Reliable Computing **1**, 41–64 (1995)

Reliable Computing
Institute for New Technologies
St. Petersburg, 1995

A COMBINED METHOD FOR ENCLOSING ALL SOLUTIONS OF NONLINEAR SYSTEMS OF POLYNOMIAL EQUATIONS

Christine Jäger and **Dietmar Ratz**

Abstract

We consider the problem of finding interval enclosures of all zeros of a nonlinear system of polynomial equations. We present a method which combines the method of Gröbner bases (used as a preprocessing step), some techniques from interval analysis, and a special version of the algorithm of E. Hansen for solving nonlinear equations in one variable. The latter is applied to a triangular form of the system of equations, which is generated by the preprocessing step. Our method is able to check if the given system has a finite number of zeros and to compute verified enclosures for all these zeros. Several test results demonstrate that our method is much faster than the application of Hansen's multidimensional algorithm (or similar methods) to the original nonlinear systems of polynomial equations.

1 Introduction

The general problem we address is:

Find, with certainty, all solutions in $\mathbb{R}^n$ of the nonlinear system

$$f_k(x_1, x_2, \ldots, x_n) = 0 \quad \text{for} \quad k = 1, \ldots, m$$

of m polynomials $f_k : \mathbb{R}^n \to \mathbb{R}$.

Successful approaches to the corresponding bound constrained problem (i.e. lower and upper bounds for the variables x_i are known) are interval Newton methods in conjunction with generalized bisection. These methods are also important tools for nonlinear optimization methods, since they can be used to compute all critical points of the objective function by applying the methods to its gradient. Such interval Newton methods are described for example in [1], [10], [12], [13], [16], [17].

In this paper, we present a method which combines the method of Gröbner bases, some techniques from interval analysis, and a special version of the algorithm of E. Hansen for solving nonlinear equations in one variable. The method is able to check if the given problem has a finite number of zeros and to compute verified enclosures of *all* these zeros in $\mathbb{R}^n$ *without* any constraints.

The idea of our combined method is to first generate a triangular form of the system of equations and to call Hansen's algorithm recursively for the reduced polynomials.

We first describe some details of the two algorithms combined in our method, and we discuss special techniques to overcome problems occurring due to the interval arithmetical modification of the method of Gröbner bases. We give a detailed overview of the steps of our algorithm, and we introduce some strategies to accelerate the computations.

Finally, we give some notes about the portable implementation in PASCAL–XSC, and we demonstrate by several test results that our method is much faster than the application of multi-dimensional interval Newton methods to the original nonlinear system of polynomial equations.

2 The Method of Gröbner Bases

Our method uses the method of Gröbner bases as a preprocessing step, so we first give some details of this method. To make understanding easier, we start with some remarks on polynomials. Afterwards, the essential parts of the theoretical background are explained. Beginning with the most simple form of the algorithm of Buchberger using polynomials with coefficients in $I\!R$, we mention some criteria from the literature in order to present a more comfortable and structured version of the algorithm. Then we develop the method for polynomials with interval coefficients and discuss some improvements.

2.1 Basic Properties of Polynomials and Polynomial Equations

The set of polynomials

$$\Big\{ f(x) = \sum a_p \cdot x^p \mid a_p \in I\!R, \quad p = 0, 1, 2, \ldots \Big\}$$

forms the ring $R[x]$ with $x \in I\!R$, [19]. We can also form a ring for the polynomials in several variables, the ring $R[x_1, x_2, \ldots, x_n] = R[x_1][x_2] \ldots [x_n]$ ($x_i \in I\!R$, $i = 1, \ldots, n$) with the elements

$$f(x_1, x_2, \ldots, x_n) = \sum a_{p_1 \ldots p_n} \cdot x_1^{p_1} \cdots x_n^{p_n},$$

where $a_{p_1 \ldots p_n} \in I\!R$ and $p_i = 0, 1, 2, \ldots$ for $i = 1, \ldots, n$.

Now, we interpret the polynomials (the left-hand sides of our polynomial system of equations) as a basis B (a system of generators), and we consider the set of all linear combinations (with polynomial coefficients) of these generators, which is called an *ideal* [8]. That is, our problem is specified by $B = (f_1, f_2, \ldots, f_m)$ and the equations $f_k = 0$ with $f_k \in R[x_1, x_2, \ldots, x_n]$ for $k = 1, \ldots, m$.

Two polynomials f and g are *equivalent* with respect to an ideal, if their difference belongs to the ideal. We can always add any linear combination of the generators of the ideal to the basis or we can discard one of the generators if it is a linear combination of the others. Now, our aim is to produce a "*simple*" basis.

The product of powers of variables $x_1, \ldots, x_n$ is called *monomial* (short form: MON). That is, a monomial is given by

$$x_1^{p_1} \cdots x_n^{p_n},$$

where $p_i \in \{0, 1, 2, \ldots\}$ for $i = 1, \ldots, n$. Each polynomial consists of such monomials. In representing a polynomial, we can order the monomials differently, and we can also order the variables inside a monomial differently. For example, we can write $x_1 + x_2 + x_3$ instead of $x_2 + x_3 + x_1$ or $x_1^2 x_2 + x_3^2 + x_2^2$ instead of $x_3^2 + x_2^2 + x_1^2 \cdot x_2$. Therefore, it would be nice to have a canonical representation. So, we have to fix some ordering that describes which monomials should be placed first in the canonical representation of a polynomial.

In the following, let $\succ$ be an order over the monomials that satisfies the following conditions:

- If $M_2 \succ M_1$, then for every monomial M_3, we have $M_2 M_3 \succ M_1 M_3$.
- For all monomials M_1 and M_2 with $M_2 \neq 1$, we have $M_1 M_2 \succ M_1$.

For our purpose, we choose a lexicographic order with respect to the names of the variables. That is we fix

$$x_1 \succ x_2 \succ x_3 \succ \ldots \succ x_n$$

and

$$x_1^{p_1} \cdots x_n^{p_n} \succ x_1^{q_1} \cdots x_n^{q_n} \Longleftrightarrow (p_1 > q_1) \vee ((p_i = q_i, \quad i = 1, \ldots, s) \wedge (p_{s+1} > q_{s+1}))$$

where $s < n$ (e.g. we have $x_1^5 \succ x_1 \cdot x_2^8$ and $x_1^2 \cdot x_2^2 \succ x_1^2 \cdot x_3^2$).

Given such an order, we are now able to transform a polynomial into an exactly determined form, where all monomials are written in decreasing order. We write

$$f = a_0 \cdot \mathrm{MON}_0 + \ldots + a_l \cdot \mathrm{MON}_l$$

where $\mathrm{MON}_0 \succ \mathrm{MON}_1 \succ \ldots \succ \mathrm{MON}_l$ and l is the number of terms in the polynomial f. We call the polynomial *normalized*, if $a_0 = 1$.

The monomial PMON := MON_0 we call the *principal monomial*, and PMON together with the corresponding coefficient a_0 we call the *principal term*. So, within an ideal basis B, f_k is given by

$$f_k = a_{k0} \cdot \mathrm{MON}_{k0} + \ldots + a_{kl} \cdot \mathrm{MON}_{kl}$$

or by

$$f_k = a_{k0} \cdot \mathrm{PMON}_k + a_{k1} \cdot \mathrm{MON}_{k1} + \ldots + a_{kl} \cdot \mathrm{MON}_{kl}$$

for $k = 1, \ldots, m$.

Example 2.1 *Using the order $x_1 \succ x_2 \succ x_3$, we have that*

$f_1 = 3 \cdot x_3 \cdot x_2 + x_1$ *is an invalid form,*
$f_2 = 5 \cdot x_1^2 + x_2 \cdot x_3 + 3 \cdot x_3 + 3$ *is a valid form, and*
$f_3 = x_1 \cdot x_2 + 2 \cdot x_2^2$ *is a valid normalized form.*

In the following, we assume the polynomials to be normalized, i.e. we always assume the coefficient of the principal monomial to be 1.

Definition 2.1 *A monomial* $\mathrm{MON}_p = x_1^{p_1} \cdots x_n^{p_n}$ *is called a multiple of the monomial* $\mathrm{MON}_q = x_1^{q_1} \cdots x_n^{q_n}$, *if* $p_i \geq q_i$ *for all* $i = 1, \ldots, n$. *We also say that* MON_q *divides* MON_p.

Example 2.2 $x_1^4 \cdot x_2^5 \cdot x_3^2$ *is a multiple of* $x_1^2 \cdot x_2^2 \cdot x_3^2$, *but* $x_1^4 \cdot x_2 \cdot x_3^6$ *is not a multiple of* $x_1^2 \cdot x_2^2 \cdot x_3^2$.

Definition 2.2 *The least common multiple (short form:* LCM*) of two monomials* $\mathrm{MON}_p = x_1^{p_1} \cdots x_n^{p_n}$ *and* $\mathrm{MON}_q = x_1^{q_1} \cdots x_n^{q_n}$ *is defined by*

$$\mathrm{LCM}(\mathrm{MON}_p, \mathrm{MON}_q) := x_1^{m_1} \cdots x_n^{m_n}$$

with $m_i := \max(p_i, q_i)$ *for all* $i = 1, \ldots, n$.

Example 2.3 $\mathrm{LCM}\ (x_1^4 \cdot x_2 \cdot x_3^6,\ x_1^2 \cdot x_2^2 \cdot x_3^2) = x_1^4 \cdot x_2^2 \cdot x_3^6$

2.2 Buchberger's Algorithm

2.2.1 Construction of S-polynomials and M-reductions

The reader should keep in mind that arithmetical operations within a given ideal of nonlinear polynomials are possible and lead back to elements of the same ideal. For this reason we are allowed to make the following definitions.

Definition 2.3 *Let* f_i *and* f_j *be two non-zero polynomials in normalized form, and let* PMON_i *and* PMON_j *be their principal monomials. Then the polynomial*

$$S(f_i, f_j) = \frac{\mathrm{LCM}(\mathrm{PMON}_i, \mathrm{PMON}_j)}{\mathrm{PMON}_i} \cdot f_i - \frac{\mathrm{LCM}(\mathrm{PMON}_i, \mathrm{PMON}_j)}{\mathrm{PMON}_j} \cdot f_j$$

is called S-polynomial of f_i *and* f_j.

Due to the multiplications of the two polynomials by the special factors and the succeeding subtraction, the resulting S-polynomial has a principal monomial that is less than the LCM of both principal monomials of the given polynomials f_i and f_j.

Example 2.4 *Let the polynomials* $f_i = x_1^2 \cdot x_2^2 + x_2^2$ *and* $f_j = x_1^3 + x_1 \cdot x_2^2$ *be given in the order* $x_1 \succ x_2$. *Then we get*

$$\mathrm{LCM}(\mathrm{PMON}_i, \mathrm{PMON}_j) = x_1^3 \cdot x_2^2$$

and

$$\begin{aligned} S(f_i, f_j) &= \frac{x_1^3 \cdot x_2^2}{x_1^2 \cdot x_2^2} \cdot (x_1^2 \cdot x_2^2 + x_2^2) - \frac{x_1^3 \cdot x_2^2}{x_1^3} \cdot (x_1^3 + x_1 \cdot x_2^2) \\ &= x_1 \cdot (x_1^2 \cdot x_2^2 + x_2^2) - x_2^2 \cdot (x_1^3 + x_1 \cdot x_2^2) \\ &= (x_1^3 \cdot x_2^2 + x_1 \cdot x_2^2) - (x_1^3 \cdot x_2^2 + x_1 \cdot x_2^4) \\ &= -x_1 \cdot x_2^4 + x_1 \cdot x_2^2. \end{aligned}$$

Remark: Obviously, $S(f,f)=0$ and $S(f,g)=-S(g,f)$.

A reduction of a polynomial can be done by a special form of an S-polynomial construction.

Definition 2.4 *Let f_i and f_j be two given polynomials in normalized form, and let* PMON_i *and* PMON_j *be their principal monomials, respectively. If* PMON_i *is a multiple of* PMON_j*, then we define a simplified version of an S-polynomial construction by*

$$M(f_i,f_j)=f_i-\frac{\mathrm{PMON}_i}{\mathrm{PMON}_j}\cdot f_j.$$

The replacement of f_i by $M(f_i,f_j)$ we call M-reduction of f_i, and we say "f_i was reduced by f_j". If $M(f_i,f_j)=0$ after performing the M-reduction, we call f_i reduced to zero.

Example 2.5 *Let the polynomials $f_i=x_1\cdot x_2^4-x_2^2$ and $f_j=x_1\cdot x_2-x_2$ be given in the order $x_1\succ x_2$. Then a single M-reduction is given by*

$$\begin{aligned}
f_i &:= (x_1\cdot x_2^4-x_2^2)-\frac{x_1\cdot x_2^4}{x_1\cdot x_2}\cdot(x_1\cdot x_2-x_2)\\
&= (x_1\cdot x_2^4-x_2^2)-x_2^3\cdot(x_1\cdot x_2-x_2)\\
&= (x_1\cdot x_2^4-x_2^2)-(x_1\cdot x_2^4-x_2^4)\\
&= x_2^4-x_2^2.
\end{aligned}$$

2.2.2 Theoretical Background

For a polynomial system of equations, i.e. a set G of polynomials generating an ideal, let us give some relevant definitions and theorems from [8].

Definition 2.5 *A polynomial f is reduced with respect to G, if no principal monomial of an element of G divides the principal monomial of f.*

Definition 2.6 *A system of generators (or a basis) G of an ideal I is called a* standard basis *or* Gröbner basis *(with respect to the order $\succ$), if every reduction of an f of I to a reduced polynomial (with respect to G) always gives zero.*

Theorem 2.1 *Every ideal has a Gröbner basis with respect to the lexicographic order.*

Theorem 2.2 *Two ideals are equal if and only if they have the same reduced standard basis with respect to the lexicographic order.*

These theorems can be generalized to other orders than the lexicographic one if the chosen order fulfills the criteria mentioned above. The proofs can be found in the publications of Buchberger ([2], [3], [4], [5], [6], [7]).

2.2.3 Notation

In the description of the algorithms, we use the following notations:

G	basis of an ideal (i.e. the given system of equations)
F	triangular form of the basis after termination of the algorithm
f_i, g_j	polynomials of F and G
$S(.,.)$	S-polynomial construction
B	set of all combinations of S-polynomial constructions to be performed
L_F	index of the last polynomial in F
L_G	index of the last polynomial in G
h	polynomial

We use indentation to mark compound statements and to avoid "begin – end" notation in some cases.

2.2.4 The Algorithm in $I\!R$

The algorithm of Buchberger is based on the theory of Gröbner bases and transforms an arbitrary polynomial system of equations into a triangular form. By triangular form in this context, we mean a system of polynomial equations

$$f_k(x_1, x_2, \ldots, x_n) = 0, \quad k = 1, \ldots, l,$$

where

$$\begin{array}{lcr} f_1 & = & f_1(x_1, x_2, \ldots, x_{l-1}, x_l, \ldots, x_n), \\ f_2 & = & f_2(x_2, \ldots, x_{l-1}, x_l, \ldots, x_n), \\ & \vdots & \\ f_{l-1} & = & f_{l-1}(x_{l-1}, x_l, \ldots, x_n), \\ f_l & = & f_l(x_l, \ldots, x_n). \end{array}$$

The transformation to triangular form is done by using constructions of S-polynomials and M-reductions. The triangular system then has the same zeros as the original system. For the final numerical part of our combined method, we are interested in a perfect triangular form where $l = n$.

Theorem 2.3 *A basis G is a standard basis if and only if, for every pair of polynomials f and g of G, $S(f, g)$ reduces to zero with respect to G.*

A basic version of the algorithm was given by Buchberger in [5].

$\boxed{\textit{Algorithm I:}}$

```
input a polynomial system of equations $G$

$F := G$;   $L_F := L_G$;   $B := \{(i,j) \mid 1 \le i < j \le L_G\}$;
while exists $(i,j) \in B$ do
begin
   $h := S(f_i, f_j)$;
   if $h \neq 0$ then
   begin
      $L_F := L_F + 1$;
      $B := B \cup \{(k, L_F) \mid 1 \le k < L_F\}$;
      $F := F \cup \{h\}$;                      { $h$ is added to the basis $F$ }
   end;
   $B := B - (i,j)$;
end;

output a Gröbner basis $F$ with ideal($F$) = ideal($G$)
```

Algorithm I performs the construction of S-polynomials for all possible combinations of polynomials of G (including those coming up during the computation of new polynomials). For each S-polynomial which is not equal to zero, the following steps have to be done:

- The new polynomial has to be added to the ideal.
- The set B of pairs of polynomials which still have to be used for a construction of an S-polynomial has to be updated, that means
  - (i,j) has to be removed as it was already performed, and
  - all new pairs that can be built from the new polynomial and the existing polynomials in F have to be added to B.

In [2], Buchberger gives some criteria for superfluous steps:

- If the LCM of the principal monomials of two polynomials is the product of the principal monomials, then their S-polynomial can always be reduced to zero. Therefore those combination can be left out.
- If the LCM of the principal monomials of two polynomials is equal to one of these principal monomials, then the corresponding polynomial can be removed from the ideal basis after the S-polynomial is built and all possible M-reductions are done.

The following algorithm incorporates these two criteria, and it gives a more structured form of the method of Gröbner bases.

$\boxed{\textit{Algorithm II:}}$

```
input a polynomial system of equations $G$

$F := G$;   $L_F := L_G$;   $i := 1$;
while $(i \le L_F - 1)$ do
begin
   $j := i + 1$;
   while $(j \le L_F)$ do
   begin
      if $\mathrm{LCM}(\mathrm{PMON}(f_i), \mathrm{PMON}(f_j)) \neq \mathrm{PMON}(f_i) \cdot \mathrm{PMON}(f_j)$ then
         begin
            $h := S(f_i, f_j)$;
            while (M-reduction of $h$ by $f_k$ from $F$ is possible) do
               $h := M(h, f_k)$;
            if ($h$ is not reduced to 0) then
               $F := F \cup \{h\}$;   $L_F := L_F + 1$;              { $h$ is added to $F$ }
            if $\mathrm{PMON}(f_j) = \mathrm{LCM}(\mathrm{PMON}(f_i), \mathrm{PMON}(f_j))$ then
               $F := F - f_j$;   $L_F := L_F - 1$;              { Remove $f_j$ from $F$ }
               Renumber the polynomials in $F$;
            else if $\mathrm{PMON}(f_i) = \mathrm{LCM}(\mathrm{PMON}(f_i), \mathrm{PMON}(f_j))$ then
               $F := F - f_i$;   $L_F := L_F - 1$;              { Remove $f_i$ from $F$ }
               Renumber the polynomials in $F$;
               $i := i - 1$;   $j := L_F + 1$;                  { Exit $j$-loop }
            else
               $j := j + 1$;
         end
      else
         $j := j + 1$;
   end
   $i := i + 1$;
end

output a Gröbner basis $F$ with $\mathrm{ideal}(F) = \mathrm{ideal}(G)$
```

In Algorithm II, similarities to the Gaussian algorithm are noticeable. In general, we can say it is a generalization of the Gaussian algorithm for systems of nonlinear polynomials. Therefore, there are two cases:

1. *Linear polynomials:* in this case, the algorithm is close to the Gaussian algorithm except for the order of the reductions. All constructions of S-polynomials are also M-reductions.

2. *Nonlinear polynomials:* in this case, the desired triangular form of the given system can only be achieved by adding new polynomials to the basis. With the help of the new polynomials, it is possible to perform the necessary reductions. This change in the nonlinear case is due to the fact that no limits (depending on the dimension of the system) for the occurring monomials can be given. Thus, not all monomials needed for the reductions are necessarily in the given system.

Example 2.6 *We apply Algorithm II to the system*

$$\begin{array}{rcrcrcrcl} f_1 &=& x_1^2 & & & - & x_3 &=& 0\\ f_2 &=& x_1^2 &-& x_2^2 & & &=& 0\\ f_3 &=& & & x_2 &-& x_3^2 &=& 0. \end{array}$$

For the first S-polynomial construction with LCM $= x_1^2$, *we obtain*

$$h_1 = S(f_1, f_2) = \frac{x_1^2}{x_1^2}(x_1^2 - x_3) - \frac{x_1^2}{x_1^2}(x_1^2 - x_2^2) = x_2^2 - x_3.$$

The M-reduction with f_3 *results in*

$$h_2 = M(h_1, f_3) = x_2^2 - x_3 - \frac{x_2^2}{x_2}(x_2 - x_3^2) = x_2 x_3^2 - x_3.$$

A further M-reduction with f_3 *gives*

$$h_3 = M(h_2, f_3) = x_2 x_3^2 - x_3 - \frac{x_2 x_3^2}{x_2}(x_2 - x_3^2) = x_3^4 - x_3,$$

and thus, h_3 *is added to the basis, and* f_2 *is removed. No more constructions of S-polynomials have to be done subsequently, and the resulting system is given by*

$$\begin{array}{rcrcrcrcl} f_1 &=& x_1^2 & & &-& x_3 &=& 0\\ f_2 &=& & & x_2 &-& x_3^2 &=& 0\\ f_3 &=& & & & & x_3^4 - x_3 &=& 0. \end{array}$$

Due to the simplifications, Algorithm II might result in a system that is not totally reduced to triangular form. We correct this "mistake" by starting the algorithm again and again as long as there are possible S-polynomial constructions and M-reductions that still alter the system. In most cases, about three calls of the algorithm are sufficient.

2.2.5 Termination and Solution Set Criteria

In [2], Buchberger proofs that Algorithm II terminates after a finite number of steps for any given ideal. He also gives two **criteria for the set of solutions** [8]:

1. *A system of polynomial equations is inconsistent (it cannot be satisfied, even if we add polynomial extensions) if and only if the corresponding standard basis contains a constant.*

2. *The system of polynomial equations has a finite number of solutions if and only if each variable appears alone (such as z^n) in one of the principal terms of the corresponding standard basis.*

We will use this second criterion in our combined method to check whether it makes sense to call our one-dimensional solver or not (see Algorithm IV).

2.3 An Interval Version of Buchberger's Algorithm

Applying infinite precision arithmetic, Algorithm II leads to sufficient results. It is also possible to implement the algorithm on a computer using rational numbers instead of real ones (that is what Computer-Algebra-Systems normally do) to guarantee exact mathematical operations on the machine if enough storage capacity is available. The price we must pay for the "exactness" is a great amount of computing time.

In a similar way as for linear systems of equations, we might implement Buchberger's algorithm with ordinary floating-point arithmetic. Additionally, this enables a very simple "communication" for Buchberger's method with other numerical procedures. Due to the fact that, in general, no exact computations are possible in floating-point arithmetic, we have to deal with the errors produced by rounding operations.

2.3.1 Some Aspects of Computer Arithmetic

In our general definition of S-polynomials and M-reductions, we used normalized polynomials. For this purpose, it was necessary to divide all polynomials by the real coefficient of the principal monomial. Using floating-point operations, even the normalization alone may produce rounding errors.

Example 2.7 *Consider the polynomial $p = 30x_1^2x_2x_3^2 + 10x_2x_3^2 + 3x_3$, the real coefficients of which are exactly representable on the computer. In order to normalize it, we must divide the principal coefficient of p by 30. On the computer, where our real coefficients of the polynomial p are represented with a finite number of mantissa digits we get the "normalized" $\frac{1}{30}p \approx x_1^2x_2x_3^2 + 0.333\ldots 3x_2x_3^2 + 0.1x_3$, which is only an approximation of the normalized p (in decimal arithmetic as well as in binary arithmetic).*

Thus, the normalized polynomial may only be an approximation of the original polynomial and the probability for the successive arithmetical operations leading to another error (for example a cancellation error) is close to 1. So, we decided to avoid every division during the algorithm and to give up the claim of normalized polynomials and rewrite our formulas for S-polynomials and M-reductions. We remark, that divisions by powers of the basis of the floating-point system are an exception to this rule due to the fact that they will not make harm to the mantissa. In our implementation, we use them to prevent the coefficients from overflow or underflow.

Further numerical difficulties arise, if the given input system contains coefficients like $\frac{1}{3}$ or just seemingly unsuspicious numbers like 0.1 which are *not* exactly representable on a binary floating-point system. In such cases, we may multiply the equations before entering them to prevent the conversion error during input. But for coefficients like $\sqrt{5}$ or $\sqrt{7}$, the multiplication will not help, rounding errors will occur anyway.

We developed an interval version of Buchberger's algorithm to *control* the rounding errors. That is, we use an algorithm which computes a triangular system with interval coefficients. This interval triangular system is an enclosure of the exact triangular system. So, when entering the system, the coefficients of the polynomials are enclosed in intervals of best possible accuracy, and all operations in our algorithm are performed in interval arithmetic.

Remark: On a computer, a guaranteed and optimal enclosure of the real triangular form of the polynomial system can be obtained by using an exact machine interval arithmetic with optimal outwardly-directed rounding (see [10], [14], and [15] for details).

2.3.2 Interval Versions of S-polynomial Construction and M-reduction

Now we develop an interval version of Buchberger's algorithm in order to get a guaranteed enclosure of the exact triangular system and later guaranteed enclosures of all zeros of the system. We use the same approach and ideas that were employed when transforming the Gaussian algorithm into an interval version (cf. [1], [16], and [17]). For an introduction to the underlying interval arithmetic, see [1], [10], or [17].

Due to the fact that the expression $[x] - [x]$ does not result in the value zero for an interval $[x]$ with positive diameter (e.g. $[1,2] - [1,2] = [-1,1]$), the algorithm will not produce the desired triangular form, if the original rules for construction of S-polynomials and M-reductions are performed simply with interval coefficient instead of real coefficients. The elimination of the principal monomial of the new polynomial would not take place in general using interval arithmetic.

Thus, we explicitly set the coefficient of the principal term of the new polynomial to zero, if necessary. We are allowed to alter the algorithm in such a way for the same reasons as in the Gaussian algorithm. The interval system is a set of real systems. Hence, each polynomial in $I\mathbb{R}$ represents a set of polynomials with real coefficients. But every construction of an S-polynomial and every M-reduction performed with two polynomials within the sets, leads to a new polynomial, where the LCM of the two principal monomials is eliminated. The inclusion isotonicity of interval arithmetic operations guarantees that these polynomials are still enclosed in the resulting set after the elimination of the principal term.

Now we use new rules for the construction of S-polynomials and for the M-reduction as explained in the following.

S-polynomial construction in $I\mathbb{R}$: Let two polynomials

$$f_i = [a_i] \cdot \mathrm{PMON}_i + \dots \quad \text{and} \quad f_j = [a_j] \cdot \mathrm{PMON}_j + \dots$$

be given. We first compute

$$f_S := A \cdot \frac{\mathrm{LCM}(\mathrm{PMON}_i, \mathrm{PMON}_j)}{\mathrm{PMON}_i} \cdot f_i + B \cdot \frac{\mathrm{LCM}(\mathrm{PMON}_i, \mathrm{PMON}_j)}{\mathrm{PMON}_j} \cdot f_j,$$

with

$$A = \begin{cases} 1 & \text{if } [a_i] = [a_j] \\ 1 & \text{if } [a_i] = -[a_j] \\ [a_j] & \text{otherwise} \end{cases}$$

and

$$B = \begin{cases} -1 & \text{if } [a_i] = [a_j] \\ 1 & \text{if } [a_i] = -[a_j] \\ -[a_i] & \text{otherwise} \end{cases},$$

where

$$f_S = [a_S] \cdot \text{PMON}_S + h_S.$$

Then we set

$$S(f_i, f_j) := \begin{cases} h_S & \text{if } \text{PMON}_S = \text{LCM}(\text{PMON}_i, \text{PMON}_j) \\ f_S & \text{otherwise} \end{cases}.$$

Example 2.8 *Let the polynomials $f_i := [2,3] \cdot x_1^2 + x_2^2$ and $f_j := [1,2] \cdot x_1 \cdot x_2 + x_3$ be given in the order $x_1 \succ x_2$. Then, we get $\text{LCM}(\text{PMON}_i, \text{PMON}_j) = x_1^2 \cdot x_2$ Thus, we compute*

$$\begin{aligned} f_S &:= [1,2] \cdot \frac{x_1^2 \cdot x_2}{x_1^2} \cdot ([2,3] \cdot x_1^2 + x_2^2) - [2,3] \cdot \frac{x_1^2 \cdot x_2}{x_1 \cdot x_2} \cdot ([1,2] \cdot x_1 \cdot x_2 + x_3) \\ &= [2,6] \cdot x_1^2 \cdot x_2 + [1,2] \cdot x_2^3 - [2,6] \cdot x_1^2 \cdot x_2 - [2,3] \cdot x_1 \cdot x_3 \\ &= [-4,4] \cdot x_1^2 \cdot x_2 - [2,3] \cdot x_1 \cdot x_3 + [1,2] \cdot x_2^3, \end{aligned}$$

and the principal term has to be eliminated. The result is

$$S(f_i, f_j) = -[2,3] \cdot x_1 \cdot x_3 + [1,2] \cdot x_2^3.$$

M-reduction in $I\!\!R$: Let two polynomials

$$f_i = [a_i] \cdot \text{PMON}_i + \ldots \quad \text{and} \quad f_j = [a_j] \cdot \text{PMON}_j + \ldots$$

be given. We first set $\text{PMON}_i^{old} := \text{PMON}_i$ and $f_i^{old} := f_i$, and we compute

$$f_i := A \cdot f_i^{old} + B \cdot \frac{\text{PMON}_i}{\text{PMON}_j} \cdot f_j$$

with

$$A = \begin{cases} 1 & \text{if } [a_i] = [a_j] \\ 1 & \text{if } [a_i] = -[a_j] \\ [a_j] & \text{else} \end{cases}$$

and

$$B = \begin{cases} -1 & \text{if } [a_i] = [a_j] \\ 1 & \text{if } [a_i] = -[a_j] \\ -[a_i] & \text{else} \end{cases},$$

where

$$f_i = [a_i] \cdot \text{PMON}_i + h_i.$$

Then, if $\text{PMON}_i^{old} = \text{PMON}_i$, we set

$$f_i := h_i.$$

Example 2.9 *Let the polynomials* $f_i := [-3,-3] \cdot x_1 \cdot x_2 + x_3$ *and* $f_j := [1,2] \cdot x_1 + x_2^2$ *be given in the order* $x_1 \succ x_2$*. Then we compute*

$$\begin{aligned} f_i &:= [1,2] \cdot ([-3,-2] \cdot x_1 \cdot x_2 + x_3) - [-3,-2] \cdot \frac{x_1 \cdot x_2}{x_1} \cdot ([1,2] \cdot x_1 + x_2^2) \\ &= [-6,-2] \cdot x_1 \cdot x_2 + [1,2] \cdot x_3 - [-6,-2] \cdot x_1 \cdot x_2 - [-3,-2] \cdot x_2^3 \\ &= [-4,4] \cdot x_1 \cdot x_2 + [-3,-2] \cdot x_2^3 + [1,2] \cdot x_3, \end{aligned}$$

and the principal term has to be eliminated. The result is

$$f_i = [-3,-2] \cdot x_2^3 + [1,2] \cdot x_3$$

The differences between the version of Buchberger's algorithm in $I\!R$ and the interval version are the constructions of the S-polynomials and the M-reductions. Moreover, the additional criteria for the set of solutions must be modified for the interval case. We cannot use the first criterion directly, because we must distinguish between constant interval polynomials which contain the value zero and constant interval polynomials which do not contain zero.

The latter can be treated in the same way as before, but those constant polynomials which contain zero, but are unequal to zero, do not lead to an elimination. We cannot decide whether this polynomial is reduced to zero or not.

Example 2.10 *Let the following system be given:*

$$\begin{array}{rcrcrcrcl} f_1 &=& \sqrt{7}x_1 &-& \sqrt{5}x_2 && &=& 0 \\ f_2 &=& &-& \sqrt{5}x_2 &-& \sqrt{3} &=& 0 \\ f_3 &=& &-& 3\sqrt{5}x_2 &-& 3\sqrt{3} &=& 0 \end{array}$$

This system has nearly triangular form. A construction of an S-polynomial for f_2 *and* f_3 *and succeeding M-reductions assuming exact arithmetic deliver*

$$S(f_2, f_3) = 0.$$

Furthermore, our algorithm removes f_3 *and the system has an exact solution with*

$$x_1 = -\frac{\sqrt{3}}{\sqrt{7}} \quad \text{and} \quad x_2 = -\frac{\sqrt{3}}{\sqrt{5}}.$$

If we apply interval arithmetic with four significant mantissa digits, we start with the polynomials

$$\begin{array}{rclcrcl} f_1 &=& [2.645, 2.646]x_1 &-& [2.236, 2.237]x_2 && \\ f_2 &=& &-& [2.236, 2.237]x_2 &-& [1.732, 1.733] \\ f_3 &=& &-& 3[2.236, 2.237]x_2 &-& 3[1.732, 1.733]. \end{array}$$

A construction of the S-polynomial $f_4 = S(f_2, f_3)$ *and succeeding M-reductions for* f_4 *deliver*

$$f_4 = [-0.015, 0.015].$$

Keeping in mind that this interval (as a constant polynomial) encloses zero, the equation $0 = 0$ *is enclosed, and we get a solution with*

$$x_1 = -[0.653, 0.657] \quad \text{and} \quad x_2 = -[0.774, 0.776]$$

enclosing the exact solution.

We have seen that there can be solvable systems, despite the validity of the first criterion in its original form. Consequently, we

- stop our algorithm, if the constant polynomial does *not* contain zero, because the system has no solution then, and we
- store the constant polynomial for later investigations if it contains zero, and we go on with the constant polynomial set to zero.

We give some further notes on the implementation of our interval version of Buchberger's algorithm at the end of the article.

2.4 Some Improvements

Practical experience and another criterion of Buchberger lead to further ideas for improvements of Algorithm II.

2.4.1 Change of Order

A great improvement in computing time can be achieved by using the optimal order for the variables in the given system. Practical experience shows that it is favorable to choose an order that puts variables with small powers and seldom appearance in front and variables that appear in many polynomials or that have large powers in the back.

Example 2.11 *For the order* $(x_1 \succ x_2 \succ x_3)$*, let the system*

$$\begin{array}{rcrcrcl} x_1 & + & 4x_2 & + & 3x_3 & = & 0 \\ -4x_1x_2^2 & - & x_1 & & & = & 0 \\ x_1^3 & & & & & = & 0 \end{array}$$

be given. If we change the order to $(x_3 \succ x_2 \succ x_1)$*, then we get the system*

$$\begin{array}{rcrcrcl} 3x_3 & + & 4x_2 & + & x_1 & = & 0 \\ & - & 4x_2^2x_1 & - & x_1 & = & 0 \\ & & & & x_1^3 & = & 0 \end{array}$$

For the first order, Algorithm II performs 7 S-polynomial constructions and 19 M-reductions. For the second order, only 1 S-polynomial construction and 1 M-reduction are necessary.

For increasing dimensions and powers, the differences between the orders can be even bigger. Our experiences led to some criteria for choosing the preferable order:

1. Number of polynomials which contain x_i
2. Largest power of each variable in the whole system
3. Number of summands in the whole system that contain x_i
4. Length of the largest polynomial containing x_i.

This heuristic cannot guarantee an improvement of computing time, but practical experiences show that it leads to a respectable improvement in many cases. The success depends also on the inner structures of the system.

2.4.2 Special Way of Pivoting

A special strategy for changing the order in which the polynomials are treated in Algorithm II, we call *pivoting*. There are two cases for the choice of polynomials during the algorithm:

- in search of polynomials for the next construction of an S-polynomial and
- in search of the next polynomial for another M-reduction.

We give our special way of pivoting that accelerated our algorithm for a large number of systems:

1. For S-polynomials, we prefer those combinations of polynomials which lead to the least LCM.
2. For M-reductions, we prefer "short" polynomials.

Example 2.12 *For the system*

$$\begin{aligned} 5x_1^9 - 6x_1^5x_2^2 + x_1x_2^4 + 2x_1x_3 &= 0 \\ -2x_1^6x_2 + 2x_1^2x_2^3 + 2x_2x_3 &= 0 \\ x_1^2 + x_2^2 - 0.265625 &= 0 \end{aligned}$$

the following table shows some comparisons between the number of S-polynomials and M-reductions performed when using different orders and pivoting.

Order	*Number of S-polynomials*		*Number of M-reductions*	
	without pivoting	*with pivoting*	*without pivoting*	*with pivoting*
(1,2,3)	*108*	*30*	*566*	*133*
(1,3,2)	*14*	*10*	*49*	*27*
(3,2,1)	*25*	*14*	*144*	*106*

Again we cannot give a guarantee for an improvement. The success of this method depends on the properties of the used polynomials. Additionally, we must modify our pivoting strategy if we want to avoid special combinations of interval coefficients resulting in intervals containing zero which possibly lead to a termination of our method.

In the current version of our algorithm, the strategies mentioned above are implemented. But those strategies are too static. To achieve a greater improvement, we should be able to influence the algorithm at the beginning and also during the computation. Then the order and the pivoting should also fit to each new situation as for example high powers or long polynomials upcoming. We are still working on further improvements.

3 Hansen's Algorithm for Nonlinear Equations in One Variable

Now we turn to the second part of our combined method, Hansen's method for finding *all* zeros of a nonlinear continuously differentiable function $f : I\!R \to I\!R$, which we will apply to a polynomial with interval coefficients. We combine this method with the theorem of Gershgorin [18] in order to get an algorithm independent of a starting interval entered by the user. Hansen's method is an extension of the *interval Newton method* which applies extended interval operations (see [10] and [12] for details).

3.1 Theoretical Background

We address the problem of finding all solutions of the one-dimensional equation

$$f(x) = 0$$

for a continuously differentiable function $f : I\!R \to I\!R$ and $x \in [x]$. The interval Newton method for solving this equation can easily be derived from the mean value form

$$f(m([x])) - f(x^*) = f'(\xi) \cdot (m([x]) - x^*),$$

where $x^*, \xi \in [x]$ and $m([x])$ denotes the midpoint of $[x]$. If we assume x^* to be a zero of f, we get

$$x^* = m([x]) - \frac{f(m([x]))}{f'(\xi)} \in \underbrace{m([x]) - \frac{f(m([x]))}{f'([x])}}_{=: N([x])}.$$

Hence, every zero of f in $[x]$ also lies in $N([x])$, and therefore in $N([x]) \cap [x]$. Using standard interval arithmetic, the interval Newton method starts with an interval $[x]^{(0)}$ satisfying $0 \notin f'([x]^{(0)})$ and iterates according to

$$[x]^{(k+1)} := [x]^{(k)} \cap N([x]^{(k)}), \quad k = 0, 1, 2, \ldots$$

The method cannot diverge due to the intersection. If the intersection is empty, we know that there is no root of f in $[x]^{(k)}$.

Using extended interval arithmetic, as defined in [10] and [12], we are able to treat the case $0 \in f'([x]^{(0)})$ that occurs, for example, if there are several zeros in the starting interval $[x]^{(0)}$. In this case, $N([x]^{(k)})$ is given by one or two extended intervals resulting from the interval division. Even though $N([x]^{(k)})$ is infinite, the intersection $[x]^{(k+1)} = N([x]^{(k)}) \cap [x]^{(k)}$ is finite and may be a single interval, the union of two intervals, or the empty set. Then, the next step of the interval Newton iteration must be applied to each of the resulting intervals. In this way it is possible to enclose *all* zeros of f in the starting interval $[x]^{(0)}$.

The following theorem summarizes the most important properties of the interval Newton method.

Theorem 3.1 *Let $f : D \subseteq I\!R \to I\!R$ be a continuously differentiable function, and let $[x] \in I I\!R$, $[x] \subseteq D$ be an interval. Then*

$$N([x]) := m([x]) - \frac{f(m([x]))}{f'([x])}$$

has the following properties:

1. *Every zero $x^* \in [x]$ of f satisfies $x^* \in N([x])$.*
2. *If $N([x]) \cap [x] = \emptyset$, then there exists no zero of f in $[x]$.*
3. *If $N([x]) \subseteq [x]$, then there exists a unique zero of f in $[x]$ and hence in $N([x])$.*

The proofs appear in [12], [16], and [17].

3.2 Algorithmic Description

In the following, we give a simplified version of our Hansen-like algorithm. We use the following notations:

f	nonlinear function of one variable
$[x]$	starting interval for the search
ε	desired relative diameter for the final intervals
$[y]$	vector (list) of intervals that still have to be examined
$[z_\infty]$	vector (pair) of two possibly infinite intervals
$[z_p]$	vector (pair) of two finite intervals
[Zero]	final vector (list) of all enclosures for zeros
N	final number of enclosures

Algorithm III:

```
input function $f$, starting interval $[x]$, desired accuracy $\varepsilon$

$N := 0$;   $i := 1$;   $[y]_1 := [\mathrm{x}]$;                       { Initializations }
repeat
  if $0 \notin f([y]_i)$ then
    $i := i - 1$
  else
    $c := m([y]_i)$;
    $[z_\infty] := c - f(c)/f'([y]_i)$;           { Extended interval Newton step }
    $[z_p] := [y]_i \cap [z_\infty]$;         { Intersection $[y]_i \cap [z_\infty] = [z_p]_1 \cup [z_p]_2$ }
    if $[z_p]_1 = [y]_i$ then
      $[z_p]_1 := [\underline{y}, c]$;   $[z_p]_2 := [c, \overline{y}]$;                { Bisection }
    $i := i - 1$;
    for $k := 1$ to $2$ do
    begin
      if $[z_p]_k = \emptyset$ then next$_k$;
      if $d_{\mathrm{rel}}([z_p]_k) < \varepsilon$ then               { Store enclosure of zero }
        begin
          if $0 \in f([z_p]_k)$ then
            $N := N + 1$;   $[\mathrm{Zero}]_N := [z_p]_k$;
        end
      else
        begin                                           { Store $[z_p]_k$ in $[y]$ }
          $i := i + 1$;   $[y]_i := [z_p]_k$;
        end;
    end;
until $i = 0$
Sort all intervals in list [Zero];
Eliminate multiple intervals enclosing the same unique zero;

output the vector of enclosures of all zeros [Zero] and the number of zeros $N$
```

The input for Algorithm III is a one-dimensional function, a starting interval, and a desired relative accuracy. The given starting interval is split into smaller intervals by the *Newton step* or by *bisection*. These intervals $[y]_i$ are stored as components of the interval $[y]$ if $0 \in f([y]_i)$ (so they may contain a zero) and if the relative diameter of $[y]_i$ is greater than the desired accuracy ε. Otherwise, if the relative diameter is small enough, the interval is stored in the list of candidates for zeros, named [Zero]. If $0 \notin f([y]_i)$, then $[y]_i$ is discarded.

The process terminates when the list of intervals $[y]$ is empty, i.e. all subintervals $[y]_i$ could either be discarded due to the condition $0 \notin f([y]_i)$ or because their relative diameter is less than ε and they are stored in the list of zeros [Zero]. For further descriptions see [10] and [11].

4 The Combined Method

Now we combine Buchberger's algorithm and Hansen's algorithm.

```
*Algorithm IV:*

input functions $f_1, \dots, f_m$ representing a polynomial system of equations $G$

step A Apply Algorithm II on $G$ resulting in triangular system $F$

check finite solvability of the System $F = (f_1, \dots, f_n)$ and exit if necessary

step B Set $k := n$ and call RecursiveHansen($k$) defined by

    Compute starting interval for $f_k$ in Algorithm III
    Apply Algorithm III on $f_k(x_k)$ to compute enclosures $[y_k]_1, [y_k]_2, \dots, [y_k]_{N_k}$
    if $k = 1$ then
        return
    else
        for $i := 1$ to $N_k$ do
            Replace $x_k$ by $[y_k]_i$ in polynomials $f_1, \dots, f_{k-1}$;
            Call RecursiveHansen($k - 1$);

output a list of enclosures for all zeros of the given system
```

Algorithm IV has some advantages profounded in the theory of the Gröbner bases and in the fact that the multi-dimensional problem is reduced to a sequence of one-dimensional problems. Between parts A and B of the algorithm, the criteria for the solvability can be applied. Thus, we are able to decide whether the system is not solvable at all and whether the system has an infinite number of solutions. In both cases, it makes no sense to start Algorithm III, and the computing can be terminated.

Moreover, we do not need to enter starting intervals for the computations. In fact, we use the theorem of Gershgorin [18] to compute an interval that contains all zeros of the polynomial in one variable (to be treated by Algorithm III).

5 Implementation

We developed a portable implementation of Algorithm IV in PASCAL–XSC [14]. The software package is divided in three independent parts:

1. *Arithmetical operators, functions, and procedures to handle the nonlinear polynomials and systems.*

   In this part, we implemented an arithmetic for nonlinear polynomials including addition, subtraction, multiplication, and division of polynomials (also with mixed operands, terms, and monomials). Due to the fact that nonlinear polynomials are not limited in their length, we made great efforts to control the storage space (garbage collection).

2. *Interval version of Buchberger's algorithm.*

   Algorithm II was implemented in its interval version including the improvements in ordering and pivoting.

3. *Special triangular version of Hansen's algorithm with a controlling environment.*

   The environment program prepares all the data necessary for the recursive calls of Hansen's algorithm in one variable (e.g. computing the starting intervals).

Each of these parts can also be applied on its own to an appropriate problem.

6 Numerical Examples

Now we give some examples to demonstrate the performance of our combined method. The listed computing times are those from a HP 9000/720 equipped with PASCAL–XSC Version 2.03. The times for Algorithm IV include the time for order changing and pivoting, the time for computing all starting intervals for Algorithm III, and the time for producing a file with the results. All results are given for a desired relative accuracy of $\varepsilon = 10^{-12}$.

We give the results of three different algorithms for comparison:

1. Our combined method (**BB+NLTSS**: BB = Buchberger's algorithm + NLTSS = Nonlinear Triangular System Solver)
2. The Hansen-like algorithm which is a part of the PXSCDEMO program of the PASCAL–XSC system (**NLSSDEMO**).
3. The nonlinear system solver as described in [10] (**TBNLSS**).

The starting intervals necessary for input in NLSSDEMO and TBNLSS are those computed by BB+NLTSS.

Example 6.1 *Compute the intersection points of two circles:*

$$\begin{array}{ccccccccccc} x_1^2 & - & 20x_1 & + & x_2^2 & - & 2x_2 & + & 100 & = 0 \\ x_1^2 & - & 22x_1 & + & x_2^2 & - & 2x_2 & + & 121 & = 0. \end{array}$$

The numerical results are

```
Zero No. 1 :
x[1] = [  1.050000000000000E+001,  1.050000000000000E+001]
x[2] = [  1.866025403784438E+000,  1.866025403784439E+000]

Zero No. 2 :
x[1] = [  1.050000000000000E+001,  1.050000000000000E+001]
x[2] = [  1.339745962155613E-001,  1.339745962155614E-001]

Statistic for Buchberger's algorithm:
Number of S-polynomials :  2
Number of M-reductions  :  1
```

If we compare the computing times for the three different methods we get

Method	*Time*
BB+NLTSS	*0:00:00,020*
NLSSDEMO	*0:00:02,580*
TBNLSS	*0:00:00,270*

Example 6.2 *Compute the intersection of three spheres:*

$$\begin{array}{ccccccccc} x_1^2 & - & 2x_1 & + & x_2^2 & + & x_3^2 & = 0 \\ x_1^2 & + & x_2^2 & + & x_3^2 & - & 2x_3 & = 0 \\ x_1^2 & + & x_2^2 & + & x_3^2 & - & 1 & = 0. \end{array}$$

The numerical results are

```
Zero No. 1 :
x[1] = [  5.000000000000000E-001,  5.000000000000000E-001]
x[2] = [  7.071067811865474E-001,  7.071067811865476E-001]
x[3] = [  5.000000000000000E-001,  5.000000000000000E-001]

Zero No. 2 :
x[1] = [  5.000000000000000E-001,  5.000000000000000E-001]
x[2] = [ -7.071067811865476E-001, -7.071067811865474E-001]
x[3] = [  5.000000000000000E-001,  5.000000000000000E-001]

Statistic for Buchberger's algorithm:
Number of S-polynomials :  2
Number of M-reductions  :  0
```

If we compare the computing times for the three different methods we get

Method	*Time*
BB+NLTSS	*0:00:00,020*
NLSSDEMO	*0:00:05,110*
TBNLSS	*0:00:00:880*

Example 6.3 *Brown's almost linear system (5-dimensional) is given by*

$$\begin{array}{cccccccccccc} 2x_1 & + & x_2 & + & x_3 & + & x_4 & + & x_5 & - & 6 & = 0 \\ x_1 & + & 2x_2 & + & x_3 & + & x_4 & + & x_5 & - & 6 & = 0 \\ x_1 & + & x_2 & + & 2x_3 & + & x_4 & + & x_5 & - & 6 & = 0 \\ x_1 & + & x_2 & + & x_3 & + & 2x_4 & + & x_5 & - & 6 & = 0 \\ x_1 & \cdot & x_2 & \cdot & x_3 & \cdot & x_4 & \cdot & x_5 & - & 1 & = 0. \end{array}$$

The numerical results are

```
Zero No. 1 :
x[1] = [    -5.790430884942E-001,    -5.790430884940E-001]
x[2] = [  -5.79043088494121E-001,  -5.79043088494113E-001]
x[3] = [  -5.79043088494121E-001,  -5.79043088494113E-001]
x[4] = [  -5.79043088494121E-001,  -5.79043088494113E-001]
x[5] = [   8.89521544247056E+000,   8.89521544247061E+000]

Zero No. 2 :
x[1] = [    9.1635458253384E-001,    9.1635458253386E-001]
x[2] = [   9.16354582533848E-001,   9.16354582533851E-001]
x[3] = [   9.16354582533848E-001,   9.16354582533851E-001]
x[4] = [   9.16354582533848E-001,   9.16354582533851E-001]
x[5] = [   1.41822708733075E+000,   1.41822708733076E+000]

Zero No. 3 :
x[1] = [   9.99999999999997E-001,   1.00000000000001E+000]
x[2] = [  9.999999999999997E-001,  1.000000000000001E+000]
x[3] = [  9.999999999999997E-001,  1.000000000000001E+000]
x[4] = [  9.999999999999997E-001,  1.000000000000001E+000]
x[5] = [  9.999999999999995E-001,  1.000000000000001E+000]

Statistic for Buchberger's algorithm:
Number of S-polynomials :  4
Number of M-reductions  :  24
```

If we compare the computing times for the three different methods we get

Method	*Time*
BB+NLTSS	*0:00:00,150*
NLSSDEMO	*> 1:00:00,000*
TBNLSS	*> 1:00:00,000*

Example 6.4 *A polynomial system of higher degree is:*

$$\begin{array}{rcrcrcrl} 5x_1^9 & - & 6x_1^5x_2^2 & + & x_1x_2^4 & + & 2x_1x_3 & = 0 \\ & & -2x_1^6x_2 & + & 2x_1^2x_2^3 & + & 2x_2x_3 & = 0 \\ & & x_1^2 & + & x_2^2 & - & 0.265625 & = 0. \end{array}$$

We skip the numerical results for the 12 solutions and list the statistics:

```
Statistic for Buchberger's algorithm:
Number of S-polynomials :  14
Number of M-reductions  :  106
```

If we compare the computing times for the three different methods we get

Method	*Time*
BB+NLTSS	*0:00:00,390*
NLSSDEMO	*0:03:19,210*
TBNLSS	*0:00:07,490*

Example 6.5 *Feigenbaum example (3-dimensional) is:*

$$\begin{array}{rcrcrl} -3.84x_1^2 & + & 3.84x_1 & - & x_2 & = 0 \\ -3.84x_2^2 & + & 3.84x_2 & - & x_3 & = 0 \\ -3.84x_3^2 & + & 3.84x_3 & - & x_1 & = 0. \end{array}$$

We skip the numerical results for the 8 solutions and list the statistics:

```
Statistic for Buchberger's algorithm:
Number of S-polynomials :  2
Number of M-reductions  :  8
```

If we compare the computing times for the three different methods we get

Method	*Time*
BB+NLTSS	*0:00:01,110*
NLSSDEMO	*0:00:11,450*
TBNLSS	*0:00:02,820*

Example 6.6 *A variant of Powell's singular function:*

$$\begin{array}{rcrcrl} x_1 & + & 10x_2 & & & = 0 \\ \sqrt{10}x_3 & - & \sqrt{10}x_4 & & & = 0 \\ x_2^2 & - & 4x_2x_3 & + & x_3^2 & = 0 \\ \sqrt{10}x_1^2 & - & 2\sqrt{10}x_1x_4 & + & \sqrt{10}x_4^2 & = 0 \end{array}$$

The numerical results are

```
Zero No. 1 :
x[1] = [  0.000000000000000E+000,  0.000000000000000E+000]
x[2] = [  0.000000000000000E+000,  0.000000000000000E+000]
x[3] = [  0.000000000000000E+000,  0.000000000000000E+000]
x[4] = [  0.000000000000000E+000,  0.000000000000000E+000]

Statistic for Buchberger's algorithm:
Number of S-polynomial :  5
Number of M-reductions :  21
```

If we compare the computing times for the three different methods we get

Method	*Time*
BB+NLTSS	*0:00:00,050*
NLSSDEMO	*0:08:33,370*
TBNLSS	*0:00:53,580*

7 Summary and Future Work

In general, the problem of solving a system of polynomial equations is NP-hard (cf. [9]), and therefore, there is no chance to get a *fast* algorithm that would always find all the solutions. For *unbounded* systems, the situation is even worse: there exists an example ($x_1 = 2$, $x_2 = x_1^2, \ldots, x_{n+1} = x_n^2$) of a system whose only solution is hyperexponential and therefore, not computable in any reasonable time because simply to produce the result digit after digit will take too long.

We have been studying a combined method for finding interval enclosures of all zeros of a nonlinear system of polynomial equations. Significant improvement can be obtained in interval Newton methods if this method is used instead of a usual interval Newton method. The reason for this is that the preprocessing step based on an interval version of Buchberger's Algorithm reduces the original problem to a sequence of simpler problems, i.e. to a special triangular system of polynomial equations which can be solved by recursively calling a special Hansen-like one-dimensional solver. We have shown with our examples, that our method is in many cases much faster than the application of Hansen's multi-dimensional algorithm (or similar methods) to the original nonlinear systems of polynomial equations. Even for the hyperexponential example above for dimensions up to $n = 10$, our solver is able to compute the solution in less than one second.

There are, of course, examples where the combined method fails because the degrees of the polynomials generated by Algorithm II are too high and demand to much storage capacity for executing Algorithm III. Additionally, there are cases where coefficients of some monomials cannot be eliminated in interval arithmetic (due to inflation effects described in Section 2.3.2) although they would be zero assuming infinite precision arithmetic. Here, we have to deal with similar problems as in the interval Gauss algorithm.

Possible future work will investigate the application of special subtraction techniques (cf. [13]) for handling "equal" coefficients. Significant work can also be done to develop a strategy for deciding when to use the combined method and when to use a usual method. Furthermore, future research should include application of the combined method to the specific problems of computer graphics (e.g. ray tracing).

References

[1] Alefeld, G., Herzberger, J.: *Introduction to Interval Computations.* Academic Press, New York, 1983.

[2] Buchberger, B.: *Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems.* Aequationes Mathematicae **4**, 374–383, 1970.

[3] Buchberger, B.: *Theoretical Basis for the Reduction of Polynomials to Canonical Forms.* SIGSAM Bulletin **39**, 19–29, 1976.

[4] Buchberger, B.: *Some Properties of Gröbner Bases for Polynomial Ideals.* SIGSAM Bulletin **40**, 19–24, 1976.

[5] Buchberger, B.: *A Criterion for Detecting Unnecessary Reductions in the Construction of Gröbner Bases*. Proceedings of the 1979 European Symposium on Symbolic and Algebraic Computation. Lecture Notes in Computer Science **72**, Springer-Verlag, Berlin, Heidelberg, 3–21, 1979.

[6] Buchberger, B.: *H–Bases and Gröbner Bases for Polynomial Ideals*. CAMP–Linz publication 81–2.0, University of Linz, 1981.

[7] Buchberger, B.: *A Survey on the Method of Gröbner Bases for Solving Problems in Connection with Systems of Multivariate Polynomials*. Proceedings of the 2nd RIKEN Symposium Symbolic and Algebraic Computation (ed. N. Inada & T. Soma), World Scientific Publ., 69–83, 1985.

[8] Davenport, J. H., Siret, Y., Tournier, E.: *Computer Algebra, Systems and Algorithms for Algebraic Computation*. Academic Press, New York, 1988.

[9] Gaganov, A. A.: *Computational complexity of the range of the polynomial in several variables*. Cybernetics, 418–421, 1985.

[10] Hammer, R., Hocks, M., Kulisch, U., Ratz, D.: *Numerical Toolbox for Verified Computing I – Basic Numerical Problems*. Springer-Verlag, Heidelberg, 1993.

[11] Hansen, E.: *A Globally Convergent Interval Method for Computing and Bounding Real Roots*. BIT **18**, 415–424, 1978.

[12] Hansen, E.: *Global Optimization Using Interval Analysis*. Marcel Dekker, New York, 1992.

[13] Kearfott, R. B., Hu, C., Novoa, M.: *A Review of Preconditioners for the Interval Gauss-Seidel Method*. Interval Computations **1**, 59–85, 1991.

[14] Klatte, R., Kulisch, U., Neaga, M., Ratz, D., Ullrich, Ch.: *PASCAL–XSC — Language Reference with Examples*. Springer-Verlag, New York, 1992.

[15] Kulisch, U., Miranker, W. L.: *Computer Arithmetic in Theory and Practice*. Academic Press, New York, 1981.

[16] Moore, R. E.: *Interval Analysis*. Prentice-Hall, Englewood Cliffs, 1966.

[17] Neumaier, A.: *Interval Methods for Systems of Equations*. Cambridge University Press, Cambridge, 1990.

[18] Stoer, J., Bulirsch, R.: *Introduction to Numerical Analysis*. Springer-Verlag, New York, 1980.

[19] van der Waerden, B. L.: *Modern Algebra*. Frederick Ungar, New York, 1949.

Institut für Angewandte Mathematik
Universität Karlsruhe
D-76128 Karlsruhe