SIMU 1999–Gröbner Bases Seminar
Week 2 Computer Laboratory Exercises

Background and Goals

This week, we will begin to work with some of the commands in the `Groebner` package of Maple, which will be one of our main tools for the remainder of the Institute. This package is new in Release 5 of Maple. It represents a big advance (both in flexibility and in performance) over the previous Gröbner basis package (called `grobner`)! The old package is still distributed with Maple, so do not get confused if you see both names. We want the `Groebner` package.

Days 1,2: New Maple Data Structures, Defining Monomial Orders

Before we start in with the `Groebner` package, we need to take a bit of time and look at some new ways of *structuring information* in Maple.

Lists or Vectors

To represent points in affine space, we use coordinate vectors. The same idea is also useful to represent any *ordered collection* of information. In Maple, these ordered collections are called *lists*. A list is indicated by a pair of *square brackets* ([,]) enclosing the items in the list, separated by commas. For instance, the input line

```
list1 := [a,b,c,d,e];
```

creates a list with five items (the letters a, b, c, d, e, treated as symbolic variables) and assigns the list to the name `list1`. We have said that lists are *ordered*. What this means, for instance, is that `alist` above is *different from* the list defined by `list2 := [a,d,c,e,b];`. You can test this by entering the definition of `list2`, then the command

```
evalb(list1 = list2);
```

(The = is the comparison operator – not assignment; the `evalb` function returns a true or false ("b̲oolean") value indicating whether the two lists are equal.)

A list can have any number of items, including *no items* at all – the empty list is written as `[]`. The same item can also occur at several places in a list (unlike the case for a *set*, which is an unordered collection of information with no repetitions. Sets can also be defined and manipulated in Maple. The notation for a set is {`item1,item2, ... itemn`}).

The items in a list can be any legal Maple objects – constants, variables, polynomials, expressions, other lists, ... For example, in many of our calculations, we will be dealing with lists of polynomials in a ring $k[x_1, \ldots, x_n]$. We will think of these as particular lists of generators for ideals $I \subset k[x_1, \ldots, x_n]$. For example,

```
Id := [x^3-y^2+2*y,x*y-y-4*z,z^4-1];
```

would be one way to represent the ideal

$$Id = \langle x^3 - y^2 + 2y, xy - y - 4z, z^4 - 1\rangle,$$

(recall that the ideal contains all the polynomials

$$g_1 \cdot (x^3 - y^2 + 2y) + g_2 \cdot (xy - y - 4z) + g_3 \cdot (z^4 - 1)$$

for $g_1, g_2, g_3 \in \mathbf{Q}[x, y, z]$. The list notation will parallel our notation from class for the ideal generated by a given set: $I = \langle f_1, \ldots, f_s\rangle$.

The items in a list *do not* all have to be items of the same *type*. For instance, another perfectly OK Maple list is the following:

```
list3 := [[a,b,c],d,e];
```

where there are three items in all. (The first list item in `list3` is the list `[a,b,c]`, and the second and third items are the symbolic variables d, e.)

Some useful list operations

- The built-in function `op` returns the items in a list, separated by commas, in the order they appear in the list.

This is particularly useful for manipulations of lists such as adding new entries. For instance, enter the command

```
list1 := [op(list1),f];
```

What this does is as follows:

1) first, the `op` function returns the items from the *"old"* `list1`.
2) second, the new item f is appended to the end after a new comma
3) third (since the right hand side is in a new set of square brackets), a new list is created containing the old entries from `list1` and the new one
4) finally, the new list is assigned back to the name `list1`. (When this is done the old value of `list1` is "erased".)

Additional commands:

- The built-in `nops` function returns the number of items in a list. (If one of the items is a list itself, it is counted as a single item.)
- To pick out the ith item in a named list (like `list1`), you can use either the notation `list1[i]`, or `op(i,list1)`.
- You can also pick out a consecutive sublist – the items in the slots numbered `first` to `last` – from a list by using the format `list1[first..last]`. This gives a list as output. The alternate form `op(first..last,list1)` gives the items in that part of the list.

Exercise 1

Explain, and give examples showing:

a) how to append a new item at the *start* of a list
b) how to create a list containing the entries of a given list, followed by a second copy of that same list
c) how to select the *last* item in a list (without counting the number of items yourself!)
d) how to insert a new item as the third entry from the start in a list (assuming it has at least two entries to begin with)? Be careful to test your idea on lists with two and with more than two entries!

Matrices

If we think of a list as a row vector, then it is natural to think of a list of lists (all of the same size) as a way to represent a *matrix*. We can do this in Maple as well, as a basic way to deal with square or more general rectangular matrices. For example,

```
mat:=[[x,y],[u,v]];
```

creates a list of two row vectors, or a 2×2 matrix. To select the entry in row i and column j from a matrix defined this way, we could use, for instance `mat[i][j]` (`mat[i]` is the ith row, so `mat[i][j]` is the jth entry in that row).

First Steps with the Groebner Package

All the commands in the `Groebner` *package are accessible if (and only if) you have loaded the package in your current Maple session, using* `with`.

Monomial (or Term) Orders

Although it is not strictly necessary for many things, it makes sense to get in the habit of starting a session with the `Groebner` package by defining the polynomial ring you want to work in. This is done by first loading another package called `Ore_algebra`, then defining the polynomial ring using the `poly_algebra` command. (The `Ore_algebra` package gives a way to work in a fairly extensive class of possibly noncommutative rings, which are defined by specifying commutation laws between the generators of the ring over the field of constants. For polynomial rings, the commutation law is as simple as possible: $xy - yx = 0$ for all x, y. A version of Gröbner basis theory is valid in all of these rings.) For example to define the ring $\mathbf{Q}[x, y, z]$ you would use:

```
with(Ore_algebra);
ring := poly_algebra(x,y,z);
```

As we have seen in class, the first step toward Gröbner bases is defining various orders on the monomials in several variables. The Groebner package has a completely general way to do this, and also several simplified predefined special cases for you to use. For example:

- the *lexicographic* order in $k[x_1, \ldots, x_n]$ with

$$x_1 > x_2 > \cdots > x_n$$

  is represented by `plex(x1, ...   , xn)`. (Note: The ordering of the variables is determined by the order in which you list them inside the `plex()`. Also, you cannot use the dots in actual commands; you need to list out all the variables explicitly.)
- the *graded reverse lex order* in $k[x_1, \ldots, x_n]$ with

$$x_1 > x_2 > \cdots > x_n$$

  is represented by `tdeg(x1, ...   , xn)`.
- From discussion, we know that one can also define monomial orders in $k[x_1, \ldots, x_n]$ called *weight orders*, $>_w$. We compare two monomials x^α and x^β first by taking dot products with a fixed *weight vector* w, then "breaking ties" using another order $>_\sigma$. That is

$$x^\alpha >_w x^\beta \Leftrightarrow w \cdot \alpha > w \cdot \beta \text{ or } [(w \cdot \alpha = w \cdot \beta) \text{ and } (x^\alpha >_\sigma x^\beta)]$$

  This type of order is represented in Maple by

```
wdeg([w1, ..., wn],[x1, ...  ,xn])
```

The $>_\sigma$ order here is the (non-graded) *reverse lex* order: $\alpha >_\sigma \beta$ if in $\alpha - \beta$, the rightmost non-zero element is negative. (This is not a monomial order itself, but it can be used to break ties here; do you see why?) Note: the weights and variables are vectors (lists) in this form. The *weight vector always comes first*, and its entries must be integers.

To actually define termorders on the polynomial ring `ring` defined above, you can enter the following commands (`termorder` is one of the `Groebner package commands`)

```
MO1 := termorder(ring,plex(x,y,z));
MO2 := termorder(ring,plex(y,z,x));
MO3 := termorder(ring,tdeg(x,y,z));
MO4 := termorder(ring,wdeg([2,4,8],[x,y,z]));
```

Then `MO1` and `MO2` are two different *lex* orders, `MO2` is the *grevlex* order with $x > y > z$, and `MO4` is the weight order $>_w$ for $w = [2, 4, 8]$.

The Leading Monomial of a Polynomial

The `leadterm` function in the `Groebner` package selects the leading monomial in a polynomial according to a specified order. Its format is `leadterm(poly,order);`. There is a very similar command called `leadmon` as well. The output of `leadmon` will be a pair of values, first the constant leading coefficient of the polynomial, then after a comma, the actual leading monomial. (Note that the author of the `Groebner` package uses the opposite terminology for "terms" and "monomials" from what we did in IVA! For them, a "term" always has coefficient 1 and a monomial can have any coefficient $c \in k$.)

Exercise 2

Using each of the four monomial orders defined above, find the leading monomial of

$$p(x, y, z) = x^3z - 5xyz^3 + 7y^5 - 3x^2y^4$$

and verify your results by hand.

Defining a Monomial Order by a Matrix

In discussion today, we saw that we can generalize the weight orders $>_w$, to define monomial orders on $k[x_1, \ldots, x_n]$ starting from any $m \times n$ matrix M with

- $m \geq n$,
- $\text{rank}(M) = n$,
- all entries non-negative integers.

Namely, suppose the rows of M are the vectors $w_1, \ldots, w_n$. Then we can compare monomials x^α and x^β by first comparing their w_1-weights, then breaking ties sucessively with the w_2-weights, w_3-weights, and so on until the w_m-weights. In symbols:

$$\begin{aligned}
x^\alpha >_M x^\beta \Leftrightarrow\ & w_1 \cdot \alpha > w_1 \cdot \beta \\
& \text{or } [(w_1 \cdot \alpha = w_1 \cdot \beta) \text{ and } (w_2 \cdot \alpha > w_2 \cdot \beta)] \\
& \text{or } [(w_1 \cdot \alpha = w_1 \cdot \beta) \text{ and } (w_2 \cdot \alpha = w_2 \cdot \beta) \text{ and } (w_3 \cdot \alpha > w_3 \cdot \beta)] \\
& \text{or } \cdots \\
& \text{or } [(w_1 \cdot \alpha = w_1 \cdot \beta) \text{ and } \cdots \text{ and } (w_{m-1} \cdot \alpha = w_{m-1} \cdot \beta) \text{ and } (w_m \cdot \alpha > w_m \cdot \beta)]
\end{aligned}$$

All monomial orders can be specified as $>_M$ orders for appropriate matrices M.

To define a matrix order in the Maple `Groebner` package, we enter a command like this

```
MO := termorder(ring,'matrix'(M,varlist));
```

(single quotes around the word *matrix*). The matrix M itself is entered as a list of lists as defined above, either first in a command assigning the matrix to a name, or else directly in the `termorder` command. The `varlist` is the list of variables. The ordering you give here will correspond to the ordering of the components in the exponent vectors. For example, if for some reason we did not want to use the builtin *lex* order, we could define something equivalent on $\mathbf{Q}[x, y, z]$ using

```
MO := termorder(ring,'matrix'([[1,0,0],[0,1,0],[0,0,1]],[x,y,z]));
```

Exercise 3

What matrix would define the weight order `M04` on $\mathbf{Q}[x, y, z]$ defined above? Create an appropriate matrix term order in Maple, and test it by finding the leading term of $f = xy^4z + 4z^4$, and a polynomial of your choice. (You will need to think about how to get a square 3×3 matrix to represent computing $[2, 4, 8]$-weights first, then breaking ties with the non-graded reverse lex order).

Additional Examples of Monomial Orders

Exercises 10 and 12 from Chapter 2, §5 of "IVA" contain two additional important examples of monomial orders – the *product* orders, and Bayer and Stillman's *elimination* orders.

Exercise 4

Let $R = \mathbf{Q}[x_1, x_2, x_3, y_1, y_2, y_3, y_4]$.

a) Using the matrix order setup from the `Groebner` package, determine how to define a mixed order $>_{mixed}$ on R where

$$x^\alpha y^\beta >_{mixed} x^\gamma y^\delta \Leftrightarrow x^\alpha >_{lex} x^\gamma \text{ or } ((x^\alpha = x^\gamma) \text{ and } (y^\beta >_{grevlex} y^\delta)).$$

(Hint: Think of making the matrix from "blocks" like the ones above.)

b) Using the `Groebner` package, determine how to define the elimination order $>_3$ from Exercise 12.

Days 3,4: The Division Algorithm in $k[x_1, \ldots, x_n]$

In these labs we will want to use Maple to work out some examples with the division algorithm for multivariable polynomials. The `Groebner` package contains a command called `normalf` which computes the remainder on division of a given polynomial f by a list of polynomials $f_1, \ldots, f_s$. Unfortunately, the quotients a_i in the expression

$$f = a_1 f_1 + \cdots + a_s f_s + r$$

are not saved or displayed.

To see the quotients on division, we will need to use a "home-grown" version of division called `DivQuots` that follows the algorithm in Chapter 2, §3 of "IVA" exactly. (NOTE: This uses several functions from the `Groebner` package, so you should *also* load that if you want to do division with quotients.) We have mentioned before that Maple actually contains a fully-developed built-in *programming language*, including comparisons, loops, etc. This means that (if necessary) it is possible to write programs, or *procedures* in Maple to make new commands not part of the basic system available to users. The `DivQuots` procedure we will use is such a case. We will actually look at the Maple code next week. For now, let's just use this new command.

In order to use it, you will need to read in the Maple code for the procedure from the class network drive. You do this in a Maple worksheet by entering the command

```
read `f:/users/simu/Little/dq.map`;
```

So that you can have a copy of this to use whenever you want, you can also "drag and drop" the icon for the file dq from this directory to your own network directory. You will probably also want to keep this on one of your diskettes if you want access to it at Palmas. The format of `DivQuots` is

```
DivQuots(f,Plist,MO);
```

where f is the dividend polynomial, $Plist$ is the list of divisors (a list of polynomials), and MO is the monomial order you want use to pick out leading terms in the division process, defined according to the rules discussed above for monomial orders in the `Groebner` package. $Plist$ is a Maple *list* – the ordering is important!

The output of DivQuots consists of two parts. First, comes the list of quotients. Following that comes the remainder. If you assign the output of DivQuots to a name, for example:

```
dq:=DivQuots(x^5+x^3*y*z+1,[3*x^2+x*y+1,x*z-y+z],plex(x,y,z));
```

then `dq[1]` is the list of quotients (so `dq[1][2]`) is the quotient corresponding to the second divisor), and `dq[2]` is the remainder. The quotients and the remainder will, in general, depend on the monomial order you are using. For example, compare the output of the command above with the output of

```
DivQuots(x^5+x^3*y*z+1,[3*x^2+x*y+1,x*z-y+z],wdeg([1,8,1],[x,y,z]));
```

Exercise 5

Check that both sets of output are correct. (How? Recall the quotients $a_1, \ldots, a_s$ and the remainder r are supposed to satisfy $f = a_1 f_1 + \cdots + a_s f_s + r$, the leading term of each of the $a_i f_i$ is supposed to be less than or equal to the leading term of f in the given order, and no term in r is supposed to be divisible by any leading term of one of the f_i.)

Exercise 6

We did problems 1 and 2 from Chapter 2, §3 of "IVA" in discussion. Check your work using Maple.

Exercise 7

Do problems 5-8 in Chapter 2, §3 of "IVA".

Note: The built-in `normalf` command in the `Groebner` package only computes a remainder on division (no quotients). Its format is

```
normalf(f,Plist,MO);
```

where f, $Plist$, and MO are as above for the `DivQuots` procedure.