Groebner bases: What are they and what are they useful for?

Michael Monagan
Department of Mathematics
Simon Fraser University
mmonagan@sfu.ca

Bruno Buchberger defined Groebner bases in his PhD thesis in 1965. He named them after his PhD supervisor Wolfgang Gröbner. Buchberger gave an algorithm for computing them which is now known as ``Buchberger's algorithm''. Groebner bases have found many applications since. All general purpose computer algebra systems like Maple have Groebner basis implementations. There are many books that have been written on them. In 2007 Buchberger received an award from the Association for Computing Machinery (ACM) for his work recognizing the importance of this contribution to mathematics and computer science.

But what is a Groebner basis? And what applications do Groebner bases have? I'd like to begin this article by giving some examples of the main application of Groebner bases which is to solve systems of polynomial equations. Below are three systems of polynomial equations.

```
> sys1 := { x^2+y^2=1, x*y=z, z^2-x^2=1, y^2+z^2=2 };
```

$$sys1 := \{x\,y = z,\ x^2 + y^2 = 1,\ -x^2 + z^2 = 1,\ y^2 + z^2 = 2\}$$

```
> sys2 := { x^2+y^2=1, x*y+x*z+y*z = 1, x+y+z = 1, x*y*z = 1 };
```

$$sys2 := \{x\,y\,z = 1,\ x + y + z = 1,\ x^2 + y^2 = 1,\ x\,y + x\,z + y\,z = 1\}$$

```
> sys3 := { w+x+y+z=1, w-z=1, y+z=1, x+y+2*z=0, w+x=0  };
```

$$sys3 := \{w + x = 0,\ w - z = 1,\ y + z = 1,\ x + y + 2\,z = 0,\ w + x + y + z = 1\}$$

The first system is a system of 4 quadratic equations in 3 unknowns. The second system has one linear, two quadratic and one cubic equation in 3 unknowns. The third system is a system of 5 linear equations in 4 unknowns. Here are some basic questions we would typically ask about about systems of equations

Question 1: Does the system have any real or complex solutions? If yes, then how many?
Question 2: How do we solve the system?
Question 3: Can the system be simplified, that is, are any equations redundant?

We will answer these questions using Groebner bases. Instead of working with equations, we will work with polynomials and write the systems as a list of polynomials $B = [f_1, f_2, ..., f_s]$ where the polynomials are implicitly equated to 0. Let me do that for our three systems.

```
> B1 := [ x^2+y^2-2, x*y-z, z^2-x^2-1, y^2+z^2-3 ]:
> B2 := [ x^2+y^2-1, x*y+x*z+y*z-1, x+y+z-1, x*y*z-1 ]:
> B3 := [ w+x+y+z-1, y+z-1, w-z-1, x+y+2*z, w+x ]:
```

We will compute a Groebner basis for each system. And we'll impose lexicographical order

on terms of the polynomials - we will worry about that later. Here are the Groebner bases.

```
> G1 := Groebner[Basis]( B1, plex(x,y,z) );
```

$$G1 := [z^4 - 3\,z^2 + 3,\ y^2 + z^2 - 3,\ -y\,z^3 + 3\,x]$$

```
> G2 := Groebner[Basis]( B2, plex(x,y,z) );
```

$$G2 := [1]$$

```
> G3 := Groebner[Basis]( B3, plex(w,x,y,z) );
```

$$G3 := [y + z - 1,\ x + 1 + z,\ w - z - 1]$$

What is the relationship between a Groebner basis G and the input basis B?

Main property of Groebner bases: Let $B = [f_1, f_2, ..., f_s]$ be the input basis (a list of polynomials with real or complex coefficients) and let $G = [g_1, g_2, ..., g_t]$ be the output basis, the Groebner basis, also a list of polynomials.
Theorem: The real and complex solutions of the two systems $\{f_1 = 0, f_2 = 0, ..., f_s = 0\}$ and $\{g_1 = 0, g_2 = 0, ..., g_t = 0\}$ are identical, even up to multiplicity of solutions.

So what that means is that we can think of the Groebner basis as a simplified system. Computing a Groebner basis effectively simplifies the system. It doesn't ``solve the system''. Let's take a look at what it did for each system.

```
> G2;
```

$$[1]$$

This means that the second system of equations is equivalent to the system { 1 = 0 } which obviously has no solutions. In fact, it is true that the (reduced) Groebner basis $G = [1]$ if and only if the system $\{f_1 = 0, f_2 = 0, ..., f_s = 0\}$ has no solution. That's nice.

More good news. I can see that the Groebner basis G_1 for the first system has ``triangularized'' the system.

```
> G1;
```

$$[z^4 - 3\,z^2 + 3,\ y^2 + z^2 - 3,\ -y\,z^3 + 3\,x]$$

The first polynomial

```
> G1[1]=0;
```

$$z^4 - 3\,z^2 + 3 = 0$$

is in *z* only. So we solve it e.g. using solve

```
> solve(G1[1]=0,z);
```

$$\frac{1}{2}\sqrt{6 - 2\,I\sqrt{3}},\ -\frac{1}{2}\sqrt{6 - 2\,I\sqrt{3}},\ \frac{1}{2}\sqrt{6 + 2\,I\sqrt{3}},\ -\frac{1}{2}\sqrt{6 + 2\,I\sqrt{3}}$$

Now the second polynomial in G_1 is in *z* and *y* only so we now can solve that too.

```
> G1[2]=0;
```

$$y^2 + z^2 - 3 = 0$$

Since it is quadratic in *y* we will have 8 solutions in total so far. Finally the third equation is linear in *x* so trivially solved once we know *y* and *z*. Thus the first system has 8 complex solutions.

```
> G1[3]=0;
```

$$-y z^3 + 3 x = 0$$

What about the third system?

```
> sys3;
```

$$\{w + x = 0,\ w - z = 1,\ y + z = 1,\ x + y + 2 z = 0,\ w + x + y + z = 1\}$$

Let's solve that one using linear algebra with $w > x > y > z$.

The coefficient matrix A and the right hand side vector b are

```
> A,b := Matrix([[1,1,0,0],[1,0,0,-1],[0,0,1,1],[0,1,1,2],[1,1,1,1]]),
        Vector([0,1,1,0,1]);
```

$$A, b := \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Solving $Ax = b$ via reduction of the augmented matrix $[A|b]$ to reduced Row Echelon form

```
> with(LinearAlgebra):
  ReducedRowEchelonForm( <A|b> );
```

$$\begin{bmatrix} 1 & 0 & 0 & -1 & 1 \\ 0 & 1 & 0 & 1 & -1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Converting back to equations in w, x, y, z this corresponds to the reduced linear system

```
> [y+z=1, x+z=-1, w-z=1 ];
```

$$[y + z = 1,\ x + z = -1,\ w - z = 1]$$

which just happens to be the same as the Groebner basis!!

```
> G3;
```

$$[y + z - 1,\ x + 1 + z,\ w - z - 1]$$

Coincidence? Nope. It turns out that Row Echelon form for a linear system is a Groebner basis. And a Groebner basis is in Row Echelon form. And just as we can impose uniqueness on Row Echelon form we can do so on a Groebner basis too. We call it **the** reduced Groebner basis. Maple is automatically computing the reduced Groebner basis. In fact, the reduced Groebner basis for a linear system is the reduced Row Echelon form. So here is one way to understand Groebner bases: They are the natural generalization of Row Echelon form to non-linear systems.

Elimination

What else can you do with Groebner bases? I once joked with my class that you can do everything with Groebner bases. Of course that's not true but you can so a lot. Do you recall the rational parameterization of the circle? It is

$x(t) = \frac{2 \cdot t}{(1+t^2)}$ and $y(t) = \frac{(1-t^2)}{(1+t^2)}$ for $-\infty < t < +\infty$.

Clearing fractions we have $(1+t^2) \cdot x = 2 \cdot t$ and $(1+t^2) \cdot y = (1-t^2)$. We have two equations in three unknowns x, y, t. I would like to eliminate t to see what I get. Let's use Groebner bases. Converting to polynomials we have

```
> B := [ (1+t^2)*x - 2*t, (1+t^2)*y - (1-t^2) ];
```

$$B := [(t^2+1)\,x - 2\,t,\ (t^2+1)\,y + t^2 - 1]$$

To eliminate t we compute a Groebner basis with $t > x > y$ or $t > y > x$ to force elimination of t. We do this by specifying plex(t, x, y) or plex(t, y, x).

```
> Groebner[Basis]( B, plex(t,x,y) );
```

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

Well, look at that! Out popped the implicit equation for the circle. If you have variables x_1, x_2,..., t_1, t_2,... then to eliminate t_1, t_2,... use plex(t_1, t_2,..., x_1, x_2,...) .

Polynomial Long Division

Back to question 3. How do we test if there are any redundant equations in these systems. We would like to remove the redundant equations to simplify the system.

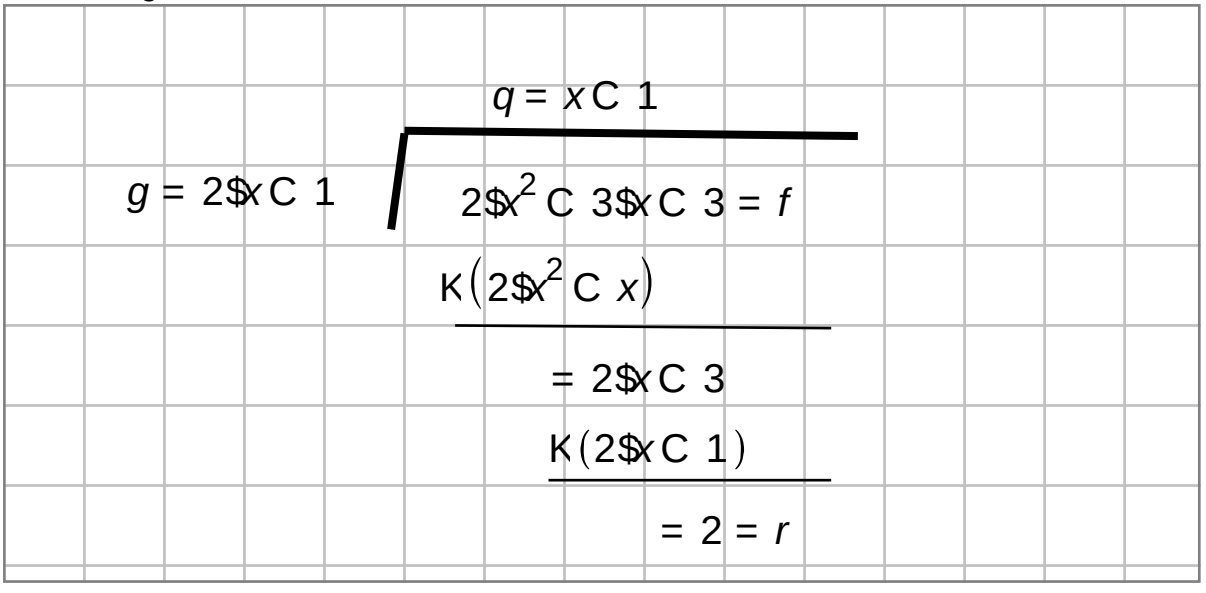
```
> sys1;
```

$$\{x\,y = z,\ x^2 + y^2 = 1,\ -x^2 + z^2 = 1,\ y^2 + z^2 = 2\}$$

```
> sys3;
```

$$\{w + x = 0,\ w - z = 1,\ y + z = 1,\ x + y + 2\,z = 0,\ w + x + y + z = 1\}$$

Let's recall division. Shown in the figure below (a Maple canvas, very useful for creating a figure) is a division of f divided g using the highschool long division algorithm as I remember seeing it.



We obtain the quotient $q = x + 1$ and remainder $r = 2$ satisfying $f = q \cdot g + r$. Have you every wondered why we always work with the terms sorted in decreasing degree? If you try executing the division algorithm with the terms sorted in increasing degree, you'll find that it works if the remainder $r = 0$ but doesn't terminate if $r \neq 0$. To make the division algorithm work for polynomials in more than one variable we will need to sort the terms.

But we will also need to be able to divide by more than one divisor. You may already seen this in linear algebra. Consider the four vectors

```
> u1,u2,u3,u4 := <1,1,0,0>, <1,0,0,-1>, <0,0,1,1>, <0,1,1,2>;
```

$$u1,\ u2,\ u3,\ u4 := \begin{bmatrix}1\\1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\\-1\end{bmatrix}, \begin{bmatrix}0\\0\\1\\1\end{bmatrix}, \begin{bmatrix}0\\1\\1\\2\end{bmatrix}$$

Suppose we are asked if there is a linear dependency between them? My son Philip is taking his first course in Linear Algebra at the University of British Columbia this Fall. He asked me how to do this last week! I had to try to remember. There is more than one way to do it. One way is to first see if $\{u_1, u_2, u_3\}$ are linearly independent. I'll do this using row Echelon form. First I put the vectors u_1^T, u_2^T, u_3^T as rows of a matrix then reduce it to Row Echelon form.

```
> A := Matrix( <Transpose(u1), Transpose(u2), Transpose(u3)> );
```

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

```
> R := ReducedRowEchelonForm( A );
```

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

This matrix has rank 3 so yes, u_1, u_2, u_3 are linearly independent. Now what about u_4? Is it also linearly independent? One way is to set up a linear system from $u_4 = a \cdot u_1 + b \cdot u_2 + c \cdot u_3$ and solve for the unknowns a, b, c. A second way (better) is to use the matrix R and **divide** the vector u_4^T by the rows of *R*? How? By subtracting multiples of the rows of R from u_4^T to **reduce** u_4^T as follows.

```
> Transpose(u4);
```

$$\begin{bmatrix}0 & 1 & 1 & 2\end{bmatrix}$$

```
> Transpose(u4) - R[2];
```

$$\begin{bmatrix}0 & 0 & 1 & 1\end{bmatrix}$$

```
> Transpose(u4) - R[2] - R[3];
```

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

The **remainder** is 0 therefore there is a dependency. We are doing a division. Repeated subtraction is division. If we had done this with the polynomials instead of with row vectors then it would look more like a polynomial division. Let's turn to the general division algorithm.

The General Division Algorithm

It may surprise you to learn that this division algorithm was not known before the 20th century. It is simple but it does depend crucially on how we order the terms in our polynomials. Anyone can learn how to do this. When we were dividing vectors, we used an implicit ordering on the unknowns w, x, y, z, namely $w > x > y > z$ because of the order we put on the columns of A. The first step to generalize the division algorithm to polynomials in more than one variable is to decide how to order terms. Suppose the terms of the polynomial are

```
> f := x^2,y^2,4*x*y^2,5*x,2,7*x*y;
```

$$f := x^2, y^2, 4\,x\,y^2, 5\,x, 2, 7\,x\,y$$

One way to order the terms is to first sort on the degree in the first variable, say x. So we have (in descending degree)

$$x^2 > (4\cdot x\cdot y^2, 5\cdot x, 7\cdot x\cdot y) > (y^2, 2)$$

We have one term of degree 2 in x, three of degree 1 and two of degree 0. Now for each set of terms with the same degree in x, sort them in decreasing degree in y to get

$$x^2 > 4\cdot x\cdot y^2 > 7\cdot x\cdot y > 5\cdot x > y^2 > 2$$

This ordering is called pure lexicographical order. In Maple we specify this using $plex(x, y)$. If there was a third variable z we would now sort all terms in $x^i\cdot y^j$ in descending degree on z. The first term, the leading term, is key to the division algorithm. We will denote the leading term of a polynomial f by LT(f). Now when I input the polynomial in this ordering to Maple I get

```
> f := x^2+4*x*y^2+7*x*y+5*x+y^2+2;
```

$$f := 4\,x\,y^2 + x^2 + 7\,x\,y + y^2 + 5\,x + 2$$

Notice that it changed the order of the terms and put the term $4\cdot x\cdot y^2$ first. Why? Because it is using a different term ordering that will also work for the division algorithm and for Groebner bases. Consider again our terms

```
> f := [x^2,y^2,4*x*y^2,5*x,2,7*x*y];
```

$$f := [x^2, y^2, 4\,x\,y^2, 5\,x, 2, 7\,x\,y]$$

This time, first sort the terms by degree (into descending order). So the term $4\cdot x\cdot y^2$ has degree 3 which is more than the term $3\cdot x^2$ which has degree 2. We get

$$4\cdot x\cdot y^2 > (y^2, x^2, 7\cdot x\cdot y) > 5\cdot x > 2$$

There is 1 term of degree 3, three of degree 2, one of degree 1 and one of degree 0. Now sort the terms of each degree by lexicographical order to get

$$4\cdot x\cdot y^2 > x^2 > x\cdot y > y^2 > 5\cdot x > 2.$$

This term ordering is called graded lexicographical order. To get a Groebner basis in this ordering use grlex(x,y). Just to see that this makes a difference let's compare the two orderings

```
> Groebner[Basis]( B1, plex(x,y,z) );
```

$$[z^4 - 2\,z^2 + 2, y^2 + z^2 - 2, -y\,z^3 + 2\,x]$$

```
> Groebner[Basis]( B1, grlex(x,y,z) );
```

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

Now we are ready to do a general division. We will divide the polynomial

$f = x^2 + x\cdot y^2 + z^2 - 1$ by two polynomials $g_1 = x^2 - z^2 + 2$, $g_2 = y^2 - x - z$. The first step is to sort the terms so we can identify the leading terms of the dividend f and divisors g_1 and g_2. We have

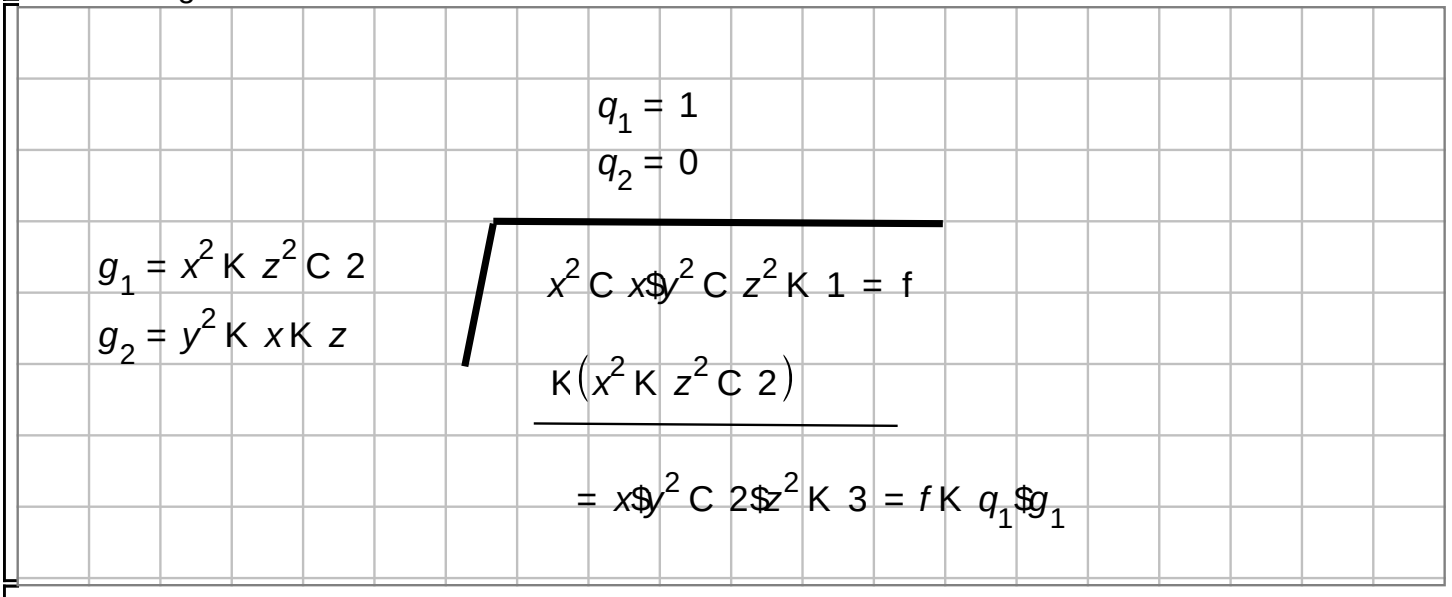
$x^2 + x\cdot y^2 + z^2 - 1$ divided by $\{x^2 - z^2 + 2,\ y^2 - x - z\}$.

The leading terms are x^2 and for the divisors x^2 and y^2. The division algorithm always works with leading terms. It will produce two quotients q_1 and q_2 and one remainder r which will satisfy

$f = q_1\cdot g_1 + q_2\cdot g_2 + r$

and either $r = 0$ or no term in the remainder r is divisable by any leading term of a divisor.

The following figure shows the first step of the division algorithm. I've put the two divisors on the left and the two quotients on the top of the radical symbol and the dividend f is the usual place. The first quotient term is $q_1 = 1$. We multiply and subtract $f - q_1\cdot g_1$ as in the division algorithm for one variable.



Notice that after the subtraction we sort the terms in the result so that we can identify the leading term of the result which is $x\cdot y^2$. Now $LT(g_1) = x^2$ does not divide $x\cdot y^2$ but $LT(g_2) = y^2$ does with quotient x. So we add the next quotient term x to q_2 and multiply and subtract. Here are the next two steps

$g_1 = x^2 - z^2 + 2$
$g_2 = y^2 - x - z$

$q_1 = 1 + 1$
$q_2 = x$

$x^2 + x\cdot y^2 + z^2 - 1 = f$
$-(x^2 - z^2 + 2)$
$= x\cdot y^2 + 2\cdot z^2 - 3 = f - q_1\cdot g_1$
$-x\cdot(y^2 - x - z)$
$= x^2 + x\cdot z + 2\cdot z^2 - 3 = f - 1\cdot g_1 - x\cdot g_2$
$-1\cdot(x^2 - z^2 - 2)$
$= x\cdot z + 3\cdot z^2 - 5 = r$

At this point we have $q_1 = 2$ and $q_2 = x$ and no terms $x\cdot z + 3\cdot z^2 - 5$ are divisible by x^2 or y^2 the leading terms of the divisors so this is the remainder r.

Let us do another division. Consider

$f = x\cdot y^2 + y^2$ divided by $\{ g_1 = x\cdot y + y$ and $g_2 = x\cdot y + x \}$

using lexicographical order with $x > y$ so that the terms are already sorted. Since $LT(g_1) = x\cdot y$ and $LT(g_2) = x\cdot y$ both divide $LT(f) = x\cdot y^2$ so the result is going to depend on which we use first. Let's try it both ways to see what happens.

$g_1 = x\cdot y + y$
$g_2 = x\cdot y + x$

$q_1 = y$
$q_2 = 0$

$x\cdot y^2 + y^2$
$-(x\cdot y^2 + y^2)$
0

$g_1 = x\cdot y + x$
$g_2 = x\cdot y + y$

$q_1 = y - 1$
$q_2 = 0$

$x\cdot y^2 + y^2$
$-(x\cdot y^2 + x\cdot y)$
$-x\cdot y + y^2$
$+ (x\cdot y + x)$
$x + y^2$ = remainder

Note, the terms in the remainder $x+y^2$ are not divisible by either $LT(g_1) = x \cdot y$ or $LT(g_2) = x \cdot y$.
What is surprising is that in one order we get 0 for the remainder but in the other we don't. And that is a serious problem. It means that there's something wrong with the division algorithm. We can't use it as expected to test if $\{g_1, g_2\}$ divides f with 0 remainder. It turns out that the problem is not the division algorithm, rather, it's the basis $\{g_1, g_2\}$ that we are using. It's not a "good basis". That is, it's "not a Groebner basis". If we'd used a Groebner basis we would get a 0 remainder. Incidentally, the same problem occurs if you try to divide a vector u by a set of vectors $\{v_1, v_2, ...\}$ if you don't first put the vectors in Row Echelon form. Let us first simplify the divisor basis. We have

```
> g1 := x*y+y;
```

$$g1 := x\,y + y$$

```
> g2 := x*y+x;
```

$$g2 := x\,y + x$$

```
> g3 := g2-g1;
```

$$g3 := x - y$$

I'd better use g3 to simplify g1 and g2

```
> g1 := g1 - expand(y*g3); # simplify g1
```

$$g1 := y^2 + y$$

```
> g2 := g2 - expand(y*g3); # simplify g2
```

$$g2 := y^2 + x$$

```
> g2 := g2-g3; # simnplify g2 some more
```

$$g2 := y^2 + y$$

```
> G := {g1,g2,g3};
```

$$G := \{x - y,\ y^2 + y\}$$

This is actually a Groebner basis and we have the following property.

Property of a Groebner Basis. A Groebner basis $G = \{g_1, g_2, ..., g_t\}$ for a set of input polynomials $B = \{f_1, f_2, ..., f_t\}$ with respect to a given term ordering, has the property that if a given polynomial f may be written as $f = h_1 \cdot f_1 + h_2 \cdot f_2 + ... + h_s \cdot f_s$ for some polynomials h_i, then the remainder of f divided by G is 0. This allows us to answer question 3.

```
> sys1;
```

$$\{x\,y = z,\ x^2 + y^2 = 1,\ -x^2 + z^2 = 1,\ y^2 + z^2 = 2\}$$

```
> f1,f2,f3,f4 := x*y-z,x^2+y^2-1,z^2-x^2-1,y^2+z^2-2;
```

$$f1, f2, f3, f4 := x\,y - z,\ x^2 + y^2 - 1,\ -x^2 + z^2 - 1,\ y^2 + z^2 - 2$$

To test if the 4th equation is redundant, we first compute a Groebner basis G for $[f_1, f_2, f_3]$ using any term ordering.

```
> G := Groebner[Basis]( [f1,f2,f3], plex(x,y,z) );
```

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

Now we just need to divide f_4 by G and see if we get 0 remainder. This command does the division and returns the remainder

```
> Groebner[NormalForm]( f4, G, plex(x,y,z) );
```

$$0$$

Since it is 0 then f_4 is redundant.

Note, division of f_4 by $\{f_1, f_2, f_3\}$ does not yield a 0 remainder

```
> Groebner[NormalForm]( f4, [f1,f2,f3], plex(x,y,z) );
```

$$y^2 + z^2 - 2$$

Limitations

There is much more to Groebner bases of course. More properties, more applications. But there is a catch that needs to be stated lest you think that you can now use Groebner bases to solve systems of hundreds of equations. If we have a system of n quadratic equations in n unknowns then the number of solutions is 2^n in general. Trying to solve the system using the lexicographical monomial ordering, in general will result in a triangular system with one polynomial of degree 2^n which even for small n is too big to compute. Here are four quadratics so you get a sense of this.

```
> f1,f2,f3,f4 := w^2+x^2+y^2+z^2-1, 2*w^2-y^2+z*y+x*z-2,
                 w^2+3*y^2-x*z-x^2-1,w*x+w*y+w*z-1;
```

$$f1, f2, f3, f4 := w^2 + x^2 + y^2 + z^2 - 1,\ 2\,w^2 + x\,z - y^2 + y\,z - 2,\ w^2 - x^2 - x\,z + 3\,y^2 - 1,\ w\,x + w\,y + w\,z - 1$$

```
> G := Groebner[Basis]([f1,f2,f3,f4],plex(w,x,y,z)):
  G[1];
```

$$5043\,z^{16} + 129437\,z^{14} + 3740002\,z^{12} - 18940119\,z^{10} + 49028829\,z^8 - 62996200\,z^6 + 45849200\,z^4 - 5360000\,z^2 + 4000000$$

If $n = 100$ there is no hope of ever computing the Groebner basis. We cannot store let alone compute a polynomial of degree 2^{100}. But a linear system of 100 equations in 100 unknowns is easy.

Term Orderings

Lastly, I want to mention that these two term orderings that we have been using satisfy three properties which together are what will make the division algorithm work. The three properties are

(i) The term ordering is a total ordering, that is, you can sort the terms $t_1, t_2, t_3, \ldots$ into $t_1 > t_2 > t_3 > \ldots$ which means that $t_1 > t_3$ by transitivity.

(i) For any three non-zero terms t_1, t_2, t_3 we have $t_1 > t_2 \Rightarrow t_3 \cdot t_1 > t_3 \cdot t_2$.

(iii) Every non-empty set of terms $\{t_1, t_2, t_3, \ldots\}$ has a least element in the term ordering.

The last property is needed for termination of the division algorithm. It is called the **well-ordering** property. It is not at all obvious that the two orderings we have defined satisfy property (iii). That needs a formal proof. We will not give it here.

For one variable, there is only one term ordering that satisfies all three properties, namely,

$$1 < x < x^2 < x^3 < \ldots$$

For two more more variables there are an infinite number of term orderings. When computing Groebner bases, some term orderings are easier than others and some properties of a Groebner basis may depend on specific term orderings.

Some helpful references are the Wiki page which explains Groebner bases in terms of ideals which I've tried to avoid here. And the CLO textbook is the authoritative treatment.

References

Ideals Varieties and Algorithms, Cox, Little, and O'Shea, Springer Verlag Undergraduate Texts in Mathematics.