5-2010

A Criterion for Identifying Stressors In Non-Linear Equations Using Gröbner Bases

Elisabeth Marie Palchak

The University of Southern Mississippi

A CRITERION FOR IDENTIFYING STRESSORS IN NON-LINEAR EQUATIONS USING GRÖBNER BASES

by

Elisabeth Marie Palchak

A Thesis
Submitted to the Honors College of
The University of Southern Mississippi
in Partial Fulfillment
of the Requirements for the Degree of
Bachelor of Science
in the Department of Mathematics

May 2010

Approved by

John Perry
Assistant Professor of Mathematics

C. S. Chen
Chair, Department of Mathematics

David R. Davies, Dean
Honors College

Contents

CHAPTER 1

Introduction

We want to know what values of the coefficients $a, b_1, b_2, c, d_1, d_2, d_3$ guarantee finitely many real solutions for x_1, x_2, and x_3 in the system

$$\begin{aligned} ax_1x_2 - b_1x_1 - cx_2 + d_1 &= 0 \\ ax_1x_3 - b_2x_1 - cx_3 + d_2 &= 0 \\ ax_2x_3 - b_2x_2 - b_1x_3 + d_3 &= 0, \end{aligned} \tag{1}$$

where $a, b_1, b_2, c, d_1, d_2, d_3$ are constants and x_1, x_2, x_3 are unknowns. This system derives from a linear algebra technique proposed by Alexander Pozhitkov, a chemist at the Max Planck Institute for Evolutionary Biology [9]. The goal of Pozhitkov's research is to take a sample fish, examine the fish's gene expression, and detect what contaminant is affecting the fish. Pozhitkov uses a tool called a microarray to generate a vector of positive integers, which we can analyze mathematically.

Genes cannot be analyzed by simply comparing the microarray data because there is not enough difference between the vectors to recognize the stressor. The reason is that most genes remain the same, regardless of the stressor. Pozhitkov believes that we can overcome this difficulty by solving the system (1) of equations.

The criterion we want for the coefficients of system (1) is similar to the discrimant of quadratic equations or the determinant of systems of linear equations. For example, the discrimant of the quadratic equation $ax^2 + bx + c = 0$ is $b^2 - 4ac$. If $b^2 - 4ac > 0$, then we have two real solutions. If $b^2 - 4ac = 0$, then we have one real solution. If $b^2 - 4ac < 0$, then we have two complex solutions. My focus on this system of equations will be to find the criterion for the values of $a, b_1, b_2, c, d_1, d_2, d_3$ that guarantee finitely many real solutions.

We will solve this system by applying principles of Gröbner bases. Gröbner bases are a tool used to analyze non-linear polynomial systems that generalize Gaussian elimination, and so they are well-suited for analyzing this type of problem.

Chapter 2 describes the background material. In Section 1 of Chapter 2, we describe in detail the origins of the system of equations given above. In Section 2, we describe Gröbner bases. In Section 3, we show how Gröbner bases address the question. Chapter 3 states the result. In Section 1 of Chapter 3, we discuss the approach taken to solve the given system. In Section 2, we state and prove the main result.

CHAPTER 2

Background Material

1. Motivation for Studying the Given System

Our scenario is this: Humans dump contaminants into the water; the fish are exposed to the contaminants; the fish have a reaction to the contaminants. The ribonucleic acid (RNA) reacts to produce new proteins. Microarrays measure the gene expression.

> "Complementary DNA sequences for every gene in a genome (the *probes*) are laid down in great quantity on individual spots on a glass slide or silicon chip. This is the microarray itself. Then the mRNAs from a cellular extract are washed over this array to allow them to find and stick to their complements. By counting the number of transcripts that bind to each spot, we can measure at least the relative abundance of each." [3, pg. 144]

The deoxyribonucleic acid (DNA) is coded with corresponding complements. "This complementarity means that for each A, C, G, T in one strand, there is a T, G, C, or A, respectively, in the other strand." [3, pg. 3] The result for the microarray data is that complements that bind glow; complements that do not bind do not glow.

> "The first approach to the problem is to investigate linear relationships between the hybridization patterns. If patterns of each treatment are unique, then they can be treated as orthogonal vectors in a multidimensional space. Hypothetically, the pattern of an unknown stress can be either a linear combination of these vectors or there would be no such combination possible." [9]

Each stressor sets off certain unique genes. Mathematically, this means that when one stressor activates a gene, the gene's value in the corresponding vector is non-zero. Since the other stressor does not activate this gene, the gene's value in the vector corresponding to the other stressor is zero. Thus the dot product of the two vectors will be zero, implying that they are orthogonal.

FIGURE 1. vector diagram

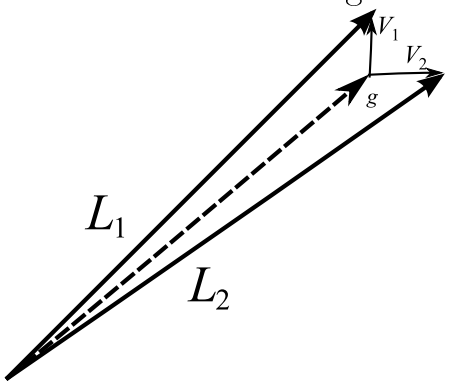


Diagram of the hypothesized relationship between L_i (microarray data), V_i (a unique gene expression for the stressor), and g (genes common to all the stressors). Notice V_1 and V_2 are orthogonal, and $g + V_i = L_i$.

The experiment includes vectors for hypoxia (or lack of oxygen, a vector represented by L_1), cadmium (a heavy metal represented by L_2), chromium (a heavy metal represented by L_3), pyrene (a poison represented by L_4), and a control (represented by L_{ctrl}). Pozhitkov explains, "The experiments were replicated: 4 times for hypoxia and 3 for the rest of the treatments. Gene expression levels were assessed by a microarray containing 16608 probes. In total, my dataset contained 16 arrays of 16608 intensity values, i.e. **hybridization patterns**." [9] However, all the stressors stimulate a common vector, g. "Each of the two vectors are orthogonal to each other. It is the vector g that causes $[L_1]$ and $[L_2]$ to have an angle less than 90 between them. Thus, subtraction of the vector g from $[L_1]$ and $[L_2]$ will result in accentuation of distinctive features of the two vectors." [9] V_1 corresponds to the genes that the first stressor stimulates and the second does not. (See Figure 1.) We want to find unique vectors V_1, V_2, V_3, V_4, and V_{ctrl} for this data set. Since L_{ctrl} is the control, we can hypothesize that all the common genes expressed will be contained in L_{ctrl}. Let A represent the matrix whose columns are the vectors $L_1, ..., L_{\text{ctrl}}$.

We can consider the matrix equation

$$A \cdot x = g.$$

This equation certainly *should* have a solution, because g is a common vector for $L_1, ..., L_{\text{ctrl}}$ and the columns of A are the vectors $L_1, ..., L_{\text{ctrl}}$.

However, we know neither x nor g. We can approximate g in the following fashion: we believe that L_{ctrl} is approximately equal to g, so the substitution

$$A \cdot x = L_{\text{ctrl}}$$

allows us to compute x. However, it is hard to compute x using simple linear algebra because A has more rows (>16000) than columns (5).

Once we know x, we substitute back into $A \cdot x = g$ and we compute g easily. One might expect $g = L_{\text{ctrl}}$ from this but because $A \cdot x = g$ is an overdetermined system this is not actually the case; the solution for x is approximate. Nevertheless, this method does give us a reasonable approximation to g; in tests Drs. Perry and Pozhitkov found that the correlation between the computed g and the given L_{ctrl} was higher than 95%.

After approximating g by L_{ctrl}, we return to the system of equations

$$L_i = x_1 g + V_i.$$

We can rewrite this as

$$V_i = L_i - x_i g.$$

If we knew the x_i, we could solve for V_i, but we don't know the x_i. However, recall that the V_i are pairwise orthogonal. Thus for $i \neq j$,

$$V_i \cdot V_j = 0.$$

By substituting the values of V_1 and V_2 in the dot product we get:

$$(L_1 - x_1 g) \cdot (L_2 - x_2 g) = 0$$

$$L_1 \cdot L_2 - x_2 L_1 g - x_1 L_2 g + x_1 x_2 g \cdot g = 0$$

This simplifies to

$$ax_1x_2 - b_1x_1 - cx_2 + d_1 = 0$$

where $a = g^2, b_1 = g \cdot L_2, c = g \cdot L_1$, and $d_1 = L_1 \cdot L_2$. We determine the other polynomials by taking the dot product of:

$$V_2 \cdot V_3 = 0$$

$$V_1 \cdot V_3 = 0.$$

Together, we have the following system of non-linear polynomial equations:

$$ax_1x_2 - b_1x_1 - cx_2 + d_1 = 0$$

$$ax_1x_3 - b_2x_1 - cx_3 + d_2 = 0$$

$$ax_2x_3 - b_2x_2 - b_1x_3 + d_3 = 0$$

where

$$a = g \cdot g$$

$$b_1 = L_2 \cdot g$$

$$b_2 = L_3 \cdot g$$

$$c = L_1 \cdot g$$

$$d_1 = L_1 \cdot L_2$$

$$d_2 = L_1 \cdot L_3$$

$$d_3 = L_2 \cdot L_3.$$

Recall this is system (1). Such systems can be analyzed precisely using a tool developed in 1964 by Bruno Buchberger called Gröbner bases.

2. Gröbner Bases

Gröbner bases are a way of generalizing row-echelon form for nonlinear polynomials.

> "Gröbner bases are now recognized as an important tool for describing solutions to systems of nonlinear equations. They form a part of all major computer algebra systems, and have found their place as an important application to scientific research in fields such as physics and engineering." [8]

Buchberger first found an algorithm to compute such a basis [1]. Buchberger's algorithm proceeds as follows, given as input a system of polynomials F:

- Set $G = F$.
- Repeat the following until there are no unconsidered pairs:
  - Choose an as-yet unconsidered pair $p, q \in G$ and construct their *subtraction polynomial* S (*S-polynomial*), the difference of the smallest polynomial multiples of p and q such that the leading terms cancel.
  - *Top reduce* S with respect to G. That is, while $t = \mathrm{lt}\,(S)$ remains divisible by $u = \mathrm{lt}\,(g)$ for some $g \in G$, put

$$S := S - \frac{\mathrm{lc}\,(S)}{\mathrm{lc}\,(g)} \frac{t}{u} \cdot g$$

where lt represents the leading term[1] and lc represents the leading coefficient.

- Once no more top-reductions of S are possible, either $S = 0$ or $\mathrm{lt}(S)$ is no longer divisible by $\mathrm{lt}(g)$ for any $g \in G$.
  (1) In the first case, we say that the *S-polynomial of p and q top-reduces to zero with respect to G.*
  (2) In the second case, append S to G. The new entry in G means that the *S-polynomial of p and q* now top-reduces to zero with respect to G. Additional pairs will need to be considered in the first step.

- The algorithm terminates once the *S-polynomials* of all pairs $p, q \in G$ top-reduce to zero.[4]

Example 1. Let's compute the Gröbner basis of:

$$G = \{p = x^2 + 1, q = y^2 + 1\}.$$

---

[1]What is a *leading term?* For univariate polynomials, the answer is easy: the variable with the highest degree, or exponent: $x+x^5+x^3+1$ has x^5 as its leading term. For multivariate polynomials, consider $x^2+y^3+xy^2$. Should the leading term be x^2 or y^3? There are valid mathematical reasons for both.
It turns out that there are infinitely many ways to choose the leading term of an arbitrary polynomial. The two most used are *lexicographic* and *total degree* (technically, *graded reverse lexicographic* but we will refer to it as total degree) [1, 2].

- In lexicographic, choose the leading term alphabetically: that is, choose the term that would come first in the dictionary. In $x^2+y^3+xy^2$, the lexicographic order would choose x^2 as the leading term.
- In total degree, choose the term with the highest sum of exponents. If two terms have the same total degree, then we drop the variable that comes last in the alphabet and recompute the total degree. In $x^2+y^3+xy^2$, the total degree of x^2 is 2, of y^3 is 3, and of xy^2 is $1+2=3$ because $xy^2 = x^1y^2$. We can eliminate x^2 because it has the lowest total degree. Since the total degree of y^3 and xy^2 are the same, we drop the y's. These two terms become 1 and x. The total degree of 1 is 0 (because $1 = x^0$) and the total degree of x is 1, so the leading term of $x^2 + y^3 + xy^2$ is xy^2 because x came from xy^2.

Different ways of choosing the leading term can produce different Gröbner bases, but for the questions in this project, all the methods give the same answers. We use total degree to choose leading terms, since it is usually the most efficient.[2]

Step 1: Generate the subtraction polynomial by multiplying y^2 by p and x^2 by q, because x^2 is the leading term of p and y^2 is the leading term of q. The subtraction cancels the leading terms of G.

$$\begin{array}{r} y^2(x^2+1) \\ \underline{-x^2(y^2+1)} \\ y^2-x^2 \end{array}$$

Step 2: Reduction Step - Top reduce S with respect to G. Here we use x^2+1 because the leading term of S is divisible by the leading term of p.

$$(y^2-x^2)-(-1)(x^2+1)$$

$$y^2-x^2+x^2+1$$

$$S=y^2+1$$

Again, we refer to G to find a polynomial with which the leading term of the new subtraction polynomial is divisible by. In this case, we use $q=y^2+1$.

$$(y^2+1)-\frac{(-1)}{(1)}\frac{(y^2)}{(y^2)}(y^2+1)$$

$$y^2+1-y^2-1=0$$

$$S=0$$

The algorithm now terminates because the *S-polynomial* has top-reduced to zero. Because the only *S-polynomial* of this system reduces to zero, we verify that G is initially a Gröbner basis.

In contrast, we will now look at an example system that is not initially a Gröbner basis.

Example 2. Let $G = \{g_1 = x^2 + y^2 - 4, g_2 = xy - 1\}$. We begin with the elimination step, by finding the subtraction polynomial that cancels the leading terms of g_1 and g_2.

$$\begin{aligned} y(x^2 + y^2 - 4) \\ \underline{-x(xy - 1)} \\ g_3 = y^3 + x - 4y \end{aligned}$$

where y^3 is the leading term of g_3.

Neither of the leading terms of G divides the leading term of g_3 so we must add g_3 to the basis. Since no top reductions are possible, we must consider new pairs. We will first consider the pair (g_1, g_3) by repeating the elimination step with the new polynomials.

$$\begin{aligned} y^3(x^2 + y^2 - 4) \\ \underline{-x^2(y^3 + x - 4y)} \\ S = y^5 - x^3 + 4x^2y - 4y^3 \end{aligned}$$

Now we can top reduce S with respect to g_3 because the leading term of S is y^5 and the leading term of g_3 is y^3.

$$\begin{aligned} \left(y^5 - x^3 + 4x^2y - 4y^3\right) - \frac{(1)}{(1)}\frac{(y^5)}{(y^3)}\left(y^3 + x - 4y\right) \\ S = -x^3 + 4x^2y - xy^2 \end{aligned}$$

Top reduce S with respect to g_1 because the leading term of S is x^3 and the leading term of g_1 is x^2.

$$\begin{aligned} \left(-x^3 + 4x^2y - xy^2\right) - \frac{(-1)}{(1)}\frac{(x^3)}{(x^2)}\left(x^2 + y^2 - 4\right) \\ S = 4x^2y - 4x \end{aligned}$$

Top reduce S with respect to g_2 because the leading term of S is x^2y and the leading term of g_2 is xy.

$$\left(4x^2y - 4x\right) - \frac{(4)}{(1)}\frac{(x^2y)}{(xy)}(xy - 1)$$

$$S = 0$$

Since S top reduced to zero, we must now consider the remaining pair (g_2, g_3). Repeat the process by first finding the subtraction polynomial.

$$\begin{array}{r} y^2(xy - 1) \\ \underline{-x(y^3 + x - 4y)} \\ S = -x^2 + 4xy - y^2 \end{array}$$

Top reduce S with respect to g_1 because the leading term of S is x^2 and the leading term of g_1 is x^2.

$$(-x^2 + 4xy - y^2) - \frac{(-1)}{(1)}\frac{(x^2)}{(x^2)}(x^2 + y^2 - 4)$$

$$S = 4xy - 4$$

Top reduce S with respect to g_2 because the leading term of S is xy and the leading term of g_2 is xy.

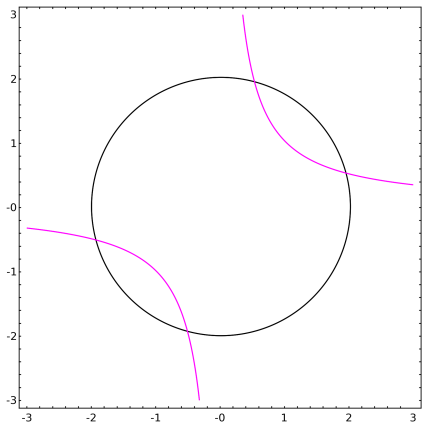
$$(4xy - 4) - \frac{(4)}{(1)}\frac{(xy)}{(xy)}(xy - 1)$$

$$S = 0$$

Since S top reduced to 0, we have no new information, and no new pairs. We have computed the Gröbner basis

$$G = \left\{g_1 = x^2 + y^2 - 4, g_2 = xy - 1, g_3 = y^3 + x - 4y\right\}.$$

FIGURE 2. Plot of $x^2 + y^2 - 4$, $xy - 1$



3. How Do Gröbner Bases Address the Question?

We use Gröbner bases to analyze polynomial systems because they are a nice form from which we can answer many questions such as: Is there a solution? Are there infinitely many solutions? If not, how many are there? We know from [2]:

Theorem 3. *A system F has common solutions if and only if any Gröbner basis G of F has common solutions. This is true if and only if no element of G is a nonzero constant.*

Let's consider a couple of examples.

Example 4. Recall Example 2. The original system was $F = \{x^2 + y^2 - 4, xy - 1\}$. We calculated a Gröbner basis $G = \{x^2 + y^2 - 4, xy - 1, y^3 + x - 4y\}$. Because no element of G is a constant polynomial, we know that F has common solutions. (See Figure 2 generated with the Sage computer algebra system [11]. Notice that the common solutions that are real occur where the circle and hyperbola intersect.)

Now we consider an example with no solutions.

Example 5. Let f_1 be any non-constant polynomial and

$$f_2 = f_1 + 1.$$

Before computing the Gröbner basis, we know that $F = \{f_1, f_2\}$ has no common solutions because if $f_1 = 0$ then $f_2 = 1$.

Now compute the Gröbner basis to compare. Since f_1 and f_2 have the same leading terms, their subtraction polynomial is $f_1 - f_2 = -1$, a constant polynomial. The polynomial -1 does not top reduce with respect to F, so we must add -1, a constant polynomial, to the Gröbner basis. Therefore, no common solutions exist.

How do we decide if there are finitely or infinitely many solutions? Again from [2],

Theorem 6. *A system F has finitely many solutions if and only if for any Gröbner basis G of F, a power of every variable in F appears as a leading term of some polynomial in G.*

For example:

Example 7. Again recall $F = \{x^2 + y^2 - 4, xy - 1\}$ from Example 2. We calculated a Gröbner basis $G = \{x^2 + y^2 - 4, xy - 1, y^3 + x - 4y\}$. The variables of F are x and y. Does a power of x appear as a leading term in G? Yes, x^2 appears as the leading term of $x^2 + y^2 - 4$. Does a power of y appear as a leading term in G? Yes, y^3 appears as the leading term of $y^3 + x - 4y$. Therefore, F has finitely many solutions.

CHAPTER 3

Results

The goal of the following approach is to use theorems 3 and 6 to find the conditions on the coefficients of system (1) that guarantee finitely many real solutions.

1. Approach

First we must compute the Gröbner basis using symbolic, unknown coefficients. To compute the Gröbner bases, we wrote programs in Sage [11] to compute the S-polynomial and to top reduce the polynomials one step at a time. Once we compute the S-polynomial, we look at the leading term. Then we top reduce and look at the new leading term. We continue top reduction until we get zero or a new constant. This constant is a factor of the leading term that could be zero which implies either addition or subtraction of the constant from the leading term.

For top reduction to be complete, either $S = 0$ or we are left with a new polynomial to add to the set, in which case we consider new S-polynomials until all pairs top reduce to zero. However, since the coefficients are represented symbolically, we cannot always be sure we are left with zero. Therefore, we consider a case analysis where each factor of the leading term is either zero or nonzero. Once top reduction is complete, we analyze the factor of the leading term. If the factor is nonzero, then we can stop. Otherwise, the factor is zero which makes the leading term zero. Since this analysis is symbolic, the computer algebra system does not recognize the leading term as zero; thus, we must subtract the leading term manually. Therefore, we can subtract the leading term that equals zero and look at the new leading term.

2. Main Result

As a result of this approach, we get the following theorem:

Theorem 8. *The system* (1) *has finitely many real solutions if and only if*

$$(b_1b_2 - ad_3)(b_2c - ad_2)(b_1c - ad_1) > 0. \tag{2}$$

When the product equals zero, there is either no solution or infinitely many solutions.

Proof. Denote

$$g_1 = ax_1x_2 - b_1x_1 - cx_2 + d_1$$

$$g_2 = ax_1x_3 - b_2x_1 - cx_3 + d_2$$

$$g_3 = ax_2x_3 - b_2x_2 - b_1x_3 + d_3.$$

We consider several cases.

Case 1: $(b_1b_2 - ad_3)(b_2c - ad_2)(b_1c - ad_1) \neq 0$.

The S-polynomial for g_1 and g_2 is $-b_1x_1x_3 + b_2x_1x_2 + d_1x_3 - d_2x_2$. After reducing by g_1 and g_2 we get

$$g_4 = (ad_1 - b_1c)x_3 + (b_2c - ad_2)x_2 - b_2d_1 + b_1d_2.$$

By assumption, $(b_2c - ad_2) \neq 0$. So the leading monomial is x_2 and we cannot reduce any further. Now we add g_4 to the basis.

The S-polynomial for g_1 and g_3 is $b_2x_1x_2 - cx_2x_3 + d_1x_3 - d_3x_1$. After reducing by g_1 and g_3 we get

$$g_5 = (b_1b_2 - ad_3)x_1 + (ad_1 - b_1c)x_3 - b_2d_1 + cd_3.$$

By assumption, $(b_1b_2 - ad_3) \neq 0$. So the leading monomial is x_1 and we cannot reduce any further. Now we add g_5 to the basis.

The S-polynomial for g_2 and g_3 is $b_1x_1x_3 - cx_2x_3 + d_2x_2 - d_3x_1$. After reducing by g_2, g_3, g_4 and g_5 we get zero.

The S-polynomial for g_3 and g_5 is

$$a(b_1c - ad_1)x_3^2 + b_2(-b_1c + ad_2)x_3 + b_2(-b_2c + ad_2)x_2 + d_3(b_2c - ad_2).$$

By assumption, $a(b_1c - ad_1) \neq 0$. So the leading monomial is x_3^2 and we cannot reduce that any further. However, we can eliminate the x_2 term by subtracting a multiple of g_4, obtaining

$$g_6 = a(b_1c - ad_1)x_3^2 + 2b_2(-b_1c + ad_1)x_3 + b_2(-b_2d_1 + b_1d_2) + d_3(b_2c - ad_2).$$

Now we add g_6 to the basis.

The rest of the S-polynomials reduce to zero. Notice that g_6 is quadratic; the leading term is squared and x_3 is the only variable. Because g_6 is quadratic, there is a real solution if and only if the discriminant of g_6 is nonnegative:

$$\begin{aligned}
(2b_2(-b_1c + ad_1))^2 - 4\left(a(b_1c - ad_1)\right)\left(b_2(-b_2d_1 + b_1d_2) + d_3(b_2c - ad_2)\right) &\geq 0 \\
4b_2^2(b_1c - ad_1)^2 - 4ab_2(b_1c - ad_1)(-b_2d_1 + b_1d_2) - 4ad_3(b_1c - ad_1)(b_2c - ad_2) &\geq 0 \\
4(b_1c - ad_1)\left[b_2^2(b_1c - ad_1) - ab_2(-b_2d_1 + b_1d_2) - ad_3(b_2c - ad_2)\right] &\geq 0 \\
4(b_1c - ad_1)\left[b_1b_2^2c - \cancel{ab_2^2d_1} + \cancel{ab_2^2d_1} - ab_1b_2d_2 - ab_2d_3c + a^2d_2d_3\right] &\geq 0 \\
4(b_1c - ad_1)\left[b_2c\left(b_1b_2 - ad_3\right) - ad_2\left(b_1b_2 - ad_3\right)\right] &\geq 0 \\
4(b_1c - ad_1)(b_1b_2 - ad_3)(b_2c - ad_2) &\geq 0.
\end{aligned}$$

Since each factor is positive, we can conclude that two real solutions exist.

Case 2: $(b_1b_2 - ad_3) = 0$ and $(b_2c - ad_2)(b_1c - ad_1) \neq 0$.

We compute g_4 the same way as in case 1. This time, however, g_5 has the form

$$g_5 = (ad_1 - b_1c)x_3 - b_2d_1 + cd_3.$$

The S-polynomial for g_3 and g_5 is

$$a(b_1c - ad_1)x_3^2 + b_2(-b_1c + ad_2)x_3 + b_2(-b_2c + ad_2)x_2 + d_3(b_2c - ad_2).$$

This reduces by g_4 and g_5 to get

$$(-b_1b_2 + ad_3)(-b_2cd_1 - b_1cd_2 + ad_1d_2 + c^2d_3).$$

By assumption the first factor is zero.

The S-polynomial of g_2 and g_5 is

$$\cancelto{0}{c(b_1b_2 - ad_3)}x_1 + c(b_1c - ad_1)x_3 + d_2(-b_1c + ad_1).$$

This reduces by g_5 to

$$g_6 = (-b_1c + ad_1)(-b_2cd_1 - b_1cd_2 + ad_1d_2 + c^2d_3).$$

By assumption the first factor is nonzero. If the second factor is nonzero, then we have a constant polynomial in the Gröbner basis, and there is no solution. Otherwise, g_6 is zero and we continue with S-polynomials. The remaining S-polynomials reduce to zero. No polynomial has a leading monomial that is a power of x_1, so there are infinitely many solutions.

The remaining cases are similar to case 2 and give infinitely many or zero solutions.

□

Appendix: Sample Sage Session

To avoid retyping certain patterns of computation, we define functions when convenient. Here, `spol` computes the S-polynomial of two polynomials p and q. It won't work properly if the inputs aren't functions.

```
sage: %hide
sage: #auto
sage: def degree(t, xs):
...     result = 0
...     for x in xs:
...       result = result + t.degree(x)
...     return result
...
sage: def lm(p): # grevlex or total degree lm of p
...     result = 0
...     mons = p.monomials()
...     coeffs = p.coefficients()
...     for i in range(len(p.monomials())):
...       t = coeffs[i]*mons[i]
...       if result == 0:
...         result = t
...       else:
...         # if total degree of result is smaller than total degree of t,
...         # change result to t
...         if (degree(result,[x1,x2,x3]) < degree(t,[x1,x2,x3])):
...           result = t
...         # if total degree is the same, check individual variables x3,
```

```
...         # then x2, then x1
...         elif (degree(result,[x1,x2,x3]) == degree(t,[x1,x2,x3])):
...           if (degree(result,[x3]) > degree(t,[x3])):
...             result = t
...           elif (degree(result,[x3]) == degree(t,[x3])):
...             if (degree(result,[x2]) > degree(t,[x2])):
...               result = t
...             elif (degree(result,[x2]) == degree(t,[x2])):
...               if (degree(result,[x1]) > degree(t,[x1])):
...                 result = t
...               elif (degree(result,[x1]) == degree(t,[x1])):
...                 result += t
...       return result
...
sage: def lt(p):
...       m = lm(p)
...       d1 = m.degree(x1)
...       d2 = m.degree(x2)
...       d3 = m.degree(x3)
...       return x1**d1*x2**d2*x3**d3
...
sage: def lc(p):
...       return lm(p)/lt(p)
...
sage: def spol(p,q):
...       t = lt(p)
...       u = lt(q)
...       a = lc(p)
...       b = lc(q)
...       v = t.lcm(u)
```

```
...       s = v/(t*a)*p-v/(u*b)*q
...       return s.numerator()
...
sage: def red(p,q):
...       t = lt(p)
...       u = lt(q)
...       a = lc(p)
...       b = lc(q)
...       r = b*p-a*t/u*q
...       return r.numerator()
```

First we define the **ring** of polynomials over which we will work. (A ring is like a group, only with a little more structure.)

```
sage: R.<x1,x2,x3,a,b1,b2,c,d1,d2,d3> = QQ[]
```

We next define G. We want the Gröbner basis of G.

```
sage: g1 = a*x1*x2-b1*x1-c*x2+d1
sage: g2 = a*x1*x3-b2*x1-c*x3+d2
sage: g3 = a*x2*x3-b2*x2-b1*x3+d3
sage: lm(g1),lm(g2),lm(g3)
(x1*x2*a, x1*x3*a, x2*x3*a)
```

Now we create the S-polynomials for g_1, g_2, and g_3.

```
sage: s12 = spol(g1, g2)
sage: lm(s12),s12
(x1*x2*b2, -x1*x3*b1 + x1*x2*b2 + x3*d1 - x2*d2)
sage: s23 = spol(g2,g3)
sage: lm(s23), s23
(x1*x3*b1, x1*x3*b1 - x2*x3*c + x2*d2 - x1*d3)
sage: s13 = spol(g1, g3)
sage: lm(s13),s13
(x1*x2*b2, x1*x2*b2 - x2*x3*c + x3*d1 - x1*d3)
```

Now we reduce S_{13}. First we check its leading monomial to find which polynomial to reduce by, then we perform the reduction.

```
sage: lm(s13)
x1*x2*b2
sage: s13_1 = red(s13,g1)
sage: lm(s13_1)
-x2*x3*a*c
sage: s13_2 = (red(s13_1,g3)/a).numerator()
sage: factor(lm(s13_2)),s13_2
(x1 * (b1*b2 - a*d3), x1*b1*b2 - x3*b1*c + x3*a*d1 - x1*a*d3 - b2*d1 + c*d3)
```

We now assume that reduction is complete. This implies that that $\mathrm{lm}(s13\_2) \neq 0$. That is,

$$b_1 b_2 - a d_3 \neq 0.$$

```
sage: g4 = s13_2
sage: lm(g4)
x1*b1*b2 - x1*a*d3
```

Now, we reduce S_{12}.

```
sage: lm(s12)
x1*x2*b2
sage: s12_1 = red(s12,g1)
sage: s12_1
-x1*x3*a*b1 + x1*b1*b2 + x2*b2*c + x3*a*d1 - x2*a*d2 - b2*d1
sage: lm(s12_1)
-x1*x3*a*b1
sage: s12_2 = (red(s12_1,g2)/a).numerator()
sage: factor(lm(s12_2)), s12_2
(x2 * (b2*c - a*d2), -x3*b1*c + x2*b2*c + x3*a*d1 - x2*a*d2 - b2*d1 + b1*d2)
```

We now assume that reduction is complete. This implies that that $lm(s12\_2) \neq 0$. That is,

$$b_2c - ad_2 \neq 0.$$

```
sage: g5 = s12_2
sage: lm(g5)
x2*b2*c - x2*a*d2
```

Now reduce S_{23}.

```
sage: lm(s23)
x1*x3*b1
sage: s23_1 = red(s23,g2)
sage: s23_1
-x2*x3*a*c + x1*b1*b2 + x3*b1*c + x2*a*d2 - x1*a*d3 - b1*d2
sage: factor(lm(s23_1))
(-1) * c * a * x3 * x2
sage: s23_2 = (red(s23_1,g3)/a).numerator()
sage: factor(lm(s23_2))
x1 * (b1*b2 - a*d3)
sage: lm(s23_2),lc(s23_2)
(x1*b1*b2 - x1*a*d3, b1*b2 - a*d3)
sage: s23_3 = ((red(s23_2,g4)/b1)/b2).numerator()
sage: s23_3
x3*b1^2*b2*c - x2*b1*b2^2*c - x3*a*b1*b2*d1 + x2*a*b1*b2*d2 -
x3*a*b1*c*d3 + x2*a*b2*c*d3 + x3*a^2*d1*d3 - x2*a^2*d2*d3 + b1*b2^2*d1 -
b1^2*b2*d2 - a*b2*d1*d3 + a*b1*d2*d3
sage: factor(lm(s23_3))
(-1) * x2 * (-b2*c + a*d2) * (-b1*b2 + a*d3)
sage: s23_4 = red(s23_3,g5)
sage: s23_4
0
```

Now we will consider g_3 and g_5.

```
sage: s35 = spol(g3, g5)
sage: s35
x3^2*a*b1*c - x3^2*a^2*d1 - x3*b1*b2*c - x2*b2^2*c + x3*a*b2*d1 + x2*a*b2*d2
+ b2*c*d3 - a*d2*d3
sage: factor(lm(s35))
(-1) * a * x3^2 * (-b1*c + a*d1)
```

We now assume that reduction is complete. This implies that that $\mathrm{lm}(s35) \neq 0$. That is,

$$b_1 c - a d_1 \neq 0.$$

```
sage: g6 = s35
```

Now we will consider g_2 and g_4.

```
sage: s24 = spol(g2, g4)
sage: lm(s24)
x3^2*a*b1*c - x3^2*a^2*d1
sage: s24_1 = red(s24,g6)
sage: factor(lm(s24_1))
(-1) * b2 * a * x1 * (-b1*c + a*d1) * (-b1*b2 + a*d3)
sage: s24_2 = red(s24_1,g4)
sage: lm(s24_2)
x2*a*b1^2*b2^3*c^2 - x2*a^2*b1*b2^3*c*d1 - x2*a^2*b1^2*b2^2*c*d2
+ x2*a^3*b1*b2^2*d1*d2 - x2*a^2*b1*b2^2*c^2*d3 + x2*a^3*b2^2*c*d1*d3
+ x2*a^3*b1*b2*c*d2*d3 - x2*a^4*b2*d1*d2*d3
sage: s24_3 = red(s24_2,g5)
sage: s24_3
0
```

Now we will consider g_1 and g_4.

```
sage: s14 = spol(g1, g4)
sage: lm(s14)
```

```
x2*x3*a*b1*c - x2*x3*a^2*d1
sage: s14_1 = red(s14, g3)
sage: lm(s14_1)
-x1*a*b1^2*b2 + x1*a^2*b1*d3
sage: s14_2 = red(s14_1,g4)
sage: s14_2
0
```

We now have a GB. Since the lm's of g_1, g_2, g_3 are redundant, we should be able to discard them. We check that here.

```
sage: red(red(red(g1,g4),g3),g4)
0
sage: red(red(red(red(g2,g4),g6),g4),g5)
0
sage: red(red(g3,g5),g6)
0
```

So the basis consists only of g_4, g_5, g_6. Since g_4 and g_5 are linear, they won't produce quadratic solutions. However, g_6 is quadratic; to see its solutions we should test the discriminant. First we reduce g_6 completely, removing x_2.

```
sage: g6.coefficient(x2)
-b2^2*c + a*b2*d2
sage: lc(g5)
b2*c - a*d2
sage: factor(g6*lc(g5) - g5*g6.coefficient(x2))
(-b2*c + a*d2) * (-x3^2*a*b1*c + x3^2*a^2*d1 + 2*x3*b1*b2*c - 2*x3*a*b2*d1
+ b2^2*d1 - b1*b2*d2 - b2*c*d3 + a*d2*d3)
sage: g6 = ((g6*lc(g5) - g5*g6.coefficient(x2))/(lc(g5))).numerator()
sage: g6
x3^2*a*b1*c - x3^2*a^2*d1 - 2*x3*b1*b2*c + 2*x3*a*b2*d1 - b2^2*d1 + b1*b2*d2
+ b2*c*d3 - a*d2*d3
sage: qu_a = g6.coefficient(x3^2)
```

```
sage: qu_a
a*b1*c - a^2*d1
sage: qu_b = g6.coefficient(x3)
sage: qu_b
-2*b1*b2*c + 2*a*b2*d1
sage: qu_c = g6 - qu_a*x3^2 - qu_b*x3
sage: qu_c
-b2^2*d1 + b1*b2*d2 + b2*c*d3 - a*d2*d3
```

Whether the solutions are real or complex depends on whether the discriminant of a quadratic polynomial is positive or negative.

```
sage: factor(qu_b**2 - 4*qu_a*qu_c)
(4) * (-b2*c + a*d2) * (b1*c - a*d1) * (-b1*b2 + a*d3)
```

Bibliography

[1] Bruno Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalem Polynomideal (An Algorithm for Finding the Basis Elements in the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal)*. Ph.D. dissertation, Mathematical Institute, University of Innsbruck, Austria, 1965. English translation published in the Journal of Symbolic Computation (2006) 475-511.

[2] David Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms: an Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, second edition, 1997.

[3] Nello Cristianini and Matthew W. Hahn. *Introduction to Computational Genomics: A Case Studies Approach*. Cambridge University Press, 2007.

[4] Christian Eder and John Perry. Implementing Faugére's F5 algorithm with reduced Gröbner bases. *Paper accepted for oral presentation to the MEGA 2009 conference in Barcelona, Spain; submission to special issue of the Journal of Symbolic Computation to follow.*

[5] Ben M. Bolstad et al. A comparison of normalization methods for high density oligonucleotide array data based bias and variance. *Bioinformatics*, 2004.

[6] Rafael Irizarry et al. Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics*, pages 249–64, 2003.

[7] Rafael Irizary et al. Summaries of affymetrix genechip probe level data. *Nucleic Acids Res.*, page 15, 2003.

[8] John Perry. *Combinatorial Criteria for Gröbner Bases*. Ph.D. dissertation, North Carolina State University, Raleigh, NC, 2005.

[9] Alexander Pozhitkov. Private communication, 2008.

[10] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical Analysis*. Springer-Verlag New York, Inc., 2000.

[11] William Stein. *Sage: Open Source Mathematical Software (version 3.4)*. The Sage Group, 2008. `http://www.sagemath.org`.