```
if finite(PList,VList) then
  RList := PList;
  for v in VList do
    p := finduni(v,PList,VList);
    pred := simplify(p/gcd(p,diff(p,v)));
    RList:=[op(RList),pred];
  od;
  RETURN(RList)
else
  ERROR('Ideal not zero-dimensional; method does not apply')
fi
end:
```

§3 Gröbner Basis Conversion

In this section, we will use linear algebra in $A = k[x_1, \ldots, x_n]/I$ to show that a Gröbner basis G for a zero-dimensional ideal I with respect to one monomial order can be converted to a Gröbner basis G' for the same ideal with respect to *any* other monomial order. The process is sometimes called *Gröbner basis conversion*, and the idea comes from a paper of Faugère, Gianni, Lazard, and Mora [FGLM]. We will illustrate the method by converting from an arbitrary Gröbner basis G to a *lex* Gröbner basis G_{lex} (using any ordering on the variables). The Gröbner basis conversion method is often used in precisely this situation, so that a more favorable monomial order (such as *grevlex*) can be used in the application of Buchberger's algorithm, and the result can then be converted into a form more suited for equation solving via elimination. For another discussion of this topic, see [BW], §1 of Chapter 9.

The basic idea of the Faugère-Gianni-Lazard-Mora algorithm is quite simple. We start with a Gröbner basis G for a zero-dimensional ideal I, and we want to convert G to a *lex* Gröbner basis G_{lex} for some *lex* order. The algorithm steps through monomials in $k[x_1, \ldots, x_n]$ in increasing *lex* order. At each step of the algorithm, we have a list $G_{lex} = \{g_1, \ldots, g_k\}$ of elements in I (initially empty, and at each stage a subset of the eventual *lex* Gröbner basis), and a list B_{lex} of monomials (also initially empty, and at each stage a subset of the eventual *lex* monomial basis for A). For each input monomial x^α (initially 1), the algorithm consists of three steps:

(3.1) Main Loop. Given the input x^α, compute $\overline{x^\alpha}^G$. Then:

a. If $\overline{x^\alpha}^G$ is *linearly dependent* on the remainders (on division by G) of the monomials in B_{lex}, then we have a linear combination

$$\overline{x^\alpha}^G - \sum_j c_j \overline{x^{\alpha(j)}}^G = 0,$$

where $x^{\alpha(j)} \in B_{lex}$ and $c_j \in k$. This implies that

$$g = x^\alpha - \sum_j c_j x^{\alpha(j)} \in I.$$

We add g to the list G_{lex} as the last element. Because the x^α are considered in increasing *lex* order (see (3.3) below), whenever a polynomial g is added to G_{lex}, its leading term is $\text{LT}(g) = x^\alpha$ with coefficient 1.

b. If $\overline{x^\alpha}^G$ is *linearly independent* from the remainders (on division by G) of the monomials in B_{lex}, then we add x^α to B_{lex} as the last element.

After the Main Loop acts on the monomial x^α, we test G_{lex} to see if we have the desired Gröbner basis. This test needs to be done only if we added a polynomial g to G_{lex} in part a of the Main Loop.

(3.2) Termination Test. If the Main Loop added a polynomial g to G_{lex}, then compute $\text{LT}(g)$. If $\text{LT}(g)$ is a power of x_1, where x_1 is the greatest variable in our *lex* order, then the algorithm terminates.

The proof of Theorem (3.4) below will explain why this is the correct way to terminate the algorithm. If the algorithm does not stop at this stage, we use the following procedure to find the next input monomial for the Main Loop:

(3.3) Next Monomial. Replace x^α with the next monomial in *lex* order which is not divisible by any of the monomials $\text{LT}(g_i)$ for $g_i \in G_{lex}$.

Exercise 3 below will explain how the Next Monomial procedure works. Now repeat the above process by using the new x^α as input to the Main Loop, and continue until the Termination Test tells us to stop.

Before we prove the correctness of this algorithm, let's see how it works in an example.

Exercise 1. Consider the ideal

$$I = \langle xy + z - xz, x^2 - z, 2x^3 - x^2yz - 1 \rangle$$

in $\mathbb{Q}[x, y, z]$. For *grevlex* order with $x > y > z$, I has a Gröbner basis $G = \{f_1, f_2, f_3, f_4\}$, where

$$\begin{aligned} f_1 &= z^4 - 3z^3 - 4yz + 2z^2 - y + 2z - 2 \\ f_2 &= yz^2 + 2yz - 2z^2 + 1 \\ f_3 &= y^2 - 2yz + z^2 - z \\ f_4 &= x + y - z. \end{aligned}$$

Thus $\langle \mathrm{LT}(I)\rangle = \langle z^4, yz^2, y^2, x\rangle$, $B = \{1, y, z, z^2, z^3, yz\}$, and a remainder $\overline{f}^G$ is a linear combination of elements of B. We will use basis conversion to find a *lex* Gröbner basis for I, with $z > y > x$.

a. Carry out the Main Loop for $x^\alpha = 1, x, x^2, x^3, x^4, x^5, x^6$. At the end of doing this, you should have

$$G_{lex} = \{x^6 - x^5 - 2x^3 + 1\}$$
$$B_{lex} = \{1, x, x^2, x^3, x^4, x^5\}.$$

Hint: The following computations will be useful:

$$\overline{1}^G = 1$$
$$\overline{x}^G = -y + z$$
$$\overline{x^2}^G = z$$
$$\overline{x^3}^G = -yz + z^2$$
$$\overline{x^4}^G = z^2$$
$$\overline{x^5}^G = z^3 + 2yz - 2z^2 + 1$$
$$\overline{x^6}^G = z^3.$$

Note that $\overline{1}^G, \ldots, \overline{x^5}^G$ are linearly independent while $\overline{x^6}^G$ is a linear combination of $\overline{x^5}^G$, $\overline{x^3}^G$ and $\overline{1}^G$. This is similar to Exercise 2 of §2.

b. After we apply the Main Loop to x^6, show that the monomial provided by the Next Monomial procedure is y, and after y passes through the Main Loop, show that

$$G_{lex} = \{x^6 - x^5 - 2x^3 + 1, y - x^2 + x\}$$
$$B_{lex} = \{1, x, x^2, x^3, x^4, x^5\}.$$

c. Show that after y, Next Monomial produces z, and after z passes through the Main Loop, show that

$$G_{lex} = \{x^6 - x^5 - 2x^3 + 1, y - x^2 + x, z - x^2\}$$
$$B_{lex} = \{1, x, x^2, x^3, x^4, x^5\}.$$

d. Check that the Termination Test (3.2) terminates the algorithm when G_{lex} is as in part c. Hint: We're using *lex* order with $z > y > x$.

e. Verify that G_{lex} from part c is a *lex* Gröbner basis for I.

We will now show that the algorithm given by (3.1), (3.2) and (3.3) terminates and correctly computes a *lex* Gröbner basis for the ideal I.

(3.4) Theorem. *The algorithm described above terminates on every input Gröbner basis G generating a zero-dimensional ideal I, and correctly computes a lex Gröbner basis G_{lex} for I and the lex monomial basis B_{lex} for the quotient ring A.*

PROOF. We begin with the key observation that monomials are added to the list B_{lex} in strictly increasing *lex* order. Similarly, if $G_{lex} = \{g_1, \ldots, g_k\}$, then

$$\mathrm{LT}(g_1) <_{lex} \cdots <_{lex} \mathrm{LT}(g_k),$$

where $>_{lex}$ is the *lex* order we are using. We also note that when the Main Loop adds a new polynomial g_{k+1} to $G_{lex} = \{g_1, \ldots, g_k\}$, the leading term $\mathrm{LT}(g_{k+1})$ is the input monomial in the Main Loop. Since the input monomials are provided by the Next Monomial procedure, it follows that for all k,

(3.5) $\mathrm{LT}(g_{k+1})$ is divisible by none of $\mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_k)$.

We can now prove that the algorithm terminates for all inputs G generating zero-dimensional ideals. If the algorithm did not terminate for some input G, then the Main Loop would be executed infinitely many times, so one of the two alternatives in (3.1) would be chosen infinitely often. If the first alternative were chosen infinitely often, G_{lex} would give an infinite list $\mathrm{LT}(g_1), \mathrm{LT}(g_2), \ldots$ of monomials. However, we have:

- (Dickson's Lemma) Given an infinite list $x^{\alpha(1)}, x^{\alpha(2)}, \ldots$ of monomials in $k[x_1, \ldots, x_n]$, there is an integer N such that every $x^{\alpha(i)}$ is divisible by one of $x^{\alpha(1)}, \ldots, x^{\alpha(N)}$.

(See, for example, Exercise 7 of [CLO], Chapter 2, §4). When applied to $\mathrm{LT}(g_1), \mathrm{LT}(g_2), \ldots$, Dickson's Lemma would contradict (3.5). On the other hand, if the second alternative were chosen infinitely often, then B_{lex} would give infinitely many monomials $x^{\alpha(j)}$ whose remainders on division by G were linearly independent in A. This would contradict the assumption that I is zero-dimensional. As a result, the algorithm always terminates for G generating a zero-dimensional ideal I.

Next, suppose that the algorithm terminates with $G_{lex} = \{g_1, \ldots, g_k\}$. By the Termination Test (3.2), $\mathrm{LT}(g_k) = x_1^{a_1}$, where $x_1 >_{lex} \cdots >_{lex} x_n$. We will prove that G_{lex} is a *lex* Gröbner basis for I by contradiction. Suppose there were some $g \in I$ such that $\mathrm{LT}(g)$ is not a multiple of any of the $\mathrm{LT}(g_i)$, $i = 1, \ldots, k$. Without loss of generality, we may assume that g is *reduced* with respect to G_{lex} (replace g by $\overline{g}^{G_{lex}}$).

If $\mathrm{LT}(g)$ is greater than $\mathrm{LT}(g_k) = x_1^{a_1}$, then one easily sees that $\mathrm{LT}(g)$ is a multiple of $\mathrm{LT}(g_k)$ (see Exercise 2 below). Hence this case can't occur, which means that

$$\mathrm{LT}(g_i) < \mathrm{LT}(g) \leq \mathrm{LT}(g_{i+1})$$

for some $i < k$. But recall that the algorithm places monomials into B_{lex} in strictly increasing order, and the same is true for the $\mathrm{LT}(g_i)$. All the

non-leading monomials in g must be less than $\text{LT}(g)$ in the *lex* order. They are not divisible by any of $\text{LT}(g_j)$ for $j \le i$, since g is reduced. So, the non-leading monomials that appear in g would have been included in B_{lex} by the time $\text{LT}(g)$ was reached by the Next Monomial procedure, and g would have been the next polynomial after g_i included in G_{lex} by the algorithm (i.e., g would equal g_{i+1}). This contradicts our assumption on g, which proves that G_{lex} is a *lex* Gröbner basis for I.

The final step in the proof is to show that when the algorithm terminates, B_{lex} consists of *all* basis monomials determined by the Gröbner basis G_{lex}. We leave this as an exercise for the reader. □

Additional Exercises for §3

Exercise 2. Consider the *lex* order with $x_1 > \cdots > x_n$ and fix a power x_1^a of x_1. Then, for any monomial x^α in $k[x_1, \ldots, x_n]$, prove that $x^\alpha > x_1^a$ if and only if x^α is divisible by x_1^a.

Exercise 3. Suppose $G_{lex} = \{g_1, \ldots, g_k\}$, where $\text{LT}(g_1) < \cdots < \text{LT}(g_k)$, and let x^α be a monomial. This exercise will show how the Next Monomial (3.3) procedure works, assuming that our *lex* order satisfies $x_1 > \cdots > x_n$. Since this procedure is only used when the Termination Test fails, we can assume that $\text{LT}(g_k)$ is *not* a power of x_1.

a. Use Exercise 2 to show that none of the $\text{LT}(g_i)$ divide $x_1^{a_1+1}$.

b. Now consider the *largest* $1 \le k \le n$ such that none of the $\text{LT}(g_i)$ divide the monomial

$$x_1^{a_1} \cdots x_{k-1}^{a_{k-1}} x_k^{a_k+1}.$$

By part a, $k = 1$ has this property, so there must be a largest such k. If x^β is the monomial corresponding to the largest k, prove that $x^\beta > x^\alpha$ is the smallest monomial (relative to our *lex* order) greater than x^α which is not divisible by any of the $\text{LT}(g_i)$.

Exercise 4. Complete the proof of Theorem (3.4) by showing that when the basis conversion algorithm terminates, the set B_{lex} gives a monomial basis for the quotient ring A.

Exercise 5. Use Gröbner basis conversion to find *lex* Gröbner bases for the ideals in Exercises 6 and 7 from §1. Compare with your previous results.

Exercise 6. What happens if you try to apply the basis conversion algorithm to an ideal that is *not* zero-dimensional? Can this method be used for general Gröbner basis conversion? What if you have more information about the *lex* basis elements, such as their total degrees, or bounds on those degrees?

Exercise 7. Show that the output of the basis conversion algorithm is actually a monic *reduced lex* Gröbner basis for $I = \langle G \rangle$.

Exercise 8. Implement the basis conversion algorithm outlined in (3.1), (3.2) and (3.3) in a computer algebra system. Hint: Exercise 3 will be useful. For a more complete description of the algorithm, see pages 428–433 of [BW].

§4 Solving Equations via Eigenvalues

The central problem of this chapter, finding the solutions of a system of polynomial equations $f_1 = f_2 = \cdots = f_s = 0$ over $\mathbb{C}$, rephrases in fancier language to finding the points of the variety $\mathbf{V}(I)$, where I is the ideal generated by $f_1, \ldots, f_s$. When the system has only finitely many solutions, i.e., when $\mathbf{V}(I)$ is a finite set, the Finiteness Theorem from §2 says that I is a zero-dimensional ideal and the algebra $A = \mathbb{C}[x_1, \ldots, x_n]/I$ is a finite-dimensional vector space over $\mathbb{C}$. The first half of this section exploits the structure of A in this case to evaluate an arbitrary polynomial f at the points of $\mathbf{V}(I)$; in particular evaluating the polynomials $f = x_i$ gives the coordinates of the points (Corollary (4.6) below). The values of f on $\mathbf{V}(I)$ turn out to be *eigenvalues* of certain linear mappings on A, and the remainder of the section discusses techniques for evaluating them.

We begin with the easy observation that given a polynomial $f \in \mathbb{C}[x_1, \ldots, x_n]$, we can use multiplication to define a linear map m_f from $A = \mathbb{C}[x_1, \ldots, x_n]/I$ to itself. More precisely, f gives the coset $[f] \in A$, and we define $m_f : A \to A$ by the rule: if $[g] \in A$, then

$$m_f([g]) = [f] \cdot [g] = [fg] \in A.$$

Then m_f has the following basic properties.

(4.1) Proposition. *Let $f \in \mathbb{C}[x_1, \ldots, x_n]$. Then*

a. *The map m_f is a linear mapping from A to A.*

b. *We have $m_f = m_g$ exactly when $f - g \in I$. Thus two polynomials give the same linear map if and only if they differ by an element of I. In particular, m_f is the zero map exactly when $f \in I$.*

Proof. The proof of part a is just the distributive law for multiplication over addition in the ring A. If $[g], [h] \in A$ and $c \in k$, then

$$m_f(c[g] + [h]) = [f] \cdot (c[g] + [h]) = c[f] \cdot [g] + [f] \cdot [h] = cm_f([g]) + m_f([h]).$$

Part b is equally easy. Since $[1] \in A$ is a multiplicative identity, if $m_f = m_g$, then

$$[f] = [f] \cdot [1] = m_f([1]) = m_g([1]) = [g] \cdot [1] = [g],$$