Real Root Isolation of Regular Chains

François Boulier[1], Changbo Chen[2], François Lemaire[1], and Marc Moreno Maza[2]

[1] LIFL, Université de Lille 1
59655 Villeneuve d'Ascq Cedex, France
{Francois.Boulier,Francois.Lemaire}@lifl.fr

[2] ORCCA, University of Western Ontario (UWO)
London, Ontario, Canada
{cchen252,moreno}@csd.uwo.ca

Abstract

We present an algorithm RealRootIsolate for isolating the real roots of a system of multivariate polynomials given by a zerodimensional squarefree regular chain. The output of the algorithm is guaranteed in the sense that all real roots are obtained and are described by boxes of arbitrary precision. Real roots are encoded with a hybrid representation, combining a symbolic object, namely a regular chain, and a numerical approximation given by intervals. Our isolation algorithm is a generalization, for regular chains, of the algorithm proposed by Collins and Akritas. We have implemented RealRootIsolate as a command of the module SemiAlgebraicSetTools of the RegularChains library in Maple. Benchmarks are reported.

1 Introduction

Finding real roots for univariate polynomials has been widely studied. Some methods guarantee the number of real roots and isolate each real root in an arbitrary small interval. The algorithm presented in this paper is a generalization to regular chains of the algorithm given by Collins and Akritas [8], whose termination proof is based on a theorem due to Vincent [32] and [24, Theorem of 2 circles].

There exist many different approaches for isolating real roots of univariate polynomials by means of Descartes rules of signs [13]. Uspensky [31] rediscovered independently* an inefficient version of Vincent's work [1]. More recent algorithms are closer to the original work of Vincent and based on continuous fractions [2, 3]. The approach of [29] is very efficient in memory since it avoids the storage of one polynomial at each node of the tree of the recursive calls. Observe that the application of this idea to our context should be possible. It is left to future work.

The methods mentioned above are all for univariate polynomials with integral or rational coefficients. In [14], the authors apply Descartes Algorithm for polynomials with bit-stream coefficients. In [9, 16], the authors present algorithms for isolating the real roots of univariate polynomials with real algebraic number coefficients.

There exist different approaches for isolating real roots of polynomial systems with finitely many complex solutions. Various constructions are employed to generalize to multivariate systems the techniques known for univariate equations: rational univariate representation [27], polyhedron algebra [23], and triangular decompositions [7, 20, 25, 35].

In this paper, we generalize the Vincent-Collins-Akritas Algorithm to zerodimensional squarefree regular chains; therefore our work falls in the same category as this latter group

*Recent investigations of A. Akritas seem to prove that Uspensky only had an incomplete knowledge of Vincent's paper, from [30, pages 363-368].

of papers. Our idea is to build inductively (one variable after another) "boxes" in which one and only one real solution lies. This basically amounts to applying the Vincent-Collins-Akritas Algorithm to polynomials with real algebraic coefficients defined by a regular chain. Our main algorithm RealRootIsolate takes a zerodimensional squarefree regular chain T as an input and returns a list of disjoint boxes (Cartesian products of intervals) such that each box contains exactly one real root of T (as a byproduct, RealRootIsolate counts the number of real roots). We have implemented our algorithm in MAPLE in the module `SemiAlgebraicSetTools` of the `RegularChains` library.

Although rediscovered independently, the techniques presented here share some ideas with those of [25, 26]. However, our algorithm focuses on finding isolation boxes for real solutions of polynomial systems whereas Rioboo's primary goal is to implement the real closure of an ordered field. Moreover, Rioboo relies on Sturm sequences and subresultants for univariate polynomial real root isolation.

The other real root isolation algorithms based on triangular decompositions, namely those reported in [7, 20, 35], rely on the so-called "sleeve polynomials", see Section 2.5.

We do not report on a comparative implementation with the methods in [7, 9, 20, 25, 35]. In order to ensure a fair comparison (similar to what was done in [4] for four triangular decomposition methods) one would need to bring these six real root isolation methods (including ours) in a common implementation framework. This would require to overcome several substantial difficulties. For instance, the method in [25] is available in AXIOM and Aldor only, and relies heavily on the features of those languages. In addition, the possible generalization of the work in [9] from simple algebraic extensions to multiple algebraic extensions would also require adjustment for those multiple extensions to be presented by regular chains that do not necessarily generate a maximal ideal. For all these reasons, we leave this comparative implementation for future work.

As mentioned, the algorithm presented here has been implemented in MAPLE interpreted code. However, it does not rely yet on fast polynomial arithmetic nor modular methods for regular chain computations. Following [18], these techniques should speed-up our implementation dramatically.

We compare our code with another real root isolation tool available in MAPLE: the RootFinding[Isolate] command based on the rational univariate representation [27]. With no surprise, the highly optimized supporting C code allows RootFinding[Isolate] to outperform our modest MAPLE implementation on systems that are in Shape Lemma position [5]. However, for different families of examples, corresponding to non-equiprojectable[†] varieties the situation is reversed which demonstrates the great interest of our approach, even in this unfair comparative implementation framework.

Another contribution of our work is that it equips MAPLE with a tool for manipulating real numbers exactly. For instance, our code provides a data-type (called a *box*) for encoding a point with n coordinates that are real algebraic numbers, together with a function for deciding whether this point cancels a given n-variate polynomial.

Our encoding of such points uses a hybrid representation, combining a symbolic object, namely a regular chain, and a numerical approximation given by intervals. We investigate the impact of different strategies on this hybrid arithmetic. In particular, we identify a family of examples where the use of normalized regular chains instead of arbitrary (but still zero-dimensional) regular chains can speed-up the root isolation even though normalization tends to substantially increase coefficient sizes, as established in [11].

[†]The notions of an equiprojectable variety and equiprojectable decomposition are discussed in [10].

2 Real root isolation of a zerodimensional regular chain

After recalling the Vincent-Collins-Akritas algorithm in Section 2.1 and introducing definitions in Section 2.2 and Section 2.3, the algorithm RealRootIsolate and its subalgorithms are presented in Section 2.4. In Section 2.5 we compare our method with other existing approaches.

2.1 The Vincent-Collins-Akritas algorithm

The Vincent-Collins-Akritas algorithm isolates the real roots of a squarefree polynomial (with rational coefficients) with an arbitrary precision. A basic version (Algorithm 1) is recalled here, as a first step to its generalization in Section 2.4.

Definition 1 *Let V be a finite set of t real numbers. An* interval decomposition *of V is a list $I_1, \ldots, I_t$ of intervals such that each interval I_i is an open rational interval $]a, b[$ or a rational singleton $\{a\}$, each interval I_i contains one element of V and $I_i \cap I_j = \emptyset$ when $i \neq j$.*

Algorithm 1 RootIsolateVCA(p)

Input: p squarefree polynomial of $\mathbb{Q}[x]$ with t real roots
Output: an interval decomposition of the real roots of p

```
1: H ← a strict bound on the roots of p
2: return RootIsolateAuxVCA(p, ]−H, H[)
```

In Algorithm 1, there are different ways to compute a strict bound H (in the sense that any root α of p satisfies $|\alpha| < H$). For example, if $p = \sum_{i=0}^{d} a_i x^i$, take the Cauchy bound $H = \frac{1}{|a_d|} \sum_{i=0}^{d} |a_i|$. Sharper bounds are given in [2].

Algorithm 2 RootIsolateAuxVCA$(p,]a, b[)$

Input: p squarefree polynomial in $\mathbb{Q}[x]$ and $a < b$ rational
Output: an interval decomposition of the real roots of p which lie in $]a, b[$

```
1: nsv ← BoundNumberRootsVCA(p, ]a, b[)
2: if nsv = 0 then
3:    return ∅
4: else if nsv = 1 then
5:    return ]a, b[
6: else
7:    m ← (a + b)/2     res ← ∅
8:    if p(m) = 0 then  res ← {{m}} end if
9:    {Next line ensures the roots are sorted increasingly}
10:   return RootIsolateAuxVCA(p, ]a, m[) ∪ res ∪
                 RootIsolateAuxVCA(p, ]m, b[)
11: end if
```

The main arguments for the correctness of Algorithm 1 are the following. Algorithm 3 computes a polynomial $\bar{p}$ whose positive real roots are in bijection with the real roots of p which lie in $]a, b[$. The application of Descartes' rule of signs on $\bar{p}$ thus provides a bound on

Algorithm 3 BoundNumberRootsVCA$(p,]a, b[)$

Input: $p \in \mathbb{Q}[x]$ and $a < b$ rational

Output: a bound on the number of roots of p in the interval $]a, b[$

1: $\bar{p} \leftarrow (x+1)^d \, p\left(\frac{a\,x+b}{x+1}\right)$ where d is the degree of p in x

2: denote $\bar{p} = \sum_{i=0}^{d} a_i x^i$

3: $a'_e, \ldots, a'_0 \leftarrow$ the sequence obtained from $a_d, \ldots, a_0$ by removing zero coefficients

4: **return** the number of sign variations in the sequence $a'_e, \ldots, a'_0$

the number of real roots of p which lie in $]a, b[$. This bound is exact when equal to 0 or 1 [24, Theorem 1.2]. Since p is squarefree, the bound returned by Algorithm 3 will eventually become 0 or 1, by [24, Theorem 2.5] so that the whole method terminates.

2.2 Regular chains

In this paper one only considers zerodimensional squarefree regular chains, abbreviated by *zs-rc*. Roughly speaking, a zerodimensional regular chain is a triangular set[‡] of polynomials, with as many equations as variables, and which has a finite number of complex roots (and consequently a finite number of real roots).

Zerodimensional regular chains are easier to understand and define than general regular chains. Let $x_1 < \cdots < x_s$ be s variables. Let $p \in \mathbb{Q}[x_1, \ldots, x_s]$ be a non-constant polynomial. We denote by $\mathrm{mvar}(p)$ the *main variable* (or largest variable) of p, by $\mathrm{init}(p)$ the *initial* (or leading coefficient w.r.t. $\mathrm{mvar}(p)$) of p, by $\mathrm{mdeg}(p)$ the degree of p in its main variable and by $\mathrm{sep}(p)$ the *separant* of p, that is $\partial p / \partial \mathrm{mvar}(p)$. If T is a set of polynomials in $\mathbb{Q}[x_1, \ldots, x_s]$, $\langle T \rangle$ denotes the ideal generated by T and $V(T)$ denotes the set of all complex solutions of the system $T = 0$. For a given x_i, $T_{\leq x_i}$ (resp. $T_{>x_i}$) denotes the elements of T whose main variable is less (resp. strictly greater) than x_i.

Definition 2 *Let $T = \{p_1, \ldots, p_s\}$ where each p_i lies in $\mathbb{Q}[x_1, \ldots, x_s]$. The set T is a* zerodimensional squarefree regular chain *(or* zs-rc*) of $\mathbb{Q}[x_1, \ldots, x_s]$ if*

- $\mathrm{mvar}(p_i) = x_i$ *for* $1 \leq i \leq s$
- $\mathrm{init}(p_i)$ *does not vanish on* $V(\{p_1, \ldots, p_{i-1}\})$ *for any* $2 \leq i \leq s$
- $\mathrm{sep}(p_i)$ *does not vanish on* $V(\{p_1, \ldots, p_i\})$ *for any* $1 \leq i \leq s$

Thanks to the first two conditions, it is easy to show that the system $T = 0$ has a finite number of complex solutions, which is equal to the product of the main degrees of the elements of T denoted $\textsc{Deg}(T)$. Moreover those solutions can be computed "incrementally" using the following solving scheme. The number of complex roots of the univariate polynomial p_1 is equal to its degree. For each root x_1^0 of p_1, consider the polynomial $p_2(x_1^0, x_2)$ which is univariate in x_2. This polynomial has the same degree in x_2 as p_2 since the initial of p_2 does not vanish on the solutions of $p_1 = 0$. Thus, the number of complex roots of $p_2(x_1^0, x_2)$ is equal to the degree of p_2. Proceeding on the remaining variables, one concludes that the number of complex solutions of $T = 0$ is equal to $\textsc{Deg}(T)$.

[‡]triangular set in the sense that each polynomial introduces exactly one more variable

The third condition, which forbids multiple roots, is the natural generalization of squarefree polynomials to regular chains. As for the algorithm RootIsolateVCA, this condition is only required to make the isolation algorithms terminate.

In practice, the zs-rc can be computed using the Triangularize algorithm [22] available in the `RegularChains` library shipped with MAPLE.

Moreover, the regular chains are not built by checking the conditions of Definition 2 but by using regularity tests of polynomials modulo ideals. A polynomial p is said to be *regular* modulo an ideal I if it is neither zero, nor a zero-divisor modulo I. If T is a regular chain, p is said to be regular modulo the regular chain T if p is regular modulo $\langle T \rangle$. Thus, the following definition is equivalent to Definition 2.

Definition 3 *Let $T = \{p_1, \ldots, p_s\}$ where each p_i lies in $\mathbb{Q}[x_1, \ldots, x_s]$. The set T is a* zerodimensional squarefree regular chain *(or* zs-rc*) of $\mathbb{Q}[x_1, \ldots, x_s]$ if*

- $\mathrm{mvar}(p_i) = x_i$ *for any* $1 \leq i \leq s$
- $\mathrm{init}(p_i)$ *is regular modulo the ideal* $\langle p_1, \ldots, p_{i-1} \rangle$ *for any* $2 \leq i \leq s$
- $\mathrm{sep}(p_i)$ *is regular modulo the ideal* $\langle p_1, \ldots, p_i \rangle$ *for any* $1 \leq i \leq s$

The expert reader has probably noticed that saturated ideals are not mentioned. Indeed, this precision in not necessary in dimension zero. The next lemma makes the link between the regularity property of a polynomial q modulo a zs-rc and the fact that q does not vanish on the solutions of a zs-rc. It is implicitly used to check whether or not a polynomial vanishes on a root of a regular chain in the CheckZeroDivisor algorithm.

Lemma 1 *Let T be a zs-rc of $\mathbb{Q}[x_1, \ldots, x_s]$ and q a polynomial of $\mathbb{Q}[x_1, \ldots, x_s]$. Then q is regular modulo T iff q does not vanish on any complex solution of T.*

2.3 Boxes

This section defines the boxes used for isolating solutions of zs-rc, as well as extra definitions needed to specify the algorithms of Section 2.4.

Definition 4 *An s-box (or box) B is a Cartesian product $B = I_1 \times \cdots \times I_s$ where each I_i is either a rational open interval $]a, b[$ (a and b are rational) or a singleton $\{a\}$ with a rational. The width of B, denoted by $|B|$, is defined as the maximum of the $|I_i|$ where $|I_i| = 0$ if it is a singleton and $b - a$ if $I_i =]a, b[$.*

Algorithm 4 EvalBox(p, B)

Input: $p \in \mathbb{Q}[x_1, \ldots, x_s]$ and B is a s-box
Output: a rational interval I such that $p(v) \in I$ for any $v \in B$

Different variants for EvalBox(p, B) exist. A simple version of this algorithm consists in using the three basic operations $(\times, -, +)$ defined in interval arithmetics after "expanding" p into a sum of monomials.

Any variant for EvalBox(p, B) satisfying the following property can be used: the box EvalBox(p, B) should tend to the singleton $\{p(x_0)\}$ when the width of B tend to zero (by keeping the condition $x_0 \in B$). This simply ensures that the interval EvalBox(p, B) should shrink as the width of the box B decreases.

Definition 5 *Let $B = I_1 \times \cdots \times I_s$ be an s-box and $T = \{p_1, \ldots, p_s\}$ be a zs-rc of $\mathbb{Q}[x_1, \ldots, x_s]$. We say (B, T) satisfies the Dichotomy Condition (or **DC**) if*

- *one and only one real root of T lies in B*
- *if $I_1 =]a, b[$ then $p_1(x_1 = a)$ and $p_1(x_1 = b)$ are nonzero and have opposite signs*
- *for each $2 \leq k \leq s$ if $I_k =]a, b[$ then the two intervals $\mathsf{EvalBox}(p_k(x_k = a), B)$ and $\mathsf{EvalBox}(p_k(x_k = b), B)$ do not meet 0 and have opposite signs.*[§]

This last condition is the natural generalization of the condition $p(a)$ and $p(b)$ are nonzero and have opposite sign, and p vanishes only once on the interval $]a, b[$ in the univariate case. Condition **DC** allows to refine a box very much like one refines the interval $]a, b[$ by dichotomy.

Please note that condition **DC** does not require anything when an I_k is a singleton. In fact, if $I_1 = a$ is a singleton, then one necessarily has $p_1(a) = 0$ since one real root of T lies in B. Equivalently, if $I_k = a$ is a singleton for some k, then $p(x_k = a)$ vanishes on the real root of T lying in B.

Definition 6 *Let V be a finite set of t points of $\mathbb{R}^s$. A list $B_1, \ldots, B_t$ of s-boxes is called a* box-decomposition *of V if each point of V lies in exactly one B_i and if $B_i \cap B_j = \emptyset$ when $i \neq j$. If T is a zs-rc, we call box-decomposition of T a box-decomposition of the real roots of $T = 0$.*

Definition 7 *A task $\mathcal{M} = \text{TASK}(p,]a, b[, B, T)$ is defined as: T is a zs-rc of $\mathbb{Q}[x_1, \ldots, x_s]$, p is a polynomial in $\mathbb{Q}[x_1, \ldots, x_{s+1}]$, $T \cup \{p\}$ is a zs-rc, B is an s-box, (B, T) satisfies **DC**, and $a < b$ are rational numbers. The solution of $\mathcal{M}$ denoted by $V_t(\mathcal{M})$ is defined as $V(T \cup \{p\}) \cap (B \times]a, b[)$ (i.e. the real solutions of $T \cup \{p\}$ which prolong the real root in B and whose component x_{s+1} lies in $]a, b[$).*

2.4 Algorithms

The main algorithm RealRootIsolate, which isolates the real roots of a zerodimensional squarefree regular chain, is presented here. Only elements of proofs are given but focus has been made on specifications. In this section, one assumes $n > 1$.

The algorithms presented here use the mechanism of exceptions which is available in a lot of programming languages (Ada, C++, Common Lisp, Java and Maple). We find it really appropriate in our case for the following reason. Doing computations using the D5 principle [12] can be seen as doing computations as if one is computing over a field. When a zero divisor is hit (leading to a splitting), one raises an exception exhibiting the splitting. This exception can then be caught to restart computations. This shortens and makes clearer[¶] the algorithms presented here, although the reader needs to be familiar with exceptions. Algorithm 5 is the only one which directly throws exceptions.

Algorithm 5 checks whether p is regular modulo T or not. If p is regular modulo T, the algorithm returns normally, otherwise an exception is raised. Algorithm 5 is called whenever one wants to know if a polynomial vanishes on a real root x^0 of T isolated by a box B. Indeed, if p is regular modulo T, thanks to Lemma 1, p does not vanish on x^0. This

[§] the sign of an interval not meeting zero is just the sign of any element of it

[¶] without using exceptions, splitting would have to be handled basically each time a function returns a value

Algorithm 5 CheckZeroDivisor(p, T)

Input: T a zs-rc $\mathbb{Q}[x_1, \ldots, x_s]$ and $p \in \mathbb{Q}[x_1, \ldots, x_s]$
Output: one of the two cases happens:

- p is regular modulo T and it terminates normally
- p is not regular modulo T and an exception is thrown exhibiting t zs-rc $T_1, \ldots, T_t$ such that

  C1 $V(T_1) \cup \cdots \cup V(T_t) = V(T)$

  C2 $\sum_{i=1}^{t} \text{DEG}(T_i) = \text{DEG}(T)$

1: $T_1, \ldots, T_t \leftarrow$ Regularize(p, T)
2: **if** p belongs to at least one $\langle T_i \rangle$ **then**
3: **throw exception**$(T_1, \ldots, T_t)$
4: **end if**

allows to refine the box B until EvalBox(p, B) does not contain 0, which gives us the sign of $p(x^0)$.

The algorithm Regularize is not recalled here (see [22] for details) but its specification is: if T is a zs-rc, Regularize(p, T) returns a list of zs-rc $T_1, \ldots, T_t$ such that for each T_i, p either belongs to $\langle T_i \rangle$ or is regular modulo T_i. Moreover $T_1, \ldots, T_t$ is (what we call) a splitting of T, which in dimension 0 satisfies the two conditions **C1** and **C2** of the output of Algorithm 5. Due to condition **C2**, splittings cannot occur indefinitely.

Algorithm 6 RefineBox(B, T)

Input: T is a zs-rc of $\mathbb{Q}[x_1, \ldots, x_s]$, (B, T) satisfies **DC** and $|B| > 0$
Output: an s-box $\bar{B}$ such that $|\bar{B}| \leq |B|/2$, $\bar{B} \subset B$ and $(\bar{B}, T)$ satisfies the **DC**

Algorithm 6 is able to refine a box containing a real root by dividing its width by 2. It is simply the generalization of the dichotomy process for splitting in two an isolating interval of a real root of a function depending on one variable. The algorithm is not detailed here for brevity. The main idea is to divide by two each interval I_i of $B = I_1 \times \cdots \times I_s$ which is larger than $|B|/2$ while keeping the **DC** condition.

Algorithm 7 is a generalization of Algorithm 1 for a zs-rc. Line 1 isolates the real roots of the univariate polynomial T_{x_1}. The variable *toDo* is a set of $(T_{>x_i}, (B, T_{\leq x_i}))$ such that each $(B, T_{\leq x_i})$ satisfies **DC**. It means that $(B, T_{\leq x_i})$ represents one (and only one) real root of $T_{\leq x_i}$. The set $T_{>x_i}$ simply is the set of polynomials which have not be solved yet. Algorithm 7 calls Algorithm 8 (which allows to solve one new variable) until all variables are solved. Note that Algorithm 7 could be followed by a refinement of each returned box so that the width of each box is smaller than a given precision.

Also remark that any raised exception will hit Algorithm 7 since none of the algorithms presented here catches any exception. It is however very easy to adjust Algorithm 7 so that it would catch exceptions and recall itself on each regular chain returned by the splitting. The recursion would eventually stop because of condition **C2** of Algorithm 5 (i.e. splittings cannot occur indefinitely).

Algorithm 8 finds the real roots of p (seen as univariate in x_{s+1}) that "prolong" the

Algorithm 7 RealRootIsolate(T)

Input: T is a zs-rc
Output: a box-decomposition $B_1, \ldots, B_p$ of T

$I_1, \ldots, I_t \leftarrow \mathsf{RootIsolateVCA}(T_{x_1})$
$toDo \leftarrow \{(T_{>x_1}, (I_i, T_{\leq x_1}))\}_{1 \leq i \leq t}$
$res \leftarrow \emptyset$
while $toDo \neq \emptyset$ **do**
  pick and remove a $(T_{>x_i}, (B, T_{\leq x_i}))$ from $toDo$
  $B'_1, \ldots, B'_{t'} \leftarrow \mathsf{SolveNewVar}(T_{x_{i+1}}, B, T_{\leq x_i})$
  if $x_{i+1} = x_n$ **then**
    $res \leftarrow res \cup \{B'_1, \ldots, B'_{t'}\}$
  else
    $toDo \leftarrow toDo \cup \{(T_{<x_{i+1}}, (B'_j, T_{\geq x_{i+1}}))\}_{1 \leq j \leq t'}$
  end if
end while
return res

Algorithm 8 SolveNewVar(p, B, T)

Input: T is a zs-rc of $\mathbb{Q}[x_1, \ldots, x_s]$, $p \in \mathbb{Q}[x_1, \ldots, x_{s+1}]$ and $T \cup \{p\}$ is a regular chain and (B, T) satisfies **DC**
Output: a box-decomposition of the roots $(x_1^0, \ldots, x_{s+1}^0)$ of $T \cup \{p\}$ such that $(x_1^0, \ldots, x_s^0)$ is the root of T which lies in B

refine B into a box B' such that $0 \notin \mathsf{EvalBox}(i_p, B')$
compute a bound H on the roots of $p(x_1^0, \ldots, x_s^0, x_{s+1})$
$toDo \leftarrow \{\textsc{task}(p,] - H, H[, B', T)\}$
$res \leftarrow \emptyset$
while $toDo \neq \emptyset$ **do**
  pick and remove a task $\mathcal{M}$ from $toDo$
  for all e in $\mathsf{SolveTask}(\mathcal{M})$ **do**
    if e is a box **then** $res \leftarrow res \cup \{e\}$
    else $toDo \leftarrow toDo \cup \{e\}$
    end if
  end for
end while
return res

real root which lies in B. Line 1 always terminates. Indeed, i_p is regular modulo T, so it does not vanish on any root of T. Thus, ultimately, B' will be small enough so that $0 \notin \mathsf{EvalBox}(i_p, T)$. Refining the box is needed to compute the bound H.

The bound H at line 2 can be computed in the following way. Denote $p = \sum_{i=0}^{d} a_i x_{s+1}^i$ and $A_i = \mathsf{EvalBox}(a_i, B')$. Then take $H = \frac{1}{\min |A_d|} \sum_{i=0}^{d} (\max |A_i|)$ where $\min |A_i|$ (resp. $\max |A_i|$) denotes the minimum (resp. maximum) of the modulus of the bounds of the interval A_i. The rest of the algorithm is based on Algorithm 9 which transforms tasks into new tasks and boxes.

Algorithm 9 $\mathsf{SolveTask}(\mathcal{M})$

Input: a task $\mathcal{M} = \textsc{task}(p,]a, b[, B, T)$ where T is a zs-rc of $\mathbb{Q}[x_1, \ldots, x_s]$
Output: one of the following cases:

- $\emptyset$ which means $V_t(\mathcal{M}) = \emptyset$
- a box B' such that $(B', T \cup \{p\})$ satisfies **DC** and B' is a box-decomposition of $V_t(\mathcal{M})$, which means $V_t(\mathcal{M})$ is composed of only one point
- two tasks $\mathcal{M}_1$ and $\mathcal{M}_2$ such that $V_t(\mathcal{M}_1)$ and $V_t(\mathcal{M}_2)$ forms a partition of $V_t(\mathcal{M})$
- two tasks $\mathcal{M}_1$ and $\mathcal{M}_2$ plus a box B' such that $(B', T \cup \{p\})$ satisfies **DC** and the three sets $V_t(\mathcal{M}_1)$, $V_t(\mathcal{M}_2)$ and $\{x^0\}$ form a partition of $V_t(\mathcal{M})$, where x^0 denotes the only real root of $T \cup \{p\}$ which lies in B'

1: $nsv, B' \leftarrow \mathsf{BoundNumberRoots}(\mathcal{M})$
2: **if** $nsv = 0$ **then**
3: **return** $\emptyset$
4: **else if** $nsv = 1$ **then**
5: refine B' until $(B' \times]a, b[, T \cup \{p\})$ satisfies **DC**
6: **return** $\{B' \times]a, b[\}$
7: **else**
8: $m \leftarrow (a+b)/2 \quad res \leftarrow \emptyset \quad p' \leftarrow p(x_{s+1} = m)$
9: **if** $p' \in \langle T \rangle$ **then** $res \leftarrow \{B' \times \{m\}\}$
10: **else** $\mathsf{CheckZeroDivisor}(p', T)$
11: **end if**
12: **return** $res \cup \{\textsc{task}(p,]a, m[, B', T), \textsc{task}(p,]m, b[, B', T)\}$
13: **end if**

Algorithm 9 is a generalization of Algorithm 2. The cases $nsv = 0$ or 1 are straightforward. When $nsv > 1$, one needs to split the interval $]a, b[$ in two, yielding the two tasks returned on line 12. Lines 8-11 corresponds to the lines 7-8 of Algorithm 2. Indeed, checking if $p(m)$ is zero is transformed into checking if p' lies in $\langle T \rangle$ or is not a zero divisor modulo T.

Algorithm 10 is a generalization of Algorithm 3. One discards the coefficients of p' which lie in $\langle T \rangle$ because they vanish on the real root v which is in B. One also ensures that the other coefficients (the a'_i) are not zero divisors, so they cannot vanish on v. Thus the loop at lines 6-8 always terminates. Moreover, this guarantees that the number of sign variations is correct. Please note that the sequence $a'_e, \ldots, a'_0$ is never empty. Indeed if all coefficients of $\bar{p}$ were in $\langle T \rangle$, then all coefficients of p would also lie in $\langle T \rangle$ (impossible since i_p is regular

Algorithm 10 BoundNumberRoots($\mathcal{M}$)

Input: a task $\mathcal{M} = \text{TASK}(p,]a, b[, B, T)$ where T is a zs-rc of $\mathbb{Q}[x_1, \ldots, x_s]$
Output: (nsv, B') such that :

- $B' \subset B$ and (B', T) satisfies **DC**
- nsv is a bound on the cardinal of $V_t(\mathcal{M})$. The bound is exact if $nsv = 0$ or 1.

1: $\bar{p} \leftarrow (x_{s+1} + 1)^d \, p\left(x_{s+1} = \frac{a\, x_{s+1} + b}{x_{s+1} + 1}\right)$ with $d = \text{mdeg}(p)$
2: denote $\bar{p} = \sum_{i=0}^{d} a_i x_{s+1}^i$
3: $a'_e, \ldots, a'_0 \leftarrow$ the sequence obtained from $a_d, \ldots, a_0$ by removing the a_i belonging to $\langle T \rangle$
4: **for all** a'_i **do** CheckZeroDivisor(a'_i, T) **end for**
5: $B' \leftarrow B$
6: **while** there is an a'_i such that $0 \in \text{EvalBox}(a'_i, B')$ **do**
7: $B' = \text{RefineBox}(B', T)$
8: **end while**
9: **return** the number of sign variations of the sequence
$\text{EvalBox}(a'_e, B'), \text{EvalBox}(a'_{e-1}, B'), \ldots, \text{EvalBox}(a'_0, B')$

modulo T).

2.5 Comparison with other methods

In the introduction we provided a comparison of our work with others. More technical details are reported below.

[25, 26] give algorithmic methods (available in AXIOM) to manipulate real algebraic numbers. These developments were designed for improving *Cylindrical Algebraic Decomposition* (CAD) methods in AXIOM. Although [25] contains all the tools to solve our problem, this paper focuses on the problem of manipulating real algebraic numbers. It does not address directly the problem of isolating the real roots of a given zerodimensional regular chain. [26] provides tools to perform univariate polynomial real root isolation by using *quasi Sylvester sequence* which according to [26] can be faster than the techniques based on the Descartes rules.

[9, 16] present algorithms for isolating real roots of univariate polynomials with algebraic coefficients. Their algorithms require the ideal to be prime, and this condition is ensured by performing univariate factorization [21] into irreducible factors for polynomials with algebraic coefficients. Our method does not require such factorizations and only requires the ideal to be squarefree. Thus, our method replaces a decomposition into prime ideals by regularity tests which are often less costly.

[27] is based on Gröbner basis computations and rational univariate representation. Thus, [27] transforms the initial problem into the problem of isolating the real roots of a univariate polynomial with rational number coefficients

[20] starts from a zerodimensional regular chain (although [20] uses the terminology of characteristic sets) and proceeds variable by variable. Their technique is different from ours. After isolating a real root say x_1^0 for $p_1(x_1) = 0$, they build two univariate polynomials $\overline{p}_2(x_2)$ (the so-called upper bound polynomial) and $\underline{p}_2(x_2)$ (the so-called lower bound polynomial) whose real roots will interleave nicely (see [20, Definition 2]) when the precision on x_1^0 is

sufficiently low, yielding isolation intervals for the variable x_2.

[35] uses a similar techniques as [20]. The main difference is that the authors use explicitly interval arithmetic and contrarily to [20] where the algorithm may have to restart from the beginning with a smaller precision (called *ac*) in some special cases, [35] uses a refinement process (algorithm NSHR) until the real roots of the upper and lower bound polynomial interleave sufficiently.

Such techniques are also used in [7], where the authors consider general zerodimensional triangular systems (which may not be a regular chain) and treat multiple zeros directly.

Quoting the abstract of [23], the Authors use *a powerful reduction strategy based on univariate root finder using Bernstein basis representation and Descartes' rule.* Basically, they reduce the problem to solving univariate polynomials by using the Bernstein basis representation and optimizations based on convex hulls.

3 Implementation

3.1 The SemiAlgebraicSetTools package

The algorithm RealRootIsolate has been coded using exceptions in MAPLE in the module `SemiAlgebraicSetTools` of the `RegularChains` library [17]. We present some implementation issues and optimizations integrated in our code.

Precision. The user can specify a positive precision so all isolation boxes have a width smaller than the given precision. If an infinite precision is provided, then the algorithm only isolates the real roots by refining the boxes the least possible. We take the precision into account as soon as possible in the algorithm, meaning that each time an isolation box is extending with a new variable, one refines the box.

Constraints. The user can restrict the solutions by imposing that some variables lie in a prescribed interval. If the intervals are restrictive (i.e. smaller than the intervals computed using bounds), this helps avoiding useless computations.

The CheckZeroDivisor algorithm is not directly called in our code. Indeed, regularity test can be very expensive and should be avoided as much as possible. When a call CheckZeroDivisor(p, T) returns, one knows that a box B isolating a real root of T can always be refined until the interval EvalBox(p, B) does not meet zero. This is in fact the only reason why we call CheckZeroDivisor. In order to avoid a regularity test, we first try to refine B a few times to see if EvalBox(p, B) still meet zero. If it does not, we do not need to check the regularity.

Refining boxes. In the MAPLE implementation, Algorithm 6 receives an extra parameter x_k. In that case, the box is only refined for the variables smaller than x_k (i.e. the variables x_i with $i \leq k$). This is useful for example at line 7 of Algorithm BoundNumberRoots. Indeed, if $\mathrm{mvar}(a'_i) = x_k$ holds, then it is not necessary to refine the complete box B' to ensure that EvalBox(a'_i, B') does not meet 0.

Change of variables. By slightly modifying algorithms 8 and 9, we call algorithm 10 with $a = 0$ and $b = 1$. This allows to replace the costly operation

$$(x_{s+1} + 1)^d \, p\left(x_{s+1} = \frac{a\, x_{s+1} + b}{x_{s+1} + 1}\right)$$

by substitutions of the form $p(x_{s+1} = x_{s+1}/2)$, $p(x_{s+1} = 1/x_{s+1})$ and $p(x_{s+1} = x_{s+1} + 1)$ which can be written very efficiently, the last one being based on fast Taylor shift [33].

Refining other branches. Due to the triangular structure of the root, many different roots share a common part (meaning the values for the smaller variables are equal). When one refines a root, we also refine other roots which share a common part to avoid unnecessary computations.

Further refining. After being computed, an isolation box isolating a real root v can be refined further using the MAPLE command RefineBox. To do so, exceptions has to be caught. Our implementation associates a regular chain T to each box B encoding a real root. Thus, if T is split into $T_1, \ldots, T_s$ after an exception, one replaces (B, T) by the right (B, T_i) which also defines the real root v as done in [26, page 528].

EvalPoly. For evaluating EvalBox(p, B), we first collect p using a Hörner scheme. For example, the polynomial $p := x_2^3 x_1 + 3x_2^2 + x_2 x_1^2 + x_1^2 + x_1 + 1$ is collected as $1 + (1 + x_1)x_1 + (x_1^2 + (3 + x_1 x_2)x_2)x_2$. This strategy seems to behave quite well on our examples. The intuition for doing that is the following. Since $x_2 > x_1$ for our ordering, the interval of B for the variable x_2 tend to be in practice wider than that for the variable x_1, since the intervals for smaller variables tend to be more refined than those for higher variables. On the example, the Hörner collected form tends to decrease the exponentiations for x_2.

3.2 Further development

Using fast polynomial arithmetic and modular methods. The current implementation of the CheckZeroDivisor algorithm can be improved in a significant manner. Indeed, the modular algorithm for regularity test reported in [18] and implemented with the MODPN library [19] outperform the regularity test used in CheckZeroDivisor by several orders of magnitude.

Computing with algebraic numbers. Using the two algorithms RefineBox and CheckZeroDivisor, one can easily encode algebraic numbers and check if a multivariate polynomial cancels on some algebraic numbers. This allows computing with algebraic numbers, very much as it is done in [25]. Moreover, inequations and inequalities could be included with almost no work. Indeed they can be handled easily at the end of RealRootIsolate using CheckZeroDivisor. They can also be treated inside the subalgorithms as soon as a box in construction involves all the variables of an inequality or inequation, allowing to cut some branches.

Floating-point computations. As suggested by Fabrice Rouillier (private communication), it would speed up the algorithm to use multiple-precision floating-point computations with exact rounding (as provided by the MPFI library [28]) instead of rational numbers.

Handling exceptions. Exceptions could be caught sooner so one does not lose the computations already done.

Continuous fractions. Techniques bases on continuous fractions [2, 3] may also be investigated.

Interval arithmetics. The algorithm EvalBox could certainly be improved by techniques such as [6] where the polynomial to evaluate is factorized using greedy algorithms.

Newton method. Some tries were made to incorporate a Newton method for system of polynomials in the RefineBox algorithm. Due to the triangular form of the system, the jacobian is also triangular which eases the method. However, although the convergence was really faster, it was not satisfactory because of the coefficient swell of the isolation intervals. However, we believe the Newton method should be investigated more carefully.

4 Benchmarks

4.1 Description of the experimentation

The names of the examples used for benchmarking are listed in Figure 4.1. Most of them are classical. The lhlp files tests are taken from [20]. The examples *chemical-reaction*, *geometric-constraints*, *neural-network*, *p3p-special* and *Takeuchi-Lu* appear in [34]. The nld-*d*-*n* and nql-*n*-*d* examples are described in Section 4.3. All examples can be found at `www.lifl.fr/~lemaire/BCLM09/BCLM09-systems.txt`.

Benchmark results are given on Figure 4.1. They were run on an Intel(R) Pentium(R) D CPU 3.00GHz with 2Gbytes of memory, using MAPLE 13 beta 64bits. Timings are in seconds. Timeouts are indicated with the sign $>$, meaning that the computation was aborted. The first column *Sys* denotes the name of the system. The second column *v/e/s* stands for the number of variables/equations/real solutions.

The Maple command RootFinding[Isolate] isolates real roots within the times indicated in the group of columns RF/Is. For multivariate systems, this command relies on Gröbner basis computations [15] and rational univariate representation [27]. In Column 1, the command used is RootFinding [Isolate](sys, variables, digits=10, output=interval). For Column 2 the same command is used but that the ordering of the variables has been reversed. We used those two commands in case the variable ordering has an effect on the command RootFinding[Isolate]. Note that the option digits=10 ensures that the ten first digits of the results are correct which is not the same as guaranteeing a width less than `1e-10` for the isolation boxes in RealRootIsolate. However, the difficulty for isolating the real roots is comparable if the real roots are not too close to zero nor too big; this is the case for our test examples.

The other groups of columns correspond to three strategies for isolating real roots using our algorithm RealRootIsolate. In each strategy, the initial system is first decomposed into zerodimensional regular chains using the Triangularize command together with the option radical='yes' ensuring those regular chains are squarefree. In order to keep things simple and uniform, the option probability=xx of Triangularize is not used, even when it could be, that is, for square systems generating radical ideals. Therefore the modular algorithm of [10] is not applied even though it can solve all our examples that the non-modular algorithm of Triangularize cannot.

Strategy 1. We build regular chains (column Tr) and call the RealRootIsolate algorithm (column Is/10) on each regular chain with a precision of `1e-10`.

Strategy 2. A variant of Strategy 1 where we compute strongly normalized regular chains (column Tr/No) using the option normalized='strongly' of Triangularize.

Strategy 3. Another variant of Strategy 1. We build regular chains (column Tr) and call the RealRootIsolate algorithm on each regular chain with an infinite precision (column Is/∞), in the sense that the width of the boxes are not constrained. Thus, only the isolation is performed. Then we call the command RefineListBox to refine the list of boxes with a precision of `1e-5` (column ∞/5). Then we refine again the boxes for a precision of `1e-10` (column 5/10).

4.2 Comparison of different strategies

Strategies 1 and 2 are comparable. Strongly normalized regular chains take more time to be computed, since normalization is a post-processing for the command Triangularize. The isolation time is roughly the same in general for both types of regular chains. For the nld-*d*-*n* (except nld-9-3) family of examples, normalization helps the isolation process. However,

		RF/Is		Strategy 1		Strategy 2		Strategy 3			
Sys	v/e/s	1	2	Tr	Is/10	Tr/No	Is/10	Tr	Is/∞	∞/5	5/10
4-body-homog	3/3/7	0.31	0.32	1.6	11	6.2	11	1.5	3.4	4	4.1
5-body-homog	3/3/11	0.31	0.36	3.1	32	38	43	3.2	9.4	11	12
Arnborg-Lazard-rev	3/3/8	<0.1	<0.1	0.43	7	0.5	6.6	0.38	1.8	3	2.6
Arnborg-Lazard	3/3/8	<0.1	<0.1	0.46	7.3	0.62	6.4	0.53	2	3.1	3
Barry	3/3/2	<0.1	<0.1	<0.1	1.5	0.11	3.8	<0.1	0.19	0.64	0.51
Caprasse-Li	4/4/18	0.13	0.14	1.2	3.1	1.4	1.7	1.1	0.44	1.4	1.2
Caprasse	4/4/18	0.13	0.12	1.2	2.9	1.5	2	1.2	0.52	1.6	1.4
chemical-reaction	4/4/4	<0.1	<0.1	0.11	2.8	0.16	2.2	0.12	0.46	1.7	1.1
circles	2/2/22	0.89	0.9	0.55	26	1.1	26	0.59	16	4.6	4.5
cyclic-5	5/5/10	0.4	0.4	2.4	4.6	3.6	1.4	2.5	0.67	3.9	1.8
Czapor-Geddes-Wang	5/5/2	<0.1	0.13	3	6	18	7.9	2.8	2.3	2.5	2.1
fabfaux	3/3/3	<0.1	<0.1	2.4	7.3	51	8.5	2.5	2	3.2	3.3
geometric-constraints	3/3/8	<0.1	<0.1	<0.1	2.2	<0.1	2.2	<0.1	0.27	1.2	0.88
GonzalezGonzalez	3/3/2	<0.1	<0.1	<0.1	0.76	0.13	0.75	0.1	0.15	0.4	0.36
Katsura-4	5/5/12	<0.1	<0.1	0.54	14	0.77	19	0.58	2.9	6.8	6.5
lhlp1	3/3/6	<0.1	<0.1	<0.1	1	<0.1	1.5	<0.1	0.14	0.53	0.39
lhlp2	3/3/2	<0.1	<0.1	<0.1	0.95	<0.1	1.4	<0.1	0.19	0.51	0.38
lhlp3	3/3/2	<0.1	<0.1	<0.1	0.63	<0.1	0.75	<0.1	<0.1	0.29	0.24
lhlp4	2/2/4	<0.1	<0.1	<0.1	2.9	<0.1	4.8	<0.1	0.48	1.5	1
lhlp5	3/3/4	<0.1	<0.1	0.26	2	0.26	2.4	0.22	0.35	0.95	0.76
lhlp6	4/4/4	<0.1	<0.1	0.26	2.4	0.34	1.7	0.23	0.36	1.5	0.78
neural-network	4/4/22	1	1	0.81	18	1.2	15	0.87	4.5	7.7	7
nld-3-4	4/4/27	1.1	1.1	3.4	13	4.3	5.1	3.2	2.2	6.6	5.8
nld-3-5	5/5/111	79	79	1804	406	1961	45	1832	67	134	133
nld-4-5	5/5/?	>2000	>2000	>2000	?	>2000	?	>2000	?	?	?
nld-7-3	3/3/7	96	95	5.8	9.8	9.1	7.7	5.8	11	0.37	0.16
nld-8-3	3/3/8	457	456	4	29	21	17	4	25	4.3	2.4
nld-9-3	3/3/7	1785	1777	39	43	121	70	40	45	0.34	0.29
nld-10-3	3/3/8	>2000	>2000	26	148	370	308	25	148	8.1	8.1
nql-5-4	5/5/2	109	102	0.1	1.3	0.12	1.3	0.1	0.39	0.27	0.39
nql-10-2	10/10/2	250	225	0.15	3.1	0.29	3.2	0.2	0.98	0.67	0.99
nql-10-4	10/10/2	>2000	>2000	0.33	3.2	0.61	3.3	0.34	0.92	0.62	0.83
nql-15-2	15/15/2	>2000	>2000	0.36	5.8	0.65	5.7	0.33	3.1	1.3	1.9
p3p-special	5/5/24	0.41	0.46	0.23	23	0.69	31	0.24	6.4	8.2	9
PlateForme2d-easy	6/6/0	<0.1	<0.1	1.1	0.12	1.4	0.12	0.99	<0.1	<0.1	<0.1
r-5	5/5/1	1.6	1.6	0.43	<0.1	0.49	<0.1	0.37	<0.1	<0.1	<0.1
r-6	6/6/1	>2000	>2000	0.96	<0.1	1.2	<0.1	0.98	<0.1	<0.1	<0.1
Rose	3/3/18	0.63	0.67	0.72	39	1.1	59	0.71	5	22	20
simple-nql-20-30	20/20/2	>2000	>2000	0.57	28	0.88	28	0.63	65	2.8	0.33
Takeuchi-Lu	4/4/14	<0.1	<0.1	0.22	7.2	0.23	9.7	0.17	0.87	4.4	3.1
Trinks-2	6/7/0	<0.1	<0.1	<0.1	<0.1	0.11	<0.1	<0.1	<0.1	<0.1	<0.1
Trinks-difficult	6/6/2	<0.1	<0.1	0.13	2.8	0.22	4	0.13	0.34	1.4	1.3
wilkinson20	1/1/21	<0.1	<0.1	<0.1	1	<0.1	0.98	<0.1	0.12	0.49	0.39
wilkinsonxy	2/2/25	<0.1	<0.1	<0.1	8	<0.1	7.9	<0.1	2.8	2.6	2.6

Figure 1: Benchmarks

for some other examples, such as 5-body-homog, p3p-special and Rose, normalization make things worse.

Compared to Strategy 1, Strategy 3 shows two things. First, it is usually faster to isolate solutions with an infinite precision rather than with a small precision. Secondly, it shows that the overall times for Strategies 1 and 3 are comparable.

4.3 Comparison with RootFinding

The RootFinding[Isolate] is obviously a lot faster on many examples. One should keep in mind that this command calls internal routines written in C that have been developed intensively for years. However, the RootFinding[Isolate] has difficulties on some systems such as the nql-n-d and nld-d-n ones.

The nql-n-d (for non quasi linear) example is very specific and was suggested by Fabrice Rouillier. It is defined by n equations in n variables $x_1^d - 2 = 0$, $x_i^d + x_i^{d/2} - x_{i-1} = 0$ for $2 \leq i \leq n$ for some even degree d. This system is already a zs-rc. The algorithm RealRootIsolate solves it easily since the degrees are distributed evenly among the equations. On the other hand, the RootFinding[Isolate] needs to build a rational univariate representation which we believe has a very large degree roughly equal to d^n (that is about one million when $d = 4$ and $n = 10$).

A similar example is simple-nql-n-d defined by $x_1^d - 2 = 0$, $x_i^d - x_{i-1} = 0$ for $2 \leq i \leq n$. The degree of the rational univariate representation is also roughly d^n. For the example simple-nql-20-30, d^n is around 10^{29}.

The second family of systems which causes difficulties to RootFinding[Isolate] are the nld-d-n (for non leading linear) defined by n equations of the form $x_1 + \cdots + x_{i-1} + x_i^d + x_{i+1} + \cdots + x_n - 1 = 0$ for $1 \leq i \leq n$. On those systems the computations performed by Triangularize tend to split into many branches, even though the equiprojectable decomposition consists of a few components (generally 2). For System nld-9-3, the command Triangularize (used without normalization option) produces 16 components where the largest coefficient has size 20 digits. The command EquiprojectableDecomposition (which requires the use normalized regular chains) produces 3 components for nld-9-3, where most coefficients have more than 1,000 digits. Since nld-9-3 has 729 complex solutions, this suggests that the univariate polynomial in the rational univariate representation has degree 729 and coefficients with size at least 1,000 digits. This makes it difficult to isolate the real roots of such polynomial. Therefore, the nld-d-n examples show that splitting can help solving some problems.

5 Conclusion

We presented a generalization of the Vincent-Collins-Akritas Algorithm for zerodimensional squarefree regular chains, and its implementation in Maple. Each box isolating a root can be refined arbitrarily after being computed. This allows manipulating algebraic numbers (encoded by a isolation box and a regular chain) very much like in [25]. Many improvements of the algorithm RealRootIsolate are possible and should be investigated. Among them, we believe that writing a C library to perform the isolation would improve a lot the timings. Yet for non-equiprojectable varieties, our algorithm and its Maple implementation show favorable performances.

References and Notes

[1] Alkiviadis G. Akritas. There is no Uspensky's method. In *proceedings of ISSAC 1986*, 1986.

[2] Alkiviadis G. Akritas, Adam W. Strzeboński, and Panagiotis S. Vigklas. Implementations of a new theorem for computing bounds for positive roots of polynomials. *Computing*, 78(4):355–367, 2006.

[3] Alkiviadis G. Akritas and Panagiotis S. Vigklas. A Comparison of Various Methods for Computing Bounds for Positive Roots of Polynomials. *Journal of Universal Computer Science*, 13(4):455–467, 2007.

[4] Philippe Aubry and Marc Moreno Maza. Triangular sets for solving polynomial systems: A comparative implementation of four methods. *J. Symb. Comp.*, 28(1-2):125–154, 1999.

[5] Eberhard Becker, Teo Mora, Maria G. Marinari, and Carlo Traverso. The shape of the shape lemma. In *Proc. of the international symposium on Symbolic and algebraic computation*, pages 129–133, New York, NY, USA, 1994. ACM Press.

[6] Martine Ceberio and Vladik Kreinovich. Greedy algorithms for optimizing multivariate horner schemes. *SIGSAM Bull.*, 38(1):8–15, 2004.

[7] Jin-San Cheng, Xiao-Shan Gao, and Chee-Keng Yap. Complete numerical isolation of real zeros in zero-dimensional triangular systems. In *ISSAC '07: Proceedings of the 2007 international symposium on Symbolic and algebraic computation*, pages 92–99, 2007.

[8] George E. Collins and Alkiviadis G. Akritas. Polynomial real root isolation using Descartes'rule of signs. In *proceedings of ISSAC'76*, pages 272–275, Yorktown Heights NY, 1976.

[9] George E. Collins, Jeremy R. Johnson, and Werner Krandick. Interval arithmetic in cylindrical algebraic decomposition. *J. Symb. Comput.*, 34(2):145–157, 2002.

[10] Xavier Dahan, Marc Moreno Maza, Éric Schost, Wenyuan Wu, and Yuzhen Xie. Lifting techniques for triangular decompositions. In *ISSAC'05*, pages 108–115. ACM Press, 2005.

[11] Xavier Dahan and Éric Schost. Sharp estimates for triangular sets. In *ISSAC 04*, pages 103–110. ACM, 2004.

[12] Jean Della Dora, Claire Dicrescenzo, and Dominique Duval. About a new method for computing in algebraic number fields. In *Proc. EUROCAL 85 Vol. 2*, volume 204 of *Lect. Notes in Comp. Sci.*, pages 289–290. Springer-Verlag, 1985.

[13] René Descartes. *Géométrie*. 1636.

[14] Arno Eigenwillig, Lutz Kettner, Werner Krandick, Kurt Mehlhorn, Susanne Schmitt, and Nicola Wolpert. A descartes algorithm for polynomials with bit-stream coefficients. In *CASC, volume 3718 of LNCS*, pages 138–149. Springer, 2005.

[15] Jean-Charles Faugère. A new efficient algorithm to compute Gröbner bases (F_4). *Journal of Pure and Applied Algebra*, 139:61–88, 1999.

[16] Jeremy R. Johnson and Werner Krandick. Polynomial real root isolation using approximate arithmetic. In *Proceedings of the 1997 international symposium on Symbolic and algebraic computation*, pages 225–232. ACM, 1997.

[17] François Lemaire, Marc Moreno Maza, and Yuzhen Xie. The `RegularChains` library. In Ilias S. Kotsireas, editor, Maple Conference 2005, pages 355–368, 2005.

[18] Xin Li, Marc Moreno Maza, and Wei Pan. Computations modulo regular chains. In *proceedings of ISSAC'09*, pages 239–246, New York, NY, USA, 2009. ACM Press.

[19] Xin Li, Marc Moreno Maza, Raqeeb Rasheed, and Éric Schost. The MODPN library: Bringing fast polynomial arithmetic into MAPLE. In *MICA'08*, 2008.

[20] Zhengyi Lu, Bi He, Yong Luo, and Lu Pan. An algorithm of real root isolation for polynomial systems. In Dongming Wang and Lihong Zhi, editors, *Proceedings of Symbolic Numeric Computation 2005*, pages 94–107, 2005.

[21] Scott McCallum. An improved projection operation for cylindrical algebraic decomposition. In B. F. Caviness and J. R. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 242–268. Springer, 1988.

[22] Marc Moreno Maza. On triangular decompositions of algebraic varieties. Technical Report TR 4/99, NAG Ltd, Oxford, UK, 1999. Presented at the MEGA-2000 Conference, Bath, England.

[23] Bernard Mourrain and Jean-Pascal Pavone. Subdivision methods for solving polynomial equations. Technical report, INRIA, August 2005. (number RR-5658).

[24] Bernard Mourrain, Fabrice Rouillier, and Marie-Françoise Roy. The Bernstein Basis and Real Root Isolation. *Combinatorial and Computational Geometry, MSRI Publications*, 52:459–478, 2005.

[25] Renaud Rioboo. Real algebraic closure of an ordered field, implementation in axiom. In *Proc. ISSAC'92*, pages 206–215. ISSAC, ACM Press, 1992.

[26] Renaud Rioboo. Towards faster real algebraic numbers. *J. Symb. Comput.*, 36(3-4):513–533, 2003.

[27] Fabrice Rouillier. Solving zero-dimensional systems through the rational univariate representation. *AAECC*, 9:433–461, 1999.

[28] Fabrice Rouillier and Nathalie Revol. The multiple precision floating-point interval library (MPFI) library. http://gforge.inria.fr/projects/mpfi/.

[29] Fabrice Rouillier and Paul Zimmermann. Efficient isolation of polynomial real roots. *Journal of Computational and Applied Mathematics*, 162(1):33–50, 2003.

[30] Joseph Alfred Serret. *Cours d'Algèbre Supérieure*. Gauthier–Villars, Paris, 4 edition, 1877.

[31] James Victor Uspensky. *Theory of Equations*. McGraw – Hill Co., New–York, 1948.

[32] Alexandre Joseph Hidulphe Vincent. Sur la résolution des équations numériques. *Journal de Mathématiques Pures et Appliquées*, 1:341–372, 1836.

[33] Joachim von zur Gathen and Jürgen Gerhard. Fast algorithms for taylor shifts and certain difference equations. In *ISSAC*, pages 40–47, 1997.

[34] Bican Xia and Lu Yang. An algorithm for isolating the real solutions of semi-algebraic systems. *J. Symb. Comput.*, 34(5):461–477, 2002.

[35] Bican Xia and Ting Zhang. Real solution isolation using interval arithmetic. *Comput. Math. Appl.*, 52(6-7):853–860, 2006.