply[Plus, polys /. sol] // Chop
```

```
{0, 0, 0, 0, 0, 0}
```

## Competing methods

(i) Sparse homotopy methods.

(ii) Multipolynomial resultant methods (can be quite similar to our method; depends on how one obtains and/or uses the resultant).

These, and the method of this talk, have all blossomed in the past 10- 12 years or so.

## Methods that do not compete (very well)

(i) Exact methods (frequently they are MUCH too slow).

(ii) Newton’s method and its relatives. It only finds one root, and needs a reasonable initial guess. If this fits your needs, by all means use it. But it is not a reliable way to find ALL solutions.

# Summary

We have presented a method to find all numeric solutions to systems of algebraic equations that have finitely many solutions. Underlying technology: numeric Gröbner basis computed to moderate precision, eigenvalue extraction, and simple linear algebra.

Gain: Demonstrated capability to solve problems previously inaccessible to *Mathematica.*

Possible task for future development:

Use a multivariate version of residues from complex analysis to obtain solutions ONLY for roots that lie in a specified region. This might be useful e.g. to obtain only real- valued roots (an important problem with no satisafactory solution to date).