TR-00-21.mws

Symbolic-Numeric Algorithms for Polynomials

Robert M. Corless, Deputy Director

Ontario Research Centre for Computer Algebra

Professor, Department of Applied Mathematics

Honorary Professor, Department of Computer Science

University of Western Ontario, London, Canada

Rob.Corless@uwo.ca

http://www.orcca.on.ca

Concepts, Definitions and Applications.

Approximate Polynomial

An *approximate polynomial* is simply a polynomial with inexactly-known coefficients. You may have data with measurement errors that provides, after some kind of fitting, a polynomial that describes your data. You may have a formula with parameters for each coefficient, but the values of the parameters may be known only through experimental measurements. Approximate polynomials arise in many applied problems, from direct mathematical models through normal forms and bifurcation theory to geometric problems described in floating-point arithmetic for Computer-Aided Design.

The idea that we don't know precisely what the coefficients are moves us from the domain of algebra (in the case of polynomials, from classical algebra) to the domain of analysis. In particular, this allows us to bring modern tools of numerical analysis to problems of computational algebra and geometry.

Experience with numerical *linear* algebra suggests that it will be useful to replace classical discrete algorithms for polynomial problems with continuous, iterative algorithms for approximate polynomial problems. In this fashion we hope to achieve a body of results useful enough to be termed *numerical nonlinear algebra* .

Groebner basis

Suppose that we have a system of multivariate polynomial equations. A Groebner basis (aka standard basis) is, for the purpose of discussion here, an equivalent but simpler system of multivariate polynomial equations: equivalent in that the Groebner basis describes the same variety (has the same zeros) and is simpler in that many questions about the polynomial system (e.g. ideal membership, counting the number of roots) are easier to answer. An excellent introduction to Groebner bases can be found in

David Cox, John Little, and Donal O'Shea, *Ideals, Varieties and Algorithms,* Springer-Verlag, 1992.

Groebner bases can be computed for many systems of practical interest, but not all: the Buchberger algorithm is a finite algorithm, but some Groebner bases are exponentially more lengthy than their input.

```
> with(Groebner);
```

$$[fglm, gbasis, gsolve, hilbertdim, hilbertpoly, hilbertseries, inter\_reduce, is\_finite, is\_solvable, leadcoeff, leadmon, leadterm, normalf, pretend\_gbasis, reduce, spoly, termorder, testorder, univpoly]$$

```
> origsys := { x^2-y^2 - 1, x^3 - y^3 - 1};
```

$$origsys := \{x^3 - y^3 - 1, x^2 - y^2 - 1\}$$

```
> gb := gbasis( origsys, plex(x,y) );
```

$$gb := [3\,y^2 + 3\,y^4 - 2\,y^3, -3\,y^3 - 2 + 2\,x - y^2]$$

Notice in that *lexicographic ordered* Groebner basis, the first polynomial contains only y; the second is linear in x, and once y is known, the corresponding x is known. Therefore there are 4 roots to the original system (and here they are easy to find). Lexicographic ordered bases are problematic, however, in that (a) they can be doubly-exponential (!) in cost to compute, and (b) they are at least as numerically unstable as computing the characteristic polynomial is for an eigenvalue problem (indeed this is a generalization of that process).

So for computation we prefer a degree ordering:

```
> gbt := gbasis( origsys, tdeg(x,y) );
```

$$gbt := [2\,y^2 + y\,x - x - y + 1, x^2 - y^2 - 1, 3\,y^3 + y^2 - 2\,x + 2]$$

To the human eye, this is not as convenient as a lex-order basis---but we will see later that computationally it is better in some ways.

```
>
```

Discontinuity

A very significant problem with Groebner bases (or many other polynomial concepts, such as GCD) is that they can be *discontinuous* with respect to parameters. This means that we will have problems with numerical computation of Groebner bases (particularly for approximate polynomials) and we will also have problems with symbolic computation of Groebner bases---because the generic-case computed solution will not always be true for all values of the parameters.

```
> restart;
```

```
> with(Groebner):
```

```
> origsys := {c*x^2 - y^2 - 1, x^3 - y^3 - 1};
```

$$origsys := \{x^3 - y^3 - 1, c\,x^2 - y^2 - 1\}$$

```
> gb := gbasis(origsys,tdeg(x,y));
```

$$gb := [c\,x^2 - y^2 - 1, -c\,y^3 + x\,y^2 + x - c, -y^4 + c^3\,y^4 - c^2\,y\,x - 2\,y^2 + c^2\,x + c^3\,y - 1]$$

```
> map(limit, gb, c=0 );
```

$$[-y^2 - 1, x\,y^2 + x, -y^4 - 1 - 2\,y^2]$$

That is what we would get by substituting c=0 into our Groebner basis, but we see that if we do things in the other order, we get a different result (indeed, the above is not even a Groebner basis).

```
> gb0 := gbasis(map(limit,origsys,c=0), tdeg(x,y) );
```

$$gb0 := [y^2 + 1, x^3 + y - 1]$$

This problem is solved in general by considering a so-called comprehensive Groebner basis (see the papers by Weispfenning, and work in REDLOG by Dolzmann and Sturm). We take a ``deferred evaluation'' approach here and simply keep this problem in mind (we will see a method in an example below).

Commuting families of matrices

A matrix family A[i], i = 1..n, is a commuting family if A[i]A[j] = A[j]A[i] for all i,j in 1..n. Of course, only special families of matrices commute with each other. When matrices commute, there are many practical consequences, the first of which is that their invariant subspaces are related.

Theorem (see e.g. Horn and Johnson): If A[i] is a commuting family, then there exists a unitary matrix V that simultaneously upper-triangularizes the family: T[i] = V* A[i] V is upper triangular for all i.

Commuting families of matrices arise in this context as the (dual of the) standard representation for multiplication by the variable x[i] in the commutative ring P/I, where I is the ideal generated by a system of polynomial equations.

For approximate polynomials and approximate computation of the commuting families, using floating-point arithmetic, we have to think about what happens if we have only a *nearly-commuting* family of matrices: ||A[i]A[j] -A[j]A[i]|| < e. It turns out that

Theorem (Corless, Gianni & Trager 1997): If A[i] is a nearly-commuting family with commutators bounded by epsilon, then there exists a unitary matrix V that simultaneously nearly block upper-triangularizes the family; the size of the subdiagonal entries is bounded by epsilon, and each block clusters nearby eigenvalues.

Problems examined to date

GCD of approximate polynomials

Corless, Gianni, Trager, & Watt ``The SVD for Polynomial Systems'', Proceedings 1995 ISSAC, Montreal.
Sigsam Bulletin Special Issue on Symbolic-Numeric Algorithms for Polynomials, 1996, RMC ed.

Karmarkar & Lakshman, ``Approximate Polynomial Greatest Common Divisors and Nearest Singular Polynomials'', Proceedings 1996 ISSAC, Zuerich

Chin, Corless, & Corliss , ``Optimization Strategies for the Approximate GCD Problem'', Proceedings 1997 ISSAC, Rostock.

Journal of Symbolic Computation Special Issue on Symbolic-Numeric Algorithms for Polynomials, Stetter & Watt eds.

Hans J. Stetter , ``The Nearest Polynomial with a Given Zero, and Similar Problems'', Sigsam Bulletin 33 no. 4 December 1999 pp 2--4.

Many other authors, including *Andre Galligo, Victor Pan* .

Functional decomposition of approximate polynomials

Corless, Giesbrecht, Jeffrey & Watt , ``Approximate Polynomial Decomposition'', Proc. ISSAC 1999 Vancouver.

Finding roots of multivariate systems via eigenproblems

Auzinger & Stetter , ``An Elimination Algorithm for the Computation of All Zeros of
a System of Multivariate Polynomial Equations'', Proc. Numerical Mathematics, Singapore, World Scientific, 1988

Corless, Gianni, Trager, & Watt ``The SVD for Polynomial Systems'', Proceedings 1995 ISSAC, Montreal.
Sigsam Bulletin Special Issue on Symbolic-Numeric Algorithms for Polynomials, 1996, RMC ed.

Journal of Symbolic Computation Special Issue on Symbolic-Numeric Algorithms for Polynomials, 1998, Stetter & Watt eds.

Many papers of *Bernard Mourrain*

Many papers of *Ioannis Emiris*

Numerical Implicitization by Linear Algebra

Corless, Giesbrecht, Kotsireas, Watt , ``Numerical Implicitization of Parametric Hypersurfaces with Linear Algebra", proceedings AISC 2000, Madrid. Springer LNAI 1930.

Division of approximate polynomials

Corless, Giesbrecht, Kotsireas, Watt, ``Division of Approximate Polynomials", in progress.

Examples

Implicitization

```
> restart;
```

Suppose we have the following parametric representation of a curve.

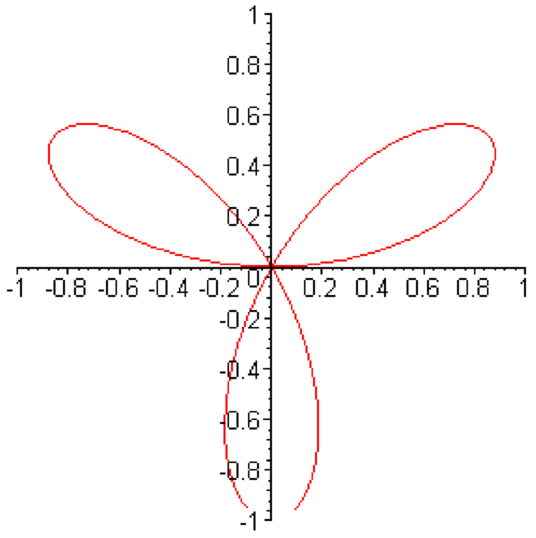
```
> xp := t -> (t*(3-t^2))/(1+t^2)^2;
```

$$xp := t \to \frac{t\,(3-t^2)}{(1+t^2)^2}$$

```
> yp := t -> (t^2*(3-t^2))/(1+t^2)^2;
```

$$yp := t \to \frac{t^2\,(3-t^2)}{(1+t^2)^2}$$

```
> plot([xp,yp,-10..10], view=[-1..1,-1..1], scaling=CONSTRAINED);
```



We can find an implicit (x,y) representation of this curve from the parameterization in many ways, for example by resultants:

```
> resultant(numer(X-xp(t)),numer(Y-yp(t)),t)/256;
```

$$X^4 + 2\,X^2\,Y^2 - 3\,Y\,X^2 + Y^4 + Y^3$$

Or, by using Groebner bases:

```
> with(Groebner):
```

```
> eqs := {denom(xp(t))*X-numer(xp(t)),
denom(yp(t))*Y-numer(yp(t)),
denom(xp(t))*Z-1};
```

$$eqs := \{(1+t^2)^2\,Z - 1, X\,(1+t^2)^2 + t\,(-3+t^2), Y\,(1+t^2)^2 + t^2\,(-3+t^2)\}$$

```
> gb := gbasis( eqs, plex(Z,t,X,Y) ):
```

```
> select(v->has(v,[X,Y]) and not(has(v,[t,Z])),gb);
```

$$[X^4 + 2\,X^2\,Y^2 - 3\,Y\,X^2 + Y^4 + Y^3]$$

"This is based on the theorem which says that the elements of the lex grobner basis that do not contain the auxiliary variables t,Z are a basis of the implicit ideal." Ilias Kotsireas, Sept 15 2000.

A new method, using linear algebra techniques, *and which works even for non-rational parameterizations* , runs as follows.

```
> with(LinearAlgebra):
```

We first assume that we know (or are progressing to compute) the degree of the implicitization. Here we simply plop down all possible terms up to degree 4.

```
> M:=(x,y) -> Vector(15,[1, x, y, x^2, x*y, y^2,x^3,x^2*y,x*y^2,y^3,x^4,x^3*y,x^2*y^2,x*y^3,y^4]);
```

$$M := (x, y) \to \mathrm{Vector}(15, [1, x, y, x^2, x\,y, y^2, x^3, x^2\,y, x\,y^2, y^3, x^4, x^3\,y, x^2\,y^2, x\,y^3, y^4])$$

We make a matrix out of these terms:

```
> M1:= M(xp(t),yp(t)) . Transpose(M(xp(t),yp(t)));
```

$$M1 := \begin{bmatrix} \text{15 x 15 Matrix} \\ \text{Data Type: anything} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{bmatrix}$$

Now we integrate those functions of t over some random interval.

```
> # M3:=map(evalf,map(Int, M1, t=-10..10));
```

Doing it that way, by numerical integration, spends a lot of time doing integration when just sampling is really good enough. We take twice as many samples as we need, to avoid sensitivity of interpolation.

```
> UseHardwareFloats := false;
```

$$\mathit{UseHardwareFloats} := \mathit{false}$$

```
> Digits := 50;
```

$$\mathit{Digits} := 50$$

```
> Mfinite := Matrix(15,15):
```

```
> Npoints := 30:
```

```
> for i to Npoints do
ti := -10. + 20.*(i-1)/(Npoints-1);
```

```
xpt := xp(ti);
ypt := yp(ti);
Mfinite := Mfinite + M(xpt,ypt).Transpose(M(xpt,ypt));
od:

> Ufinite, Sfinite, Vtfinite := SingularValues(Mfinite,output=['U','S','Vt']);
```

$$Ufinite, Sfinite, Vtfinite := \left[\begin{array}{l}\text{15 x 15 Matrix}\\ \text{Data Type: sfloat}\\ \text{Storage: rectangular}\\ \text{Order: Fortran\_order}\end{array}\right], \left[\begin{array}{l}\text{15 Element Column Vector}\\ \text{Data Type: sfloat}\\ \text{Storage: rectangular}\\ \text{Order: Fortran\_order}\end{array}\right], \left[\begin{array}{l}\text{15 x 15 Matrix}\\ \text{Data Type: sfloat}\\ \text{Storage: rectangular}\\ \text{Order: Fortran\_order}\end{array}\right]$$

```
>

> # U,S,Vt := SingularValues(M3, output=['U','S','Vt']);

> Sfinite[1];
```

$$69.368557335845072700594586696530893043127415072444$$

```
> Sfinite[14];
```

$$0.37466187911174901873340723981275715856592199368922\ 10^{-8}$$

```
> Sfinite[15];
```

$$0.29239670008727131382117935533259132750218308851330\ 10^{-49}$$

```
> nv := Vector(15,[seq(Vtfinite[15,i],i=1..15)]);
```

$$nv := \left[\begin{array}{l}\text{15 Element Column Vector}\\ \text{Data Type: anything}\\ \text{Storage: rectangular}\\ \text{Order: Fortran\_order}\end{array}\right]$$

```
> ex:=fnormal( Transpose(M(x,y)).nv ,6);
```

$$ex := 0.750000\, x^2\, y - 0.250000\, y^3 - 0.250000\, x^4 - 0.500000\, x^2\, y^2 - 0.250000\, y^4$$

```
> p := fnormal(ex/coeff(ex,y,4),6);
```

$$p := -3.00000\, x^2\, y + 1.00000\, y^3 + 1.00000\, x^4 + 2.00000\, x^2\, y^2 + 1.00000\, y^4$$

```
> simplify(subs(x=xp(t),y=yp(t),p));
```

$$0.$$

Schoenhage's GCD example

```
> restart;
```

```
> f := x^4 + x + 1;
```

$$f := x^4 + x + 1$$

```
> g := x^3 + eta*x;
```

$$g := x^3 + \eta\, x$$

```
> S := LinearAlgebra[SylvesterMatrix]( f, g, x );
```

$$S := \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & \eta & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & \eta & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \eta & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \eta & 0 \end{bmatrix}$$

```
> Sinv := LinearAlgebra[MatrixInverse](S):
```

```
> Sinvs := map(series, Sinv, eta, 3 );
```

$Sinvs :=$

$[\eta^2 + O(\eta^3), O(\eta^3), 0, 1 - \eta^2 + O(\eta^3), O(\eta^3), -\eta + O(\eta^3), -\eta^2 + O(\eta^3)]$

$[-\eta^2 + O(\eta^3), \eta^2 + O(\eta^3), 0, \eta^2 + O(\eta^3), 1 - \eta^2 + O(\eta^3), O(\eta^3), -\eta + \eta^2 + O(\eta^3)]$

$[-\eta + \eta^2 + O(\eta^3), -\eta^2 + O(\eta^3), 0, \eta - \eta^2 + O(\eta^3), \eta^2 + O(\eta^3), 1 - \eta^2 + O(\eta^3),$
$\eta - \eta^2 + O(\eta^3)]$

$[\eta - \eta^2 + O(\eta^3), -\eta + \eta^2 + O(\eta^3), 0, -\eta + \eta^2 + O(\eta^3), \eta - \eta^2 + O(\eta^3), \eta^2 + O(\eta^3),$
$1 - \eta + O(\eta^3)]$

$[1 - \eta + O(\eta^3), \eta - \eta^2 + O(\eta^3), 0, -1 + \eta + O(\eta^3), -\eta + \eta^2 + O(\eta^3), \eta - \eta^2 + O(\eta^3),$
$-1 + \eta + \eta^2 + O(\eta^3)]$

$[-1 + \eta + \eta^2 + O(\eta^3), 1 - \eta + O(\eta^3), 0, 1 - \eta - \eta^2 + O(\eta^3), -1 + \eta + O(\eta^3),$
$-\eta + \eta^2 + O(\eta^3), 1 - 2\,\eta^2 + O(\eta^3)]$

$[1 - 2\,\eta^2 + O(\eta^3), -1 + \eta + \eta^2 + O(\eta^3), 1, -1 + 2\,\eta^2 + O(\eta^3), 1 - \eta - \eta^2 + O(\eta^3),$

$$-1+\eta+O(\eta^3),\ -1-\eta+3\eta^2+O(\eta^3)]$$

```
> Si0 := map(coeff, Sinvs, eta, 0 );
```

$$Si0 := \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 & 0 & 0 & -1 \\ -1 & 1 & 0 & 1 & -1 & 0 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & -1 \end{bmatrix}$$

```
> Si1 := map(coeff, Sinvs, eta, 1 );
```

$$Si1 := \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & -1 & 0 & -1 & 1 & 0 & -1 \\ -1 & 1 & 0 & 1 & -1 & 1 & 1 \\ 1 & -1 & 0 & -1 & 1 & -1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 1 & -1 \end{bmatrix}$$

```
> resultant(f,g,x);
```

$$\eta^4+2\eta^2+\eta+1$$

```
> fsolve(%,eta,complex);
```

$$-0.3438145972 - 0.6253578118\, I, -0.3438145972 + 0.6253578118\, I,$$
$$0.3438145972 - 1.358434600\, I, 0.3438145972 + 1.358434600\, I$$

```
> map(abs,[%]);
```

$$[0.7136391736, 0.7136391736, 1.401268368, 1.401268368]$$

```
> with(LinearAlgebra):
```

```
> SingularValues(evalf(eval(S,eta=0)));
```

$$\begin{bmatrix} 2.18070725880909676 \\ 1.89161800512635268 \\ 1.55592692305565650 \\ 1.00000000000000044 \\ 0.840671853490929100 \\ 0.681727563264314607 \\ 0.271858324138691210 \end{bmatrix}$$

```
> %[7]/%[1];
```

$$0.1246652080$$

So, for small but nonzero eta, the nearest singular Sylvester matrix is at least 0.124 away in norm; that is, the nearest pair of polynomials f+deltaf and g + deltag that have a nontrivial GCD must have norms of delta g and delta f of about 0.1.

Therefore, for small but nonzero eta, the GCD of f and g is 1, and moreover the *approximate gcd* is also 1, for tolerances smaller than about 0.1.

But if we run Euclid's algorithm on this problem:

```
> r1 := rem( f, g, x );
```

$$r1 := 1 + x - x^2 \eta$$

```
> r2 := rem( g, r1, x );
```

$$r2 := \frac{(1 + \eta + \eta^3)\, x}{\eta^2} + \frac{1}{\eta^2}$$

```
> r3 := rem( r1, r2, x );
```

$$r3 := \frac{\eta^2 (\eta^4 + 2\,\eta^2 + \eta + 1)}{(1 + \eta + \eta^3)^2}$$

Note carefully: r1 and r2 have a nontrivial approximate GCD if eta is small!

```
> solve(r1,x);
```

$$\frac{1}{2}\,\frac{1 + \sqrt{1 + 4\,\eta}}{\eta},\ \frac{1}{2}\,\frac{1 - \sqrt{1 + 4\,\eta}}{\eta}$$

```
> map(series,[%],eta,4);
```

$$[\eta^{-1} + 1 - \eta + 2\,\eta^2 + \mathrm{O}(\eta^3),\ -1 + \eta - 2\,\eta^2 + \mathrm{O}(\eta^3)]$$

```
> solve(r2,x);
```

$$-\frac{1}{1 + \eta + \eta^3}$$

```
> series(%,eta,4);
```

$$-1 + \eta - \eta^2 + 2\,\eta^3 + \mathrm{O}(\eta^4)$$

So r1 and r2 have, to O(eta^2), a common root. Therefore, a small perturbation of r1 or of r2 (on the order of eta^2) will bring them to having an exact common root.

Theorem : A step of the Euclidean Algorithm does not necessarily preserve approximate gcd.

```
>
```

A Geometric Intersection Problem

```
> restart;
```

```
> with(Groebner);
```

[*MulMatrix*, *SetBasis*, *fglm*, *gbasis*, *gsolve*, *hilbertdim*, *hilbertpoly*, *hilbertseries*, *inter_reduce*, *is_finite*, *is_solvable*, *leadcoeff*, *leadmon*, *leadterm*, *normalf*, *pretend_gbasis*, *reduce*, *spoly*, *termorder*, *testorder*, *univpoly*]

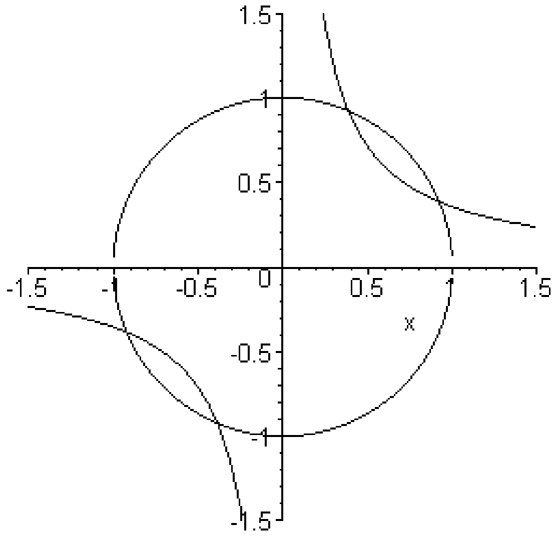
```
> f := x^2 + y^2 - 1;
```

$$f := x^2 + y^2 - 1$$

```
> g := x*y-5/13*12/13;
```

$$g := x\,y - \frac{60}{169}$$

```
> plot({sqrt(1-x^2),-sqrt(1-x^2),60/169/x},x=-1.5..1.5, view=[-1.5..1.5,-1.5..1.5],colour=black,scaling=CONSTRAINED);
```



```
> gb := gbasis([f,g],tdeg(x,y));
```

$$gb := [169\,x\,y - 60,\, x^2 + y^2 - 1,\, 60\,x + 169\,y^3 - 169\,y]$$

```
> ns,rv := SetBasis( gb, tdeg(x,y) );
```

$$ns, rv := [1, y, x, y^2], \mathrm{table}([1 = 1, x = 3, y = 2, y^2 = 4])$$

```
> Mx := MulMatrix(x,ns,rv,gb,tdeg(x,y));
```

$$Mx := \begin{bmatrix} 0 & 0 & 1 & 0 \\ \frac{60}{169} & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & \frac{60}{169} & 0 & 0 \end{bmatrix}$$

```
> My := MulMatrix(y,ns,rv,gb,tdeg(x,y));
```

$$My := \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{60}{169} & 0 & 0 & 0 \\ 0 & 1 & \frac{-60}{169} & 0 \end{bmatrix}$$

```
> Mx.My - My.Mx;
```

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

```
> with(LinearAlgebra):

> Px := JordanForm( Mx, output='Q');
```

$$Px := \begin{bmatrix} \frac{-25}{238} & \frac{-25}{238} & \frac{72}{119} & \frac{72}{119} \\ \frac{150}{1547} & \frac{-150}{1547} & \frac{-360}{1547} & \frac{360}{1547} \\ \frac{125}{3094} & \frac{-125}{3094} & \frac{-864}{1547} & \frac{864}{1547} \\ \frac{-1800}{20111} & \frac{-1800}{20111} & \frac{1800}{20111} & \frac{1800}{20111} \end{bmatrix}$$

```
> xvals := Px^(-1).Mx.Px;
```

$$xvals := \begin{bmatrix} \frac{-5}{13} & 0 & 0 & 0 \\ 0 & \frac{5}{13} & 0 & 0 \\ 0 & 0 & \frac{-12}{13} & 0 \\ 0 & 0 & 0 & \frac{12}{13} \end{bmatrix}$$

```
> yvals := Px^(-1).My.Px;
```

$$yvals := \begin{bmatrix} \frac{-12}{13} & 0 & 0 & 0 \\ 0 & \frac{12}{13} & 0 & 0 \\ 0 & 0 & \frac{-5}{13} & 0 \\ 0 & 0 & 0 & \frac{5}{13} \end{bmatrix}$$

```
> answers := [seq([xvals[i,i],yvals[i,i]],i=1..4)];
```

$$answers := \left[\left[\frac{-5}{13}, \frac{-12}{13}\right], \left[\frac{5}{13}, \frac{12}{13}\right], \left[\frac{-12}{13}, \frac{-5}{13}\right], \left[\frac{12}{13}, \frac{5}{13}\right]\right]$$

```
> for i to 4 do
subs( x=answers[i][1], y=answers[i][2], [f,g] );
od;
```

$$[0, 0]$$

$$[0, 0]$$

$$[0, 0]$$

$$[0, 0]$$

```
>
```

A Lagrange Multiplier Problem

```
> restart;
```

First load in the routines.

```
> with(Groebner);
```

$$[\mathit{MulMatrix}, \mathit{SetBasis}, \mathit{fglm}, \mathit{gbasis}, \mathit{gsolve}, \mathit{hilbertdim}, \mathit{hilbertpoly}, \mathit{hilbertseries}, \mathit{inter\_reduce}, \mathit{is\_finite}, \mathit{is\_solvable}, \mathit{leadcoeff}, \mathit{leadmon}, \mathit{leadterm}, \mathit{normalf}, \mathit{pretend\_gbasis}, \mathit{reduce}, \mathit{spoly}, \mathit{termorder}, \mathit{testorder}, \mathit{univpoly}]$$

```
> f := x^3 + 2*x*y*z - z^2:
```

```
> g := x^2 + y^2 + z^2 - 1:
```

```
> F := f + lambda*g;
```

$$F := x^3 + 2\,x\,y\,z - z^2 + \lambda\,(x^2 + y^2 + z^2 - 1)$$

```
> gF := convert(linalg[grad](F, [lambda, x, y, z]),set);
```

$$gF := \{x^2 + y^2 + z^2 - 1, 3\,x^2 + 2\,y\,z + 2\,\lambda\,x, 2\,x\,z + 2\,\lambda\,y, 2\,x\,y - 2\,z + 2\,\lambda\,z\}$$

```
> gb := gbasis(gF, tdeg(lambda,x,y,z)):
```

Now call SetBasis to get the normal set of this Groebner basis.

```
> ns, rv := SetBasis(gb, tdeg(lambda,x,y,z)):
```

```
> ns;
```

$$[1, z, y, x, \lambda, z^2, y\,z, y^2, x\,z, \lambda\,z, \lambda^2, z^3]$$

We don't wish to look at these 12 by 12 matrices here, but we will verify that the matrices commute.

```
> M[x] := MulMatrix(x, ns, rv, gb, tdeg(lambda,x,y,z)):
```

```
> M[y] := MulMatrix(y, ns, rv, gb, tdeg(lambda,x,y,z)):
```

```
> M[z] := MulMatrix(z, ns, rv, gb, tdeg(lambda,x,y,z)):
```

```
> M[lambda] := MulMatrix(lambda, ns, rv, gb, tdeg(lambda,x,y,z)):
```

```
> with(LinearAlgebra):
```

```
> Norm( M[x].M[y]-M[y].M[x], infinity);
```

$$0$$

```
> alpha := RandomVector(4);
```

$$\alpha := \begin{bmatrix} 62 \\ -79 \\ -71 \\ 28 \end{bmatrix}$$

```
> Digits := trunc(evalhf(Digits));
```

$$Digits := 14$$

```
> alpha := evalf(alpha/Norm(alpha,2));
```

$$\alpha := \begin{bmatrix} 0.49153743592058 \\ -0.62631382964075 \\ -0.56288964436066 \\ 0.22198464848026 \end{bmatrix}$$

Since the matrices commute, we know that there exists a unitary matrix V which simultaneously upper-triangularizes all four matrices. However, multiple roots cause a problem, and it is often the case (and is the case here) that because of symmetries in the problem, each individual coordinate matrix may have multiple roots --- but overall, the system has only single roots (think of x^2+y^2-1 and (x+y)(x-y)=12*5/13^2). A good way to ensure that only genuine multiplicity causes a problem is to take a generic linear combination of the multiplication matrices and use that as the basis of the eigenvalue computation.

```
> M[combination] := alpha[1]*M[x] + alpha[2]*M[y] + alpha[3]*M[z] + alpha[4]*M[lambda];
```

$$M_{combination} := \left[\begin{array}{l} \text{12 x 12 Matrix} \\ \text{Data Type: float[8]} \\ \text{Storage: sparse} \\ \text{Order: Fortran\_order} \end{array}\right]$$

```
> P[lambda] := JordanForm(M[combination],output='Q');
```

$$P_{\lambda} := \left[\begin{array}{l} \text{12 x 12 Matrix} \\ \text{Data Type: complex[8]} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{array}\right]$$

```
> xvals := P[lambda]^(-1).M[x].P[lambda];
```

$$xvals := \left[\begin{array}{l} \text{12 x 12 Matrix} \\ \text{Data Type: complex[8]} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{array}\right]$$

```
> yvals := P[lambda]^(-1).M[y].P[lambda];
```

$$yvals := \left[\begin{array}{l} \text{12 x 12 Matrix} \\ \text{Data Type: complex[8]} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{array}\right]$$

```
> zvals := P[lambda]^(-1).M[z].P[lambda];
```

$$zvals := \left[\begin{array}{l} \text{12 x 12 Matrix} \\ \text{Data Type: complex[8]} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{array}\right]$$

```
> lambdavals := P[lambda]^(-1).M[lambda].P[lambda];
```

$$lambdavals := \left[\begin{array}{l} \text{12 x 12 Matrix} \\ \text{Data Type: complex[8]} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{array}\right]$$

```
> answers := map(fnormal, [seq( [xvals[i,i], yvals[i,i], zvals[i,i], lambdavals[i,i]], i=1..12)]);
```

$answers := [[0. + 0.\,I, 1.0000000000000 - 0.\,I, 0. - 0.\,I, -0. + 0.\,I], [$
$-0.66666666666671 - 0.\,I, 0.33333333333335 - 0.\,I, 0.66666666666671 + 0.\,I,$
$1.3333333333334 + 0.\,I], [-0.37500000000000 + 0.\,I, 0.87945295496689 - 0.\,I,$
$-0.29315098498896 - 0.\,I, -0.12500000000000 + 0.\,I], [-0.87726260786236\ 10^{-6} - 0.\,I,$
$0.87724533767708\ 10^{-6} + 0.\,I, 1.0000000051223 - 0.\,I, 1.0000000051223 - 0.\,I], [$
$0.87725963535124\ 10^{-6} + 0.\,I, -0.87725421718532\ 10^{-6} - 0.\,I, 0.99999999420834 + 0.\,I,$
$0.99999999423744 + 0.\,I],$
$[-0.99999999999984 + 0.\,I, 0. - 0.\,I, -0. - 0.\,I, 1.4999999999998 - 0.\,I],$
$[0.99999999999999 - 0.\,I, 0. + 0.\,I, -0. - 0.\,I, -1.5000000000000 + 0.\,I], [$
$-0.37500000000000 + 0.\,I, -0.87945295496690 + 0.\,I, 0.29315098498897 - 0.\,I,$
$-0.12500000000000 + 0.\,I], [-0.66666666666665 + 0.\,I, -0.33333333333333 + 0.\,I,$
$-0.66666666666665 - 0.\,I, 1.3333333333333 - 0.\,I],$
$[-0. + 0.\,I, -0.99999999999996 + 0.\,I, -0. - 0.\,I, 0. + 0.\,I], [-0. + 0.25462547418217\ 10^{-5}\,I,$

$0. + 0.25462821565426\ 10^{-5}\ I, -1.0000000000000 - 0.26193447411060\ 10^{-9}\ I,$
$1.0000000000000 + 0.23283064365387\ 10^{-9}\ I], [0. - 0.25462741580396\ 10^{-5}\ I,$
$-0. - 0.25462954413975\ 10^{-5}\ I, -0.99999999999994 - 0.14551915228367\ 10^{-9}\ I,$
$0.99999999999994 + 0.12369127944112\ 10^{-9}\ I]]$

```
> for i to 12 do
map(abs, eval(gF, {x=xvals[i,i], y=yvals[i,i], z=zvals[i,i], lambda=lambdavals[i,i]} ));
od;
```

$\{0.54517742065384\ 10^{-16}, 0.46712730736440\ 10^{-15}, 0.10985930471924\ 10^{-16},$
$0.18934019769510\ 10^{-15}\}$

$\{0.10801724242681\ 10^{-18}, 0.10000000000188\ 10^{-12}, 0.20000000013204\ 10^{-13},$
$0.12000000000132\ 10^{-12}\}$

$\{0.17842077965328\ 10^{-17}, 0.16102828341556\ 10^{-17}, 0.13000000043992\ 10^{-13},$
$0.19246889726328\ 10^{-17}\}$

$\{0.32231601700098\ 10^{-10}, 0.102431\ 10^{-7}, 0.34540370700092\ 10^{-10}, 0.10246140000004\ 10^{-7}\}$

$\{0.10836280700295\ 10^{-10}, 0.13145136300243\ 10^{-10}, 0.115266\ 10^{-7}, 0.11581780000004\ 10^{-7}\}$

$\{0.20000000000008\ 10^{-12}, 0.50642870042828\ 10^{-17}, 0.25557085040334\ 10^{-17},$
$0.32000000000005\ 10^{-12}\}$

$\{0.57868199419788\ 10^{-16}, 0.20000068257403\ 10^{-13}, 0.14177296561968\ 10^{-15},$
$0.10000001364867\ 10^{-12}\}$

$$\{0.20000000017572\ 10^{-13}, 0.13000000177584\ 10^{-13}, 0.20000008747026\ 10^{-14}, 0.83850489826584\ 10^{-18}\}$$

$$\{0.17636111045087\ 10^{-18}, 0.50000000000319\ 10^{-13}, 0.20000000000797\ 10^{-13}, 0.17404700375555\ 10^{-18}\}$$

$$\{0.13884668114857\ 10^{-17}, 0.50100693309873\ 10^{-15}, 0.80000000010506\ 10^{-13}, 0.15357808216608\ 10^{-17}\}$$

$$\{0.52403022327030\ 10^{-9}, 0.46584271434219\ 10^{-9}, 0.54829442027463\ 10^{-10}, 0.58178709875106\ 10^{-10}\}$$

$$\{0.42566715967903\ 10^{-10}, 0.29133387912212\ 10^{-9}, 0.24771867196483\ 10^{-9}, 0.46798613658913\ 10^{-10}\}$$

```
> Fvals := Vector(12):
```

```
> for i to 12 do
Fvals[i] := fnormal(eval( F, {x=xvals[i,i], y=yvals[i,i], z=zvals[i,i], lambda=lambdavals[i,i]} ));
od;
```

$$Fvals_1 := 0. + 0.\ I$$

$$Fvals_2 := -1.0370370370370 + 0.\ I$$

$$Fvals_3 := 0.054687500000005 + 0.\ I$$

$$Fvals_4 := -0.99999999999996 - 0.\ I$$

$$Fvals_5 := -1.0000000000000 - 0.\ I$$

$$Fvals_6 := -1.0000000000000 + 0.\ I$$

$$Fvals_7 := 1.0000000000000 - 0.\ I$$

$$Fvals_8 := 0.054687500000005 + 0.\ I$$

$$Fvals_9 := -1.0370370370371 + 0.\ I$$

$$Fvals_{10} := -0. - 0.\ I$$

$$Fvals_{11} := -1.0000000000000 - 0.\ I$$

$$Fvals_{12} := -1.0000000000000 + 0.\ I$$

```
> min(seq(Re(Fvals[i]),i=1..12));
```

$$-1.0370370370371$$

```
> answers[8];
```

$$[-0.37500000000000 + 0.\ I, -0.87945295496690 + 0.\ I, 0.29315098498897 - 0.\ I, -0.12500000000000 + 0.\ I]$$

```
> eval(F,[x=-2/3,y=1/3,z=2/3,lambda=4/3]);
```

$$\frac{-28}{27}$$

```
> evalf(%);
```

$$-1.0370370370370$$

```
> max(seq(Re(Fvals[i]),i=1..12));
```

$$1.0000000000000$$

```
>
```

An example from Li, Sauer, and Yorke ``The Cheater's Homotopy''

```
> restart;
```

A problem with free parameters. This is taken from

the paper ``The Cheater's Homotopy'' by T. Y. Li, Tim Sauer, and J. A.

Yorke, from the SIAM J. Numerical Analysis, Volume 26, No. 5, pp. 1241-1251, October 1989. This problem cannot be done by a pure lexicographic ordered Groebner basis (the answer is just too complicated to be of any use even if we could calculate it in under two hours and 267 Mbytes).

But the approach here works in under a minute on a 486, with only a small

amount of memory.

```
> LSY[1] := x^3*y^2+c1*x^3*y+y^2+c2*x+c3;
```

$$LSY_1 := x^3 y^2 + c1\, x^3 y + y^2 + c2\, x + c3$$

```
> LSY[2] := c4*x^4*y^2-x^2*y+y+c5;
```

$$LSY_2 := c4\, x^4 y^2 - x^2 y + y + c5$$

```
> with(Groebner):
```

The following step does not succeed if you ask for a plex order basis.

Since this problem has free parameters, and Groebner bases can be discontinuous with respect to parameters, we have to be a bit careful. The Maple computation is guaranteed to be correct only if the sequence of leading coefficients of the result are nonzero, and if the sequence of gcd's removed in the computation of the primitive parts as the computation progresses is nonzero. To find out about that last sequence, we must use an undocumented feature, namely

```
> infolevel[primpart] := 1;
```

$$infolevel_{primpart} := 1$$

```
> gb := gbasis({LSY[1],LSY[2]},tdeg(x,y)):
```

```
Groebner/primpartscale:   remove content   1

Groebner/primpartscale:   remove content   1

Groebner/primpartscale:   remove content   1

Groebner/primpartscale:   remove content   1

Groebner/primpartscale:   remove content   1

Groebner/primpartscale:   remove content   1

Groebner/primpartscale:   remove content   1

Groebner/primpartscale:   remove content   1

Groebner/primpartscale:   remove content   1

Groebner/primpartscale:   remove content   1

Groebner/primpartscale:   remove content   66

Groebner/primpartscale:   remove content   1

Groebner/primpartscale:   remove content   1840712798943858272237612105371699316541728043

Groebner/primpartscale:   remove content   c3^2+2*c3*c1^2+c1^4

Groebner/primpartscale:   remove content
10728291952625167346729110632599977987582603096873725819108467215868918194062213685034046431801554686779885909220864

Groebner/primpartscale:   remove content   c4^2*c1^7*c5+3*c4^2*c1^5*c5*c3+3*c1^3*c4^2*c5*c3^2+c1*c4^2*c5*c3^3+2*c4^2*c1^8+6*c4^2*c1^6*c3
+6*c4^2*c1^4*c3^2+2*c4^2*c1^2*c3^3-c1^7*c4-3*c1^5*c4*c2-3*c1^5*c4*c3-6*c1^3*c4*c3*c2-3*c1^3*c4*c3^2-3*c1*c4*c2*c3^2-c1*c4*c3^3-
c1^4*c4*c2*c5-2*c1^2*c4*c3*c2*c5-c2*c3^2*c4*c5+c1^4*c2+c1^2*c2^2+2*c1^2*c3*c2+c3*c2^2+c2*c3^2

Groebner/primpartscale:   remove content
103077198649121846052949180212963218623589713585678274405982928270910334890984846313914435688963200802577570247857736242879359562410393600

Groebner/primpartscale:   remove content   -4*c1^2*c4^2*c3^3+c3^3*c5^2*c4^3*c1-2*c3^3*c5*c4^2*c1+12*c1^4*c4^3*c5*c3^2+4*c1^9*c4^3
+c1^7*c5^2*c4^3-4*c1^8*c4^2+c1^7*c4+c1*c4*c3^3+3*c1^3*c4*c3^2+3*c1^5*c4*c3+12*c4^3*c1^7*c3+4*c1^3*c4^3*c3^3-12*c1^6*c4^2*c3-
12*c1^4*c4^2*c3^2+12*c4^3*c1^5*c3^2-2*c1^7*c5*c4^2-4*c1^3*c5*c3*c4^2*c2-2*c1^5*c5*c4^2*c2+12*c1^6*c5*c4^3*c3+4*c1^2*c5*c4^3*c3^3
+4*c1^8*c5*c4^3+3*c1^5*c5^2*c4^3*c3-6*c1^3*c5*c4^2*c3^2-6*c1^5*c5*c4^2*c3-2*c2*c1*c4^2*c5*c3^2+2*c2*c1^5*c4+2*c2*c1*c4*c3^2
+4*c2*c1^3*c4*c3-4*c2*c1^6*c4^2+c1^3*c4*c2^2+c3*c4*c1*c2^2+3*c1^3*c4^3*c3^2*c5^2-4*c2*c1^2*c4^2*c3^2-8*c2*c1^4*c4^2*c3
```

```
Groebner/primpartscale:   remove content   2*c4*c1^3+c4*c1^2*c5-c1^2+2*c1*c4*c3+c3*c5*c4-c2-c3

Groebner/primpartscale:   remove content   1

Groebner/primpartscale:   remove content   -c4*c3^2+2*c1^5*c4^2-c4*c1^4-2*c4*c1^2*c3+c4^2*c3^2*c5+2*c3*c4^2*c1^2*c5+c4^2*c1^4*c5-c2*c4*c3
+4*c4^2*c1^3*c3-c4*c1^2*c2+2*c1*c4^2*c3^2

Groebner/primpartscale:   remove content   1

Groebner/primpartscale:   remove content   2*c4*c1^3+c4*c1^2*c5-c1^2+2*c1*c4*c3+c3*c5*c4-c2-c3

Groebner/primpartscale:   remove content   1

Groebner/primpartscale:   remove content   1

Groebner/primpartscale:   remove content   1

Groebner/primpartscale:   remove content   1

Groebner/primpartscale:   remove content   1
```

```
> leadcoeffs := map(leadcoeff, gb, tdeg(x,y) );
```

$$\mathit{leadcoeffs} := [c4^2\, c1\, (c4^2\, c2\, c1^2 + c1\, c5 + c4^2\, c2\, c3 + c2),\\ c4\, c1\, (c4^2\, c2\, c1^2 + c1\, c5 + c4^2\, c2\, c3 + c2), c1\, (c4^2\, c2\, c1^2 + c1\, c5 + c4^2\, c2\, c3 + c2),\\ c1\, (c4^2\, c2\, c1^2 + c1\, c5 + c4^2\, c2\, c3 + c2), c4\, c1\, (c1^4\, c4^2\, c5\, c2 + c4^2\, c1^3\, c2^2 + c1^3\, c5^2\\ + c1^2\, c4^3\, c2^3 + c3\, c4^2\, c2\, c1^2\, c5 + 2\, c2\, c1^2\, c5 + c1\, c4^2\, c3\, c2^2 + c1\, c4\, c5\, c2^2 + c1\, c2^2\\ + c2^3\, c4^3\, c3 + c2^3\, c4)]$$

We will assume henceforth (for this exposition) that all of these polynomials in the parameters are nonzero for the parameter values we will use.

```
> ns, rv := SetBasis(gb, tdeg(x,y)):

> ns;
```

$$[1, y, x, y^2, x\, y, x^2, y^3, x\, y^2, x^2\, y, x^3]$$

```
> nops(ns);
```

$$10$$

```
> infolevel[primpart] := 0;
```

$$infolevel_{primpart} := 0$$

```
> M_x := MulMatrix(x, ns, rv, gb, tdeg(x,y)):
```

The entries of these sparse matrices are rational functions in the parameters c1 through c5. For example,

```
> M_x[10,3];
```

$$\begin{aligned}&(c4^2\,c3^2\,c2^2+2\,c1^2\,c5^2\,c3+c2^2\,c3+c1^3\,c5\,c3-c1^2\,c4^2\,c3^2\,c2+c1^2\,c4\,c5\,c3^2\\&+2\,c1^2\,c4^2\,c2^2\,c3+2\,c1^2\,c4\,c5\,c2^2+c1\,c4\,c2^3+2\,c3\,c2\,c1\,c5+c4\,c1\,c3^3+c4\,c1^3\,c2^2\\&+2\,c1\,c4^3\,c2^3\,c3+2\,c3^2\,c5\,c4^2\,c2\,c1+c1^3\,c4^3\,c3^3+c4^3\,c3^4\,c1)\Big/(c1\,(c1^4\,c4^2\,c5\,c2\\&+c4^2\,c1^3\,c2^2+c1^3\,c5^2+c1^2\,c4^3\,c2^3+c3\,c4^2\,c2\,c1^2\,c5+2\,c2\,c1^2\,c5+c1\,c4^2\,c3\,c2^2\\&+c1\,c4\,c5\,c2^2+c1\,c2^2+c2^3\,c4^3\,c3+c2^3\,c4))\end{aligned}$$

```
> M_y := MulMatrix(y, ns, rv, gb, tdeg(x,y)):
```

At this point, one may insert numerical values for c1 through c5 and find eigenvalues of a generic (convex random) combination of these two matrices, cluster any multiple roots, and use the Schur vectors to find the roots of the system. We see that there are generically 10 roots. We take random values for the parameters below, as an example, ignoring the possibility of multiple roots.

```
> c1 := rand()/10.^12:
c2 := rand()/10.^12:
c3 := rand()/10.^12:
c4 := rand()/10.^12:
c5 := rand()/10.^12:
```

```
> Digits := trunc(evalhf(Digits)):
with(LinearAlgebra):
```

```
> M_x_f := map(eval,M_x):
```

```
> xvals, V := Eigenvectors(evalf(M_x_f)):
```

```
> yvalmtx := map(fnormal, V^(-1) . evalf(map(eval,M_y)) . V ):
```

```
> yvals := [seq(yvalmtx[i,i],i=1..10)]:
```

Substitute the computed values of x and y into the original equations, to see how nearly the computed quantities satisfy the original equations. To know how accurate our computed x and y are, we need more than just these residuals; we should look at how perturbations in these polynomials affect the roots. We do not do that here.

```
> residuals := [seq(subs(x=xvals[i],y=yvals[i],{LSY[1],LSY[2]}),i=1..10)];
```

$$\mathit{residuals} := [\{0.10\ 10^{-12} + 0.14\ 10^{-12}\,I, -0.24\ 10^{-12} - 0.17\ 10^{-12}\,I\},$$
$$\{0.10\ 10^{-12} - 0.14\ 10^{-12}\,I, -0.24\ 10^{-12} + 0.17\ 10^{-12}\,I\},$$
$$\{0.11\ 10^{-12} - 0.9\ 10^{-13}\,I, 0.16\ 10^{-12} + 0.\,I\}, \{0.11\ 10^{-12} + 0.9\ 10^{-13}\,I, 0.16\ 10^{-12} + 0.\,I\},$$
$$\{-0.218\ 10^{-13} - 0.11\ 10^{-13}\,I, -0.3\ 10^{-13} - 0.63\ 10^{-13}\,I\},$$
$$\{-0.3\ 10^{-13} + 0.63\ 10^{-13}\,I, -0.217\ 10^{-13} + 0.11\ 10^{-13}\,I\},$$
$$\{-0.5\ 10^{-13} + 0.1\ 10^{-12}\,I, 0.3\ 10^{-13} + 0.20\ 10^{-13}\,I\},$$
$$\{-0.5\ 10^{-13} - 0.1\ 10^{-12}\,I, 0.3\ 10^{-13} - 0.20\ 10^{-13}\,I\}, \{0. - 0.9\ 10^{-14}\,I, 0. - 0.1\ 10^{-13}\,I\},$$
$$\{0. + 0.9\ 10^{-14}\,I, 0. + 0.1\ 10^{-13}\,I\}]$$

```
>
```

Alternatively one can use tdeg bases to explore the bifurcation behaviour of these equations; one can quite easily determine parameter sets that force double, triple, or even quadruple roots.

What's next?

We have currently made some progress on the problem of factorization of approximate polynomials. This is a challenge problem, as listed in Erich Kaltofen's recent J.Symb.Comp. paper ``Open Problems in Computer Algebra''.

A proper comparison of the methods of sparse resultants, homotopy methods, and eigenvalue methods for the solution of systems of polynomial equations needs to be done.

There is a great deal of ``cleaning up'' and implementing these SNAP algorithms; a coherent, robust package would be very useful.

Over the horizon, there is the problem of symmetries: a polynomial system may have all its solutions generated from a much smaller set by symmetries. An example of this is quadrature formulae: they can be formulated (by smart humans) as an eigenproblem with n eigenvalues, encapsulating the n! arrangements nicely; whilst a Groebner basis formulation must necessarily lead to a commuting eigenproblem of size n!, with eigenvalues determining each possible arrangement of the nodes. An automatic procedure to detect symmetries or approximate symmetries would be very useful indeed.