Solving Polynomial Equation Systems

by Groebner Type Methods

Herbert Melenk
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Heilbronner Str. 10, D-1000 Berlin 31
Federal Republic of Germany
E--mail: melenk@sc.zib-berlin.dbp.de

This lecture was given at the opening session of the CAN expertise centre, Amsterdam, December 1989.

1. Introduction

Multivariate polynomial equation systems of the type

$$0=f_j(x_1,\ldots,x_k)=\sum_j c_{jl}x_1^{i_{1jl}}\cdots x_k^{i_{kjl}},$$
$$l=1,\ldots,n,\quad i_{mjl}\in\mathbb{N},\quad c_{jl}\in\mathbb{Q}\quad(\text{or}\quad\mathbb{Z}),$$

arise in many areas of computational sciences. They are derived, for example, from the following classes of problems:

- *Geometrical descriptions.* Here the variables typically represent cartesian coordinates, distances, angles (embedded in trigonometric functions). Examples are the geometrical problems in robotics, where one asks, if a specially designed robot can reach a specific position in order to perform its task.
- *Steady state analysis of differential equation systems.* Given an explicit system of ordinary differential equations with polynomial right-hand sides, we may ask for the steady states. This is algebraically described by vanishing left-hand sides and the result is an algebraic polynomial equation system. Examples are systems which describe the kinetics of chemical reaction systems, where the steady state might represent a stable production situation in a chemical reactor. Examples are found in [17].
- *Truncated power series.* Given a problem where an *ansatz* with formal power series is substituted into an equation, the vanishing of the coefficients of corresponding powers is a necessary condition for the power series to constitute a solution of the problem. Then these coefficients often are polynomials, thus mapping the original problem to a system of algebraic equations.

CWI Quarterly 3, 121-136

In many cases it is desirable to find algebraic solutions or at least some algebraic information about the possible solutions. In contrast to numerical approximation, the algebraic approach typically gives an answer which is definite, complete and not disturbed by rounding errors. On the other hand, solutions in closed forms are available only for a limited range of problems. For example, the Galois theory has shown that there is no closed formal description for the roots of a general polynomial of degree above four, and with systems of polynomials this situation is not any simpler. Nevertheless, statements about the cardinality of solutions are possible. The closed form solutions—even if algebraically available—often are very large and hard to compute. So one target of the practical computation with systems of polynomials is, to decompose the system as much as possible into simpler systems. This facilitates the computation and improves the quality of the result. In recent years the Groebner technology has been developing in that direction. So today we have a growing set of tools for these problems available in modern computer algebra systems like REDUCE [14,18].

The term 'polynomial' in this context encloses cases, which can be transformed into polynomial systems, e.g.:

- *Systems of rational equations.* A quotient can be zero only if its numerator vanishes; so if one omits the denominators, one can apply polynomial methods to the numerators and handle the zeros of the denominators separately.
- *Systems with algebraic functions.* To each algebraic function a defining polynomial is associated; this polynomial can be used to eliminate the nonpolynomial part of the algebraic function. For instance, the equation

$$x+\sqrt{y}-22=0$$

can be replaced by

$$x+q-22=0, \quad q^2-y=0$$

with a new variable q and one additional polynomial.
- *Systems with trigonometric functions.* A similar technique can be used to introduce the algebraic relations between *sin* and *cos* of an angle, here using $\sin\alpha$ and $\cos\alpha$ as formally independent variables, which are linked by the additional polynomial relation

$$\sin^2\alpha+\cos^2\alpha-1=0.$$

However, one limitation of such transformations lies in the fact that algebraic methods for multivariate polynomials in general suffer from an increasing number of variables. So typically a Groebner calculation with a problem of order n may be possible with a reasonable computing time, while the same problem with order $n+1$ is impossible.

2. Algebraic background

2.1. Polynomial equations and ideal theory

Let $f_1(x_1, \ldots, x_k), \ldots, f_n(x_1, \ldots, x_k)$ be a set of polynomials with the unknowns $x_1, \ldots, x_k$ and coefficients from some field or Euclidean domain. We look at the set of equations

$$f_i(x) = 0, \quad i = 1, \ldots, n \tag{1}$$

where x abbreviates the vector $(x_1, \ldots, x_k)$. We say that a vector $\bar{x}$ is a *solution* of the system

$$G = \{ f_i(x) \mid i = 1, \ldots, n \}$$

if all equations in (1) are satisfied for $x = \bar{x}$. It is obvious that the set of solutions remains unchanged if we add to G new combinations of polynomials:

$$G \cup \{ q(x)\bar{f}(x) + p(x)\tilde{f}(x) \mid \bar{f}, \tilde{f} \in G \}$$

with arbitrary polynomials q and p. The algebraic closure defined by this invariance is that of the *polynomial ideal*, denoted by $I = (f_i(x))$. The ideal is defined by the axioms

$$a \in I \Rightarrow r \cdot a \in I \text{ for arbitrary polynomials } r$$
$$a, b \in I \Rightarrow a + b \in I.$$

A set $B = \{g_i\}$ is said to be a *basis* of an ideal I, if all elements of I can be written as a linear combination from the g_i:

$$a \in I \Rightarrow a = \sum c_i g_i \text{ for certain polynomials } c_i.$$

The ideal theory was developed from the late 19th century on, mainly by D. Hilbert, E. Nöther, B.L. v.d. Waerden and W. Groebner. From that theory it is well known that all members of $I = (f_i(x))$ vanish on the set of solutions of G and that the system $\{f_i\}$ can be replaced by any other basis of the ideal $(f_i(x))$ without modifying the set of solutions.

Bruno Buchberger's concept of a *Groebner basis* [3] and his algorithm for its computation in 1965 opened a new computational approach to polynomial ideals which can be exploited for many types of questions in this context [5]. The search for solutions of (1) is a prominent aspect to which the rest of this paper is devoted.

2.2. The Buchberger algorithm

To set up the Groebner basis theory, we need a total ordering, denoted by $\succ$, on the monomials occurring in the polynomials. Different orderings are possible; the only restriction is that the ordering has to be compatible with multiplication:

$$m_1 \cdot m_2 \succcurlyeq m_1, m_2 \text{ and } m_1 \succcurlyeq m_2 \Rightarrow m \cdot m_1 \succcurlyeq m \cdot m_2 \text{ for monomials } m, m_1, m_2.$$

The most important term of a polynomial f with respect to $\succcurlyeq$ (that is, the term $cx_1^{i_1} \cdots x_k^{i_k}$ of f for which the corresponding monomial $x_1^{i_1} \cdots x_k^{i_k}$ is largest) then is denoted by $ht(f)$ ('head term').

The following are prominent examples for such orderings:

- *Lexicographical ordering.* Based on a fixed sequence of variables the more important monomial is the one with the bigger exponent in the leftmost place;
- *Graduated ordering.* The monomial with the higher total degree (= sum of exponents) comes first; if the total degrees coincide, the lexicographical ordering is applied.

Basic operations among the polynomials (ordering fixed) are:

- *reduce*:
  If $ht(f_j)$ divides a term, say t, of f_i replace

  $$f_i \text{ by } f_i - q \cdot f_j$$

  where q is selected in such a way that the term t vanishes. For example

  $$f_1 = 4x^2 + 4y^2 - 1,\ f_2 = x - y : f_1 \leftarrow f_1 - 4x f_2$$

- *combine* (S-*polynomial*):
  Compute from f_i, f_j the new polynomial

  $$S(f_i, f_j) = p \cdot f_i - q \cdot f_j,$$

  where p, q are selected in such a way that the head terms cancel. For example

  $$f_1 = 4x^2 + 4y^2 - 1,\ f_2 = xy - 1 \Rightarrow S(f_1, f_2) = y f_1 - 4x f_2.$$

The Buchberger algorithm is constructed with these operations. In simplified form it can be written as

Input: $G = \{f_1, \ldots, f_n\}$
  $P := \{(p,q) \mid p,q \in G,\ p \prec q\}$
  while $P \neq \{\}$
    $(p,q) :=$ **pop** P
    $s := S(p,q)$
    $h := NF(s,G)$
    if $h \neq 0$
      $P := P \cup \{(h,q) \mid q \in G\}$
      $G := G \cup \{h\}$
  return G

with the subalgorithm for the normal form

```
NF(s, G)
  while ∃q ∈ G with ht(q) divides a term t of s
        s := s − (t / ht(q))·q
  return s
```

Buchberger [3] has proved that given a compatible ordering,

- the algorithm ends,
- its result is a Groebner basis.

It should be mentioned here, that the property 'G is a basis of $(f_1, \ldots, f_n)$' is an invariant of the algorithm.

For a production version many more features have to be added; there are criteria for polynomials to be eliminated from G, for pairs that are known in advance to result in the instance $h = 0$; there is complete freedom for a strategy of selecting the next pair to be processed, but for an efficient implementation this strategy is of the greatest importance; and last but not least the ordering itself is a parameter with an important influence on the algorithm.

2.3. Properties of Groebner bases

If all redundant (superfluous) polynomials are deleted from G and if additionally all reducible terms behind the head terms have been reduced too, the basis is called a 'fully reduced Groebner basis'. In the following we will deal with fully reduced bases only and so we will not mention that predicate any more. A (fully reduced) Groebner basis for a given set f_i of polynomials will be denoted by $GB_{\text{ord}}(f_i)$ where the index denotes that in general different orderings lead to different Groebner bases.

An overview of Groebner basis properties is given in [4] and [5]. Some properties which are most relevant in the context of equation solving are:

- *Canonical forms*:
  - Two ideals are identical if their GB's are equal:

$$(f_i) = (g_j) \Leftrightarrow GB_{\text{ord}}(f_i) = GB_{\text{ord}}(g_j)$$

  - The GB defines an algorithm for computation modulo the ideal (f_i):

$$p \bmod (f_i) \equiv NF(p, GB(f_i))$$

  - The GB defines an algorithm for the decision of ideal membership:

$$p \in (f_i) \Leftrightarrow p \bmod (f_i) = 0$$

  - The dimension of the ideal (or (in case of dimension 0) cardinality of solutions) can be read off immediately from the Groebner basis.
- *Elimination / Minimality*:
  - The polynomial p in ideal (f_i) which is minimal with respect to the order is guaranteed to be an explicit member of the GB.
  - When the ordering defines a lexicographical relation between two variables, e.g.

$$x^n \succ_{\text{lex}} y^m \quad \text{for all} \quad n,m>0,$$

and there are polynomials in the ideal which do not depend on x, then in the *GB* all multiples of x are eliminated from the biggest possible part of the polynomials. The dependency pattern then has the shape

$$\begin{aligned} &g_1(\ldots x \ldots y \ldots)\\ &\vdots\\ &g_l(\ldots x \ldots y \ldots)\\ &g_{l+1}(\ldots y \ldots)\\ &\vdots\\ &g_m(\ldots y \ldots) \end{aligned}$$

- If the complete lexical ordering is used, the dependency pattern is of triangular shape

$$\begin{aligned} &g_1(x_1,\ x_2,\ \ldots,\ x_n,x_{n+1},\ \ldots,x_k)\\ &\qquad\ \vdots\\ &g_l(\qquad x_2,\ \ldots,\ x_n,x_{n+1},\ \ldots,x_k)\\ &\qquad\qquad\ \vdots\\ &g_m(\qquad\qquad\ x_n,x_{n+1},\ \ldots,x_k) \end{aligned}$$

- The complete triangular pattern ($n=k$) is guaranteed in the case of zero dimensional ideals (that is, the solutions form a finite set of isolated points).

3. Equation solving with Groebner Bases

Based on lexicographical ordering. The simplest approach to problem (1) is to compute

$$G = GB_{\text{lex}(x_1,\ldots,x_k)}(f_1(x_1,\ldots,x_k),\ldots,f_n(x_1,\ldots,x_k))$$

or the lexicographical *GB* for some permutation of the $(x_1,\ldots,x_k)$. If the solution space is zero dimensional, the last polynomial is a univariate polynomial in x_k. If its degree is less than or equal to 4, it can be solved algebraically with the Cardano formulae, even if the coefficient domain contains formal parameters. If the degree is higher, the roots can be approximated to arbitrary accuracy using one of the classical methods, e.g. the Sturm or Uspensky techniques. For example, REDUCE contains a well elaborated package for the computation of polynomial roots to arbitrary precision. These roots then can be substituted into the remaining elements of the *GB*, giving now a univariate polynomial in x_{k-1}. So step by step all points of the zero dimensional solution set are determined either as closed forms or as approximations with arbitrary precision.

EXAMPLE. Modified Edelstein (chemical reaction network) with symbolic coefficients:

$$\alpha c_1 - \beta c_1^2 - \gamma c_1 c_2 + \epsilon c_3,$$
$$-\gamma c_1 c_2 + \epsilon c_3 + 2\theta c_3 - 2\eta c_2^2,$$
$$\gamma c_1 c_2 - \epsilon c_3 - \theta c_3 + \eta c_2^2.$$

The Groebner Basis with respect to (c_3, c_2, c_1) here is

$$\{c_3\epsilon - c_2 c_1 \gamma,\ c_2^2 \epsilon\eta - c_2 c_1 \gamma\theta,\ c_1^2\beta - c_1\alpha\}.$$

Here the third polynomial can be decomposed formally and with substituting back we immediately get the solutions

$$\{\ \{c_3 = 0, c_2 = 0, c_1 = \alpha/\beta\},$$
$$\{c_3 = 0, c_2 = 0, c_1 = 0\},$$
$$\{c_3 = (\alpha^2\gamma^2\theta)/(\beta^2\epsilon^2\eta),\ c_2 = (\alpha\gamma\theta)/(\beta\epsilon\eta),\ c_1 = \alpha/\beta\}\ \}.$$

In the following example ([8,11], here with four variables)

$$X + Y + Z + T,$$
$$XY + XT + YZ + ZT,$$
$$XYZ + XYT + XZT + YZT,$$
$$XYZT + 1$$

the ideal is not zero-dimensional: its *lex* basis is

$$\{X + Y + Z + T,$$
$$Y^2 + 2YT + T^2,$$
$$YZ - YT - Z^2T^4 + ZT - 2T^2,$$
$$YT^4 + Y + T^5 + T,$$
$$Z^3T^2 + Z^2T^3 + Z + T,$$
$$Z^2T^6 + Z^2T^2 + T^4 + 1\}$$

where the last polynomial is bivariate in (z,t). Here the structure of the solution set, which is a manifold parametrized by either z or t, becomes obvious. If we fix either z or t, then the corresponding isolated roots can be determined by substituting back.

Alternatives to the direct lexicographical approach. Generally, the lexicographical Groebner basis gives the best insight into the structure of the ideal, as far as the solution of (1) is concerned. On the other hand, lexicographical bases are the hardest to compute: their calculation often explodes in space and time because of an enormous growth of intermediate expressions (number of polynomials, coefficient size). Here different methods have to be used instead. The most important technique is decomposition of the ideal by factorization, which will be discussed in the following section in more detail. But other orderings

may help here (sometimes in connection with decomposition techniques) too.

The dimension of the ideal is a property invariant with respect to representation. So the dimension can be computed from any *GB* with arbitrary ordering. Here the **graduated orderings** can be used, which in general allow a much faster *GB* computation.

A graduated *GB* has no guaranteed elimination property, but instead it finds the polynomial(s) with minimal total degree in the ideal, which often is very useful information too; e.g., if there are polynomials of degree 1, these are the best possible solutions.

A graduated *GB* has the canonical properties as well. So the congruence 'modulo (f_i)' is computable as well, which in the following cases can be directly used for the following procedures:

- *Zero dimensional ideal.* The graduated *GB* can be converted to a lexicographical one (by the *FGLM* algorithm of Faugere, Gianni, Lazard & Mora [11]) by solving linear systems derived from the congruence. In general the graduated basis plus subsequent conversion needs much less resources than a direct approach via the *lex* ordering. Nevertheless, this may be a hard job too.
- *Zero dimensional ideal with numerical coefficients.* From the congruence established by the graduated *GB*, the isolated solutions can directly be approximated by the second part of the algorithm by Auzinger & Stetter [1]. Here the solutions are found as eigenvalues of a linear transformation modulo the ideal. This computation is applicable if there is one variable with all distinct values in the solutions (the algorithm detects if that is not the case).

For special questions special orderings can be applied. If one is interested in isolating a partial set of variables (finding those polynomials which do not/do only depend on them), it is not necessary to do a *lex* computation; here a stepwise graduated ordering serves the same purpose with much less effort. In a stepwise ordering the set of variables is divided into two or more groups. These groups enter a *lex* type ordering with their local total degrees, thus guaranteeing the elimination property groupwise. This can be used e.g. to isolate a set of parameters in the ideal.

4. Decomposition of ideals

Factorization. If during a Groebner basis computation we are able to find, in the ideal I with an intermediate basis $(h_1(x), \ldots, h_n(x))$, a polynomial $g(x)=g_1(x)\cdot g_2(x)$ with non-constant g_1 and g_2, we know (assuming a coefficient domain with characteristic zero) that g can only vanish for some x if either $g_1(x)$ or $g_2(x)$ vanishes. On the other hand we can add $g(x)$ to any basis of I without modifying I and so we obtain a basis of the form

$$(g_1(x)\cdot g_2(x), h_1(x), \ldots, h_n(x)).$$

We see immediately that a solution in I is a solution in either

$$I_1=(g_1(x),h_1(x),\ldots,h_n(x)) \quad \text{or} \quad I_2=(g_2(x),h_1(x),\ldots,h_n(x)).$$

On the other hand, the total degrees of g_1 and g_2 are below the total degree of g; so, if g was member of the basis already, the bases of I_1 and I_2 are definitely of lower degree than that of I. The Groebner bases for I_1 and I_2 now can be computed in separate cycles of the Buchberger algorithm, such that the complete task forks into two separate ones. The decomposition effect is twofold: The solution space is decomposed, giving a better insight in its structure, and the order of the algorithm is decreased. In general the sum of computing times for the branches is significantly below the computing time for the undecomposed problem.

Of course, a g might have more than two factors and then a higher order forking takes place. Also, each branch can fork in a recursive style, if more factorable polynomials are found.

Not necessarily all branches lead to significant parts of the solution. If one of the factors does not have a zero in common with the rest of the basis, its branch will end with the ideal (1); this in general will be determined soon, and the effect of the lower order for the other branch remains in effect. It may happen that the ideal I_2 is identical to I_1, although g_2 is not identical to g_1; in this case the branch is superfluous and this property can be verified when g_1 is found as member of I_2.

Passive factorization. The simplest approach for finding factorizations is to inspect all intermediate polynomials during the run of Buchberger's algorithm. In the success case forking takes place. This technique was used in several contexts [7,8,16], and especially [8] describes an important approach for lowering the costs of that analysis.

EXAMPLE. Consider once again the chemical reaction network of Section 3:

$$\begin{aligned} 0 &= -c_1^2\beta - c_1c_2\gamma + c_1\alpha + c_3\epsilon \\ 0 &= -c_1c_2\gamma - 2c_2^2\eta + c_3\epsilon + 2c_3\theta \\ 0 &= c_1c_2\gamma + c_2^2\eta - c_3\epsilon - c_3\theta. \end{aligned}$$

Here the algorithm soon detects a polynomial h: $-c_1^2\beta + c_1\alpha$; it can be factored into a product *variable* · *polynomial,* here

$$c_1\cdot(-c_1\beta+\alpha).$$

The algorithm detects more factorizations of this type and generates separate calculation branches, which result in three Groebner bases:

$$\{c_3,c_2,c_1\},$$
$$\{c_3,c_2,c_1\beta-\alpha\},$$
$$\{-c_3\beta^2\epsilon^2\eta+\alpha^2\gamma^2\theta, c_2\beta\epsilon\eta-\alpha\gamma\theta, c_1\beta-\alpha\}.$$

Here the first basis represents the trivial solution $c_3=c_2=c_1=0$; the second basis has one non-zero value $c_1=\alpha/\beta$. The third basis offers the unique

positive solution

$$c_3 = \frac{\alpha^2\gamma^2\theta}{\beta^2\epsilon^2\eta}, \quad c_2 = \frac{\alpha\gamma\theta}{\beta\epsilon\eta}, \quad c_1 = \frac{\alpha}{\beta}.$$

Sometimes, especially with high variable numbers, so many factorizations are found, that the problem tree explodes too. This effect can be controlled by a careful combination algorithm [16] which at the branching point looks ahead into the computation and tries to detect common parts, especially common future factorizations which then can be included in the fork design.

Active factorization. With the above approach, the forking is more or less casual; it depends highly on the strategy by which decisions are taken. There are many cases where the ideal contains factorable polynomials but they will not be found by any strategy. For example, the system

$$\{x^2 + 2xy + 101, y^2 - 101\}$$

is already a Groebner basis with the *lex* ordering over $\mathbb{Z}$. It is obvious that the sum of the two polynomials factors nicely, but this sum will never occur in the Buchberger algorithm.

There are two novel approaches for a more systematic factorization:

- *Testfactors.* If there is the assumption, that a factor $q(x)$ might lead to a significant decomposition of the ideal, the expressions

  $$r(x) = \text{remainder}\,(h(x), q(x))$$

  can be investigated; if there is a syzygy among these remainders (that is a linear dependency in the ideal of remainders), the corresponding combination of the h's is (if not zero) a polynomial in I that has q as a factor. So we have a way to look actively for multiples of $q(x)$ in I. This technique corresponds algebraically to an ideal quotient computation, which here is performed quasi on line. It can be executed for several testfactors simultaneously. Testfactors often arise from obvious properties of the polynomials: for example homogeneous systems tend to monomial factors. Or they are derived from the insight in the (e.g. physical) background of the underlying application.
- *Symmetries.* In many applications, e.g. from physics, the systems are symmetric with respect to some or all variables: the ideal is invariant under some permutations of its variables (rotation, pairwise, exchange, ...). In [12] a new approach for some classes of this type is given, where polynomials in the ideal are detected which in advance are known to factor. This leads to a decomposition of the ideal before the Buchberger algorithm starts and thus in many cases to a dramatic decrease of the complexity/computing requirements, so that much larger systems become solvable.

Arithmetic restrictions. A second source for decomposition is the formal restriction that only nonnegative real values are allowed as components of solutions. This restriction often arises if the variables represent nonnegative

physical quantities like masses, distances, ... So we can exploit a set of inequalities

$$x_i \geqslant 0,\ i = 1, \ldots, k.$$

In generalizing Descartes rule of signs we can state for a multivariate polynomial

$$p = a_0 + a_1 x_1^{i_{11}} \cdots x_k^{i_{1k}} + \cdots + a_m x_1^{i_{m1}} \cdots x_k^{i_{mk}}$$

where all a_i are real and have the same sign:
if $a_0 \neq 0$ then $p(x_1, \ldots, x_k) = 0$ implies that at least one $x_i < 0$;
for $a_0 = 0$ then $p(x_1, \ldots, x_k) = 0$ and $x_i \geqslant 0$ implies, that each monomial in p is already zero.

In other words: If $a_0 \neq 0$ there cannot exist a solution which satisfies the above inequalities and the calculation branch can be cancelled. For $a_0 = 0$, the polynomial p is decomposed into a list of monomials each of which has to be zero. As these monomials are immediately factorizable, we easily get a twofold decomposition. For example, the polynomial

$$17xy^2r + 4x^2z$$

leads to the decomposition

$$x = 0 \vee (y = 0 \wedge z = 0) \vee (r = 0 \wedge z = 0)$$

which describes the nonnegative real solutions.

Arithmetic restrictions together with testfactors and symmetries are means to make present knowledge about the problem to be solved available and exploitable for the solver, thus enabling computations which would be infeasible otherwise. These techniques were successfully applied to complicated problems given by H. Caprasse [6] and A. Noonburg [20], Speer [21].

5. Some examples

Here we present some examples where the effect of ideal decomposition is demonstrated:

Ebert, Deuflhard & Jaeger [10]: Cyclohexane.

The molecular geometry of a cyclic carbon hydrogen molecule with six nodes is described by four equations; these are calculated from the two determinants

$$f_1=\begin{vmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 8/3 & y_1 & 8/3 \\ 1 & 1 & 0 & 1 & 8/3 & y_2 \\ 1 & 8/3 & 1 & 0 & 1 & 8/3 \\ 1 & y_1 & 8/3 & 1 & 0 & 1 \\ 1 & 8/3 & y_2 & 8/3 & 1 & 0 \end{vmatrix} \quad f_4=\begin{vmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 8/3 & y_1 & 8/3 & 1 \\ 1 & 1 & 0 & 1 & 8/3 & y_2 & 8/3 \\ 1 & 8/3 & 1 & 0 & 1 & 8/3 & y_3 \\ 1 & y_1 & 8/3 & 1 & 0 & 1 & 8/3 \\ 1 & 8/3 & y_2 & 8/3 & 1 & 0 & 1 \\ 1 & 1 & 8/3 & y_3 & 8/3 & 1 & 0 \end{vmatrix}$$

The first equation below is $0=f_1$, the second and third equations are calculated from f_1 by interchanging y_1, y_2 and y_3 in a cyclic manner and the fourth equation is $0=f_4$.

$$\begin{aligned} 0 = {} & (-81y_1^2y_2^2+594y_1^2y_2-225y_1^2+594y_1y_2^2-3492y_1y_2 \\ & -750y_1-225y_2^2-750y_2+14575)/81 \\ 0 = {} & (-81y_2^2y_3^2+594y_2^2y_3-225y_2^2+594y_2y_3^2-3492y_2y_3 \\ & -750y_2-225y_3^2-750y_3+14575)/81 \\ 0 = {} & (-81y_1^2y_3^2+594y_1^2y_3-225y_1^2+594y_1y_3^2-3492y_1y_3 \\ & -750y_1-225y_3^2-750y_3+14575)/81 \\ 0 = {} & (162y_1^2y_2^2y_3+162y_1^2y_2y_3^2-1188y_1^2y_2y_3-450y_1^2y_2 \\ & -450y_1^2y_3+3300y_1^2+162y_1y_2^2y_3^2-1188y_1y_2^2y_3 \\ & -450y_1y_2^2-1188y_1y_2y_3^2+5184y_1y_2y_3+5100y_1y_2 \\ & -450y_1y_3^2+5100y_1y_3-7150y_1-450y_2^2y_3+3300y_2^2 \\ & -450y_2y_3^2+5100y_2y_3-7150y_2+3300y_3^2-7150y_3-60500)/81 \end{aligned}$$

Here the algorithm detects nonhomogeneous factorizations and splits the problem into seven separate Groebner bases:

$\{\{3y_1-11,9y_2-25,3y_3-11\},\{9y_1-25,3y_2-11,3y_3-11\},$

$\{3y_1-11,3y_2-11,3y_3-11\},$

$\{3y_1y_2+3y_1y_3-22y_1+3y_2y_3-22y_2-22y_3+121,$

$27y_1y_3^2-198y_1y_3+75y_1+27y_2y_3^2-198y_2y_3+75y_2-198y_3^2+1164y_3+250,$

$81y_2^2y_3^2-594y_2^2y_3+225y_2^2-594y_2y_3^2+3492y_2y_3+750y_2+225y_3+750y_3-14575\},$

$\{3y_1+5,3y_2+5,3y_3+5\},\{3y_1-19,3y_2+5,3y_3+5\},$

$\{3y_1+5,3y_2-19,3y_3+5\}\}$

CAPRASSE & DEMARET [6]: MODEL FROM ASTROPHYSICS:

$$
\begin{aligned}
3x_1^3 - x_1(3x_1a_1 + 3a_1 + 3a_2 - 6) + a_2 - a_3 &= 0,\\
3x_2^3 - x_2(3x_2a_1 + 3a_1 + 3a_2 - 6) + a_2 - a_3 &= 0,\\
3x_3^3 - x_3(3x_3a_1 + 3a_1 + 3a_2 - 6) + a_2 - a_3 &= 0,\\
3x_4^3 - x_4(3x_4a_1 + 3a_1 + 3a_2 - 6) + a_2 - a_3 &= 0,\\
3x_5^3 - x_5(3x_5a_1 + 3a_1 + 3a_2 - 6) + a_2 - a_3 &= 0,
\end{aligned}
$$

where

$$a_i = x_1^i + x_2^i + x_3^i + x_4^i + x_5^i, \quad i = 1,2,3.$$

It took little time to find the ideal decomposition. In fact, we are able to compute the solutions for the Caprasse/Demaret systems up to nine variables; the computation requires several days on a modern workstation.

SPEER [21]: SYSTEM OF POLYNOMIAL EQUATIONS.

$$
\begin{aligned}
4\cdot\beta\cdot(n + 2\cdot a_1 - 8\cdot x_1)\cdot(a_2 - a_3) - x_2\cdot x_3\cdot x_4 + x_2 + x_4 &= 0,\\
4\cdot\beta\cdot(n + 2\cdot a_1 - 8\cdot x_2)\cdot(a_2 - a_3) - x_1\cdot x_3\cdot x_4 + x_1 + x_3 &= 0,\\
4\cdot\beta\cdot(n + 2\cdot a_1 - 8\cdot x_3)\cdot(a_2 - a_3) - x_2\cdot x_1\cdot x_4 + x_2 + x_4 &= 0,\\
4\cdot\beta\cdot(n + 2\cdot a_1 - 8\cdot x_4)\cdot(a_2 - a_3) - x_1\cdot x_3\cdot x_2 + x_1 + x_3 &= 0,
\end{aligned}
$$

where

$$
\begin{aligned}
a_1 &= x_1 + x_2 + x_3 + x_4,\\
a_2 &= x_1\cdot x_2\cdot x_3\cdot x_4,\\
a_3 &= x_1\cdot x_2 + x_2\cdot x_3 + x_3\cdot x_4 + x_4\cdot x_1.
\end{aligned}
$$

This system over $\mathbb{Z}[\beta,n]$ has been solved here for the first time; it was decomposed into ten different bases with complete separation of variables ([12]).

6. PRACTICAL CONSIDERATIONS WHEN SOLVING PROBLEMS

With the background of a great number of available Groebner tools, it is hard to design a general guideline how to proceed with a fresh equation system. With the state of knowledge of today, it is too early to design a Groebner expert system. Nevertheless some steps can be defined here which should enable the non-experienced user of the REDUCE Groebner package [18] or a comparable system to do an investigation with success.

Variables/Parameters. First, the variables and parameters (if any) have to be identified. For *lex* type calculations there is freedom to handle parameters as variables of lowest order or to handle them as part of the coefficient domain. There is not much difference here and the results are comparable; in general the inclusion as variables is preferable. For other orderings, the parameters

must be excluded from the variable list in order to get a proper insight into the behavior of the variables in the ideal.

If the calculations with a parameterized ideal become hard, it can be useful to do a piloting calculation where the parameters are substituted by (non-trivial!) numeric values. If that calculation is hard too, there is no chance to succeed with the general case. A growth of parameters in the substituted problem signals high degrees of the parameters in the coefficients.

Factoring. In most cases the best approach is to execute the factorizing Groebner calculation from the very beginning on. Of course, the factorizer causes some additional overhead, but if any factorization is found it is worth the effort. Only in the case where no factorization is found for a long computing time, or where it is known from other sources that a factorization is not to be expected, the non-factoring Groebner algorithm should be used. The factorization is independent of the ordering; however, different decompositions may be found with different orderings.

If the factoring is successful, each partial basis represents an ideal of its own and further local processing can take place. It will often be the case that some bases represent narrow special aspects from the original problem (e.g. trivial solutions), while others contain more interesting portions.

Restricting the variable ranges to the nonnegative section can facilitate a factoring computation by an important amount, and ruling out trivial results—if these are not interesting by themselves—may also cut down the computing time.

Ordering. Starting with a graduated ordering is the best in most cases, e.g. with *revgradlex* which is the 'fastest' ordering from the standpoint of complexity. If the graduated basis is not found, there is almost no chance of finding a basis with a different order without any additional information. Only in very rare cases a different factorization may cause an earlier decomposition which allows for an easier solution.

Once a graduated basis is found, the dimension of the ideal should be investigated by calculating the Hilbert polynomial of that basis. If it is a constant, the ideal is zero-dimensional and its value is the number of isolated solution points.

If the ideal is zero-dimensional, one can proceed in several directions:

- Convert the basis to a full *lex* basis by using the FGLM algorithm.
- If the application has numerical coefficients only, or if a version with all parameters substituted is relevant, approximations for the solution points can be calculated using the Auzinger/Stetter approach.

7. Conclusion

The recent progress in the Groebner technology has made it a useful tool of computer algebra. Especially the decomposition techniques have moved the horizon of computability so dramatically that real life problems of moderate size can be handled effectively. The development of this technique is still

going on in several places all over the world and, in contrast to many other fields of computer science, an important contribution from Europe takes place here.

8. Appendix

Because of the limited space only a few small examples for Groebner applications could be mentioned here. More examples are found in [2,5,18] and in the test series for the REDUCE Groebner package available via electronic mail from the REDUCE netlib. An additional collection of more advanced application cases can be obtained directly from the Konrad-Zuse-Zentrum für Informationstechnik Berlin.

References

1. Auzinger, Stetter (1988). An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations. R. Agarwal, Y.M. Chow, S.J. Wilson (eds.). *Numerical Mathematics Singapore 1988*, ISNM, Vol. 86, Birkhäuser Verlag Basel.
2. W. Boege, R. Gebauer, H. Kredel (1986). Examples for solving systems of algebraic equations by calculating Groebner Bases. *J. Symbolic Computation 2*, 83-98.
3. B. Buchberger (1965). *An Algorithm for Finding a Basis for a Residue Class Ring of a Zero-Dimensional Polynomial Ideal* (German), Ph.D.Thesis, Univ. of Innsbruck, Austria, Dept. of Mathematics .
4. B. Buchberger (1985). Groebner Bases: an algorithmic method in polynomial ideal theory. N.K. Bose (ed.). *Progress, directions and open problems in multidimensional systems theory*, 184-232, Dordrecht, Reidel.
5. B. Buchberger (1988). Applications of Groebner Bases in non-linear computational geometry. R. Janssen (ed.). *Trends in Computer Algebra*, 52-80, Berlin, Heidelberg.
6. H. Caprasse, J. Demaret (1989). *Private communication.*
7. S.R. Czapor (1989). Solving algebraic equations via Buchberger's algorithm. J.H. Davenport (ed.). *Eurocal'87*, 260-269, Berlin, Heidelberg.
8. J.H. Davenport (1987). *Looking at a Set of equations*, Bath Computer Science Technical Report 87-06.
9. J.H. Davenport, Y. Siret, E. Tournier (1988). *Computer Algebra: Systems and Algorithms for Algebraic Computation*, London.
10. K.H. Ebert, P. Deuflhard, W. Jaeger (ed.) (1981). Modeling of chemical reaction systems. Springer Ser. Chem. Phys. 18.
11. J.C. Faugère, P. Gianni, D. Lazard, T. Mora (1989). *Efficient Computation of Zero-Dimensional Groebner Bases by Change of Ordering*, Preprint.
12. K. Gatermann (1990). *Symbolic Solution of Polynomial Equation Systems with Symmetry*, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Preprint SC 90-3.
13. R. Gebauer, H.M. Möller (1988). A new implementation of Buchberger's algorithm. *J. Symbolic Computation 6*, 275-286.

14. A.C. HEARN (1987). *REDUCE User's Manual, Version 3.3*, July. The RAND Corporation CP78, Santa Monica.
15. H. KREDEL (1988). Admissible termorderings used in computer algebra systems. *SIGSAM Bulletin, Vol. 22*, 1.
16. H. MELENK, H.M. MÖLLER, W. NEUN (1988). *On Groebner Bases Computation on a Supercomputer Using REDUCE*, Konrad-Zuse-Zentrum für Informationstechnik Berlin, SC 88-2.
17. H. MELENK, H.M. MÖLLER, W. NEUN (1989). Symbolic solution of large stationary chemical kinetics problems. *Impact of Computing in Science and Engineering 1*, 136-167.
18. H. MELENK, H.M. MÖLLER, W. NEUN (1990). *GROEBNER. A Package for Calculating Groebner Bases*, Version January 1990. Available from data base REDUCE-netlib at rand.org.
19. F. MORA, H.M. MÖLLER (1983). The computation of the Hilbert function. *Proceedings EUROCAM 1983*, Springer Lecture Notes in Comp. Sci., Vol. 162, 157-167.
20. V.W. NOONBURG. A neural network modeled by an adaptive Lotka-Volterra system. To appear in *SIAM J. Appl. Math.*
21. E.R. SPEER. Private communication.