Real Root Isolation of Regular Chains

François Boulier, Changbo Chen, François Lemaire and Marc Moreno Maza

Abstract We present an algorithm RealRootIsolate for isolating the real roots of a polynomial system 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 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 [6].

There exist many different approaches for isolating real roots of univariate polynomials by means of Descartes rules of signs [11]. Uspensky [30] rediscovered

F. Boulier · F. Lemaire (✉)
LIFL, Université de Lille 1, 59655 Villeneuve D'Ascq Cedex, France
e-mail: Francois.Boulier@lifl.fr

F. Lemaire
e-mail: Francois.Lemaire@lifl.fr

C. Chen · M. Moreno Maza (✉)
ORCCA, University of Western Ontario (UWO), London, ON, Canada

C. Chen
Chongqing Key Laboratory of Automated Reasoning and Cognition, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China
e-mail: changbo.chen@hotmail.com

M.Moreno Maza
e-mail: moreno@csd.uwo.ca

R. Feng et al. (eds.), *Computer Mathematics*, DOI 10.1007/978-3-662-43799-5_4

independently[1] 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 [28] is efficient in memory since it avoids the storage of one polynomial at each node of the tree of the recursive calls.

The methods mentioned above are all for univariate polynomials with integral or rational coefficients. In [12], the authors apply Descartes Algorithm for polynomials with bit-stream coefficients. In [7, 14], 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 [26], polyhedron algebra [22], and triangular decompositions [5, 19, 24, 33].

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 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. 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 [24, 25]. 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 uses Sturm sequences and subresultants for univariate polynomial real root isolation.

Other real root isolation algorithms based on triangular decompositions [5, 19, 33] rely on the so-called "sleeve polynomials", see Sect. 2.5.

We do not report on a comparative implementation with the methods in [5, 7, 19, 24, 33]. In order to ensure a fair comparison, one would need to bring these six real root isolation methods (including ours) in a common implementation framework, which would require a significant amount of 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. As shown in [17], these techniques should speedup our implementation dramatically.

We compare our code with the RootFinding[Isolate] command (available in MAPLE) based on the rational univariate representation [26]. 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 [4].

[1] Recent investigations of A. Akritas seem to prove that Uspensky only had an incomplete knowledge of Vincent's paper, from [29, pp. 363–368].

However, for different families of examples, corresponding to non-equiprojectable[2] varieties the situation is reversed which demonstrates the 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.

We investigate the impact of different strategies for isolating roots. 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 speedup the root isolation even though normalization tends to substantially increase coefficient sizes, as established in [9].

2 Real Root Isolation of a Zerodimensional Regular Chain

After recalling the Vincent-Collins-Akritas algorithm in Sect. 2.1 and introducing definitions in Sects. 2.2 and 2.3, the algorithm RealRootIsolate and its subalgorithms are presented in Sect. 2.4. In Sect. 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, before its generalization in Sect. 2.4.

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

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].

The main arguments for the correctness of Algorithm 1 are the following. Algorithm 1 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 the number of real roots of p which lie in $]a, b[$. This bound is exact when equal to 0 or 1 [23, Theorem 1.2]. Since p is squarefree, the bound returned by Algorithm 3 will eventually become 0 or 1, by [23, Theorem 2.5] so that the whole method terminates.

[2] The notions of an equiprojectable variety and equiprojectable decomposition are discussed in [8].

Algorithm 1 RootIsolateVCA(p)

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

$H \leftarrow$ a strict bound on the roots of p
return RootIsolateAuxVCA($p,]-H, H[$)

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 $]a, b[$

$nsv \leftarrow$ BoundNumberRootsVCA($p,]a, b[$)
if $nsv = 0$ **then return** $\emptyset$
else if $nsv = 1$ **then return** $]a, b[$
else
  $m \leftarrow (a + b)/2 \quad res \leftarrow \emptyset$
  if $p(m) = 0$ **then** $res \leftarrow \{\{m\}\}$
  {Next line ensures the roots are sorted increasingly}
  return RootIsolateAuxVCA($p,]a, m[$) $\cup$ res $\cup$ RootIsolateAuxVCA($p,]m, b[$)

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 $]a, b[$

$\bar{p} \leftarrow (x + 1)^d \, p\left(\frac{a\,x+b}{x+1}\right)$ where d is the degree of p, and denote $\bar{p} = \sum_{i=0}^{d} a_i x^i$
$a'_e, \ldots, a'_0 \leftarrow$ the sequence obtained from $a_d, \ldots, a_0$ by removing zero coefficients
return the number of sign variations in the sequence $a'_e, \ldots, a'_0$

2.2 Regular Chains

In this paper one only considers zerodimensional squarefree regular chains, denoted *zs-rc*. Roughly speaking, a zerodimensional regular chain is a triangular set[3] 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).

Let $x_1 < \cdots < x_s$ be s variables, and denote $\mathbb{Q}_s = \mathbb{Q}[x_1, \ldots, x_s]$. Let $p \in \mathbb{Q}_s$ be a non-constant polynomial. We denote by $\text{mvar}(p)$ the *main variable* of p, by $\text{init}(p)$ the *initial* (or leading coefficient w.r.t. $\text{mvar}(p)$) of p, by $\text{mdeg}(p)$ the degree of p in its main variable and by $\text{sep}(p)$ the *separant* of p, that is, $\partial p/\partial \text{mvar}(p)$. If T is a set of polynomials in $\mathbb{Q}_s$, $\langle T \rangle$ 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 $\mathbb{Q}_s$. The set T is a *zerodimensional squarefree regular chain* (or *zs-rc*) of $\mathbb{Q}_s$ if $\text{mvar}(p_i) = x_i$ for $1 \leq i \leq s$, $\text{init}(p_i)$ does not vanish on $V(\{p_1, \ldots, p_{i-1}\})$ for any $2 \leq i \leq s$, and $\text{sep}(p_i)$ does not vanish on $V(\{p_1, \ldots, p_i\})$ for any $1 \leq i \leq s$.

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

Thanks to the first two conditions, it is easy to show that the system $T = 0$ has a finite number of complex solutions (counted with multiplicity), which is equal to the product of the main degrees of the elements of T denoted $\text{DEG}(T)$. 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 TriangularizeAlgorithm [21] 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 T if p is regular modulo the ideal $\langle T\rangle$. Thus, the following definition is equivalent to Definition 2.

Definition 3 Let $T = \{p_1, \ldots, p_s\}$ where each p_i lies in $\mathbb{Q}_s$. The set T is a *zerodimensional squarefree regular chain* (or *zs-rc*) of $\mathbb{Q}_s$ if $\text{mvar}(p_i) = x_i$ for any $1 \le i \le s$, $\text{init}(p_i)$ is regular modulo the ideal $\langle p_1, \ldots, p_{i-1}\rangle$ for any $2 \le i \le s$, and $\text{sep}(p_i)$ is regular modulo the ideal $\langle p_1, \ldots, p_i\rangle$ for any $1 \le i \le s$.

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 $\mathbb{Q}_s$ and q a polynomial of $\mathbb{Q}_s$. 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 Sect. 2.4.

Definition 4 An s-box (or box) B is a Cartesian product $B = I_1 \ldots I_s$ 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 $I_i =]a, b[$.

The algorithm $\text{EvalBox}(p, B)$, where $p \in \mathbb{Q}_s$, and B is a s-box, returns an interval I such that $p(v) \in I$ for any $v \in B$. Different variants for $\text{EvalBox}(p, B)$ exist. Any variant for $\text{EvalBox}(p, B)$ satisfying the following property can be used: the box $\text{EvalBox}(p, B)$ should tend to the singleton $\{p(x_0)\}$ when the width of B tends to zero (by keeping the condition $x_0 \in B$). This simply ensures that the interval $\text{EvalBox}(p, B)$ should shrink as the width of the box B decreases.

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

- one and only one real root of T lies in B
- if $I_1 =]a, b[$, $p_1(x_1 = a)$ and $p_1(x_1 = b)$ are nonzero and have opposite signs
- for each $2 \le k \le s$, 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.[4]

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.

Definition 6 Let V be a finite set of t points of $\mathbb{R}^s$. A list $B_1, \dots, B_t$ of s-boxes is called a *box-decomposition* of V if each point of V lies in exactly one B_i and $B_i \cap B_j = \emptyset$ if $i \ne j$. 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 $\mathbb{Q}_s$, $p \in \mathbb{Q}_{s+1}$, $T \cup \{p\}$ is a zs-rc, B is an s-box, (B, T) satisfies **DC**, and $a < b$ are rational numbers. The solutions of $\mathcal{M}$ denoted by $V_t(\mathcal{M})$ are 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 $\mathsf{RealRootIsolate}$, which isolates the real roots of a zerodimensional squarefree regular chain, is presented here. Only elements of proofs are given. However, specifications are stated with details. One assumes $n > 1$.

The algorithms presented here use the mechanism of exceptions which is available in a lot of programming languages. We find it appropriate since doing computations using the D5 principle [10] can be seen as doing computations as if the coefficient ring were 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[5] the algorithms presented here. Only Algorithm 4 throws exceptions.

Algorithm 4 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 4 is called whenever one needs to know whether a polynomial vanishes or not, 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 allows to refine B until $\mathsf{EvalBox}(p, B)$ does not contain 0, which gives the sign of $p(x^0)$.

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

[5] otherwise, splittings need to be handled each time a function returns a value.

Algorithm 4 CheckZeroDivisor(p, T)

Input: T a zs-rc $\mathbb{Q}_s$ and $p \in \mathbb{Q}_s$

Output: If p is regular modulo T, then the algorithm terminates normally. Otherwise, an exception is thrown exhibiting t zs-rc $T_1, \ldots, T_t$ such that **C1** $V(T_1) \cup \cdots \cup V(T_t) = V(T)$, and **C2** $\sum_{i=1}^{t} \text{DEG}(T_i) = \text{DEG}(T)$ hold.

$T_1, \ldots, T_t \leftarrow$ Regularize(p, T)

if p belongs to at least one $\langle T_i \rangle$ **then throw exception**$(T_1, \ldots, T_t)$

Algorithm 5 RefineBox(B, T)

Input: T is a zs-rc of $\mathbb{Q}_s$, (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 RealRootIsolate(T)

Input: T is a zs-rc

Output: a box-decomposition $B_1, \ldots, B_p$ of T

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

The algorithm Regularizeis not recalled here (see [21] 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 4. Due to condition **C2**, splittings cannot occur indefinitely.

Algorithm 5 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 into two an isolating interval. 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 6 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 6 calls Algorithm 7 (which allows to solve one new variable) until all variables are solved. Note that Algorithm 6 could be followed by a refinement of each returned box so that the width of each box is smaller than a given precision.

Algorithm 7 SolveNewVar(p, B, T)

Input: T is a zs-rc of $\mathbb{Q}_s$, $p \in \mathbb{Q}_{s+1}$, $T \cup \{p\}$ is a regular chain, (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

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

Algorithm 8 SolveTask$(\mathcal{M})$

Input: a task $\mathcal{M} = \text{TASK}(p,]a, b[, B, T)$ where T is a zs-rc of $\mathbb{Q}_s$
Output: One of the four following cases:
1: $\emptyset$ which means $V_t(\mathcal{M}) = \emptyset$.
2: 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 **3**: 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 $V_t(\mathcal{M})$ **4**: 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'.

$nsv, B' \leftarrow$ BoundNumberRoots$(\mathcal{M})$
if $nsv = 0$ **then return** $\emptyset$
else if $nsv = 1$ **then**
$B'' \leftarrow B' \times]a, b[$
refine B'' until $(B'', T \cup \{p\})$ satisfies **DC**
return $\{B''\}$
else
$m \leftarrow (a+b)/2 \quad res \leftarrow \emptyset \quad p' \leftarrow p(x_{s+1} = m)$
if $p' \in \langle T \rangle$ **then** $res \leftarrow \{B' \times \{m\}\}$ **else** CheckZeroDivisor(p', T)
return $res \cup \{\text{TASK}(p,]a, m[, B', T), \text{TASK}(p,]m, b[, B', T)\}$

Also remark that any raised exception will hit Algorithm 6 since none of the algorithms presented here catches any exception. It is however easy to adjust Algorithm 6 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 4 (i.e. splittings cannot occur indefinitely).

Algorithm 7 finds the real roots of p (seen as univariate in x_{s+1}) that "prolong" the real root which lies in B. Line 1 always terminates, since i_p is regular modulo T, so it does not vanish on any root of T. 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 $A_i = \text{EvalBox}(a_i, B')$. 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 8 which transforms tasks into new tasks and boxes.

Algorithm 9 BoundNumberRoots($\mathcal{M}$)

Input: a task $\mathcal{M} = \text{TASK}(p,]a, b[, B, T)$ where T is a zs-rc of $\mathbb{Q}_s$
Output: (nsv, B') such that $B' \subset B$, (B', T) satisfies **DC**, and 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)
5: $B' \leftarrow B$
6: **while** there is an a'_i such that $0 \in$ EvalBox(a'_i, B') **do** $B' =$ RefineBox(B', T)
7: **return** the number of sign variations of the sequence
EvalBox(a'_e, B'), EvalBox(a'_{e-1}, B'), ..., EvalBox(a'_0, B')

Algorithm 8 is a generalization of Algorithm 2. The cases $nsv = 0$ or 1 are straightforward. When $nsv > 1$, one needs to split the interval $]a, b[$ into two, yielding two tasks returned on line 10. Lines 8–9 correspond to the lines 5–6 of Algorithm 2. Indeed, checking $p(m) = 0$ is transformed into checking if p' lies in $\langle T \rangle$ or is not a zero divisor modulo T.

Algorithm 9 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 line 6 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 a_i's were in $\langle T \rangle$, then all coefficients of p would lie in $\langle T \rangle$ (impossible since i_p is regular 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.

References [24, 25] 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 [24] 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 [25] provides tools to perform univariate polynomial real root isolation by using *quasi Sylvester sequence* which according to [25] can be faster than the techniques based on the Descartes rules.

References [7, 14] 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 [20] 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. Please note that the regularity tests we perform are in fact replaced by interval arithmetics, as explained in the paragraph CheckZeroDivisorof Sect. 3.1.

Reference [26] is based on Gröbner basis computations and rational univariate representation. Thus, [26] transforms the initial problem into the problem of isolating the real roots of a univariate polynomial with rational number coefficients

Reference [19] starts from a zerodimensional regular chain (although [19] 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 $\overline{p}_2(x_2)$ and $\underline{p}_2(x_2)$ whose real roots will interleave nicely (see [19, Definition 2]) when the precision on x_1^0 is sufficiently low, yielding isolation intervals for the variable x_2 [33] improves techniques of [19] by avoiding restarting the isolation from the beginning when decreasing the precision. Such techniques are also used in [5], where the authors consider general zerodimensional triangular systems (which may not be a regular chain) and treat multiple zeros directly.

Quoting the abstract of [22], 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 [16]. 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 the box is refined each time an box is extended with a new variable.

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), many useless branches are cut.

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 meets zero. If it does not, we do not need to check the regularity.

Refining boxes. In the MAPLE implementation, Algorithm 5 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 at line 6 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 modifying Algorithms 7 and 8, we call Algorithm 9 with $a = 0$ and $b = 1$. This replaces $(x_{s+1}+1)^d\ p\left(x_{s+1} = \frac{a\,x_{s+1}+b}{x_{s+1}+1}\right)$ by several substitutions $p(x_{s+1} = x_{s+1}/2)$, $p(x_{s+1} = 1/x_{s+1})$ and $p(x_{s+1} = x_{s+1}+1)$ which can be written efficiently, the last one using fast Taylor shift [31].

Refining other branches. Due to the triangular structure of the system, many roots share a common part (i.e., values for some variables are equal). When refining a root, we refine the roots sharing a common part to save computations.

Further refining. After being computed, an isolation box of a real root v can be refined further using the MAPLE command RefineBox. To do so, exceptions have to be caught. Our implementation associates a regular chain T to each box B encoding a real root. Thus, if T is split into $T_1, \ldots, T_s$, one replaces (B, T) by the right (B, T_i) which also defines the real root v as done in [25, p. 528].

EvalPoly. For evaluating EvalBox(p, B), we apply a Hörner scheme to p. 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 rearranged as $1 + (1 + x_1)x_1 + (x_1^2 + (3 + x_1 x_2)x_2)x_2$. Assuming $x_2 > x_1$, the interval of B for the variable x_2 tends 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 form decreases the exponents of 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 of [17] and implemented with the MODPN library [18] 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 encode algebraic numbers and check if a multivariate polynomial cancels on some algebraic numbers. This allows computing with algebraic numbers, very much as in [24]. Moreover, inequations and inequalities could be included with almost no work. Indeed they can be handled at the end of RealRootIsolate using CheckZeroDivisor. They can also be treated inside the sub-algorithms 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 speedup the algorithm to use multiple-precision floating-point computations with exact rounding (as in the MPFI library [27]).

Exceptions could be caught sooner to avoid losing already done computations.

Using continuous fractions as in [2, 3] may also be investigated.

Interval arithmetics. The algorithm EvalBox could certainly benefit from techniques for "optimizing" polynomial expressions, as in [15].

Newton's method. Some attempts 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 in the isolation intervals. However, we believe that Newton's method should be investigated further.

4 Benchmarks

4.1 Description of the Experimentation

The names of the examples used for benchmarking are listed in Fig. 1. Most of them are classical. The lhlp files tests are taken from [19]. The examples *chemical-reaction*, *geometric-constraints*, *neural-network*, *p3p-special* and *Takeuchi-Lu* appear in [32]. The nld-*d*-*n* and nql-*n*-*d* examples are described in Sect. 4.3. The set of all the examples can be found at www.lifl.fr/~lemaire/BCLM09/BCLM09-systems.txt. Benchmark results are given in Fig. 1. They were run on an AMD Phenom II X4 (4 Gb of mem.) using MAPLE 14 64 bits and our latest development version of the `RegularChains` library. Timings are in seconds. Timeouts are indicated with >. The column *Sys* denotes the name of the system. The 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 [13] and rational univariate representation [26]. In Column 1, the command used is RootFinding [Isolate](sys, variables, digits=10, output=interval). For Column 2 the same command is used with the variable ordering reversed, in case the variable ordering is important. Note that the option digits = 10 ensures that the first ten 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 in isolating the real roots is comparable since the roots are neither close to zero nor too big.

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 zs-rc using the Triangularize command together with the option

		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	0.16	0.17	0.58	4.1	1.9	4.6	0.58	1.5	1.5	1.5
5-body-homog	3/3/11	0.19	0.2	0.83	11	10	16	0.81	3.5	3.9	3.7
Arnborg-Lazard-rev	3/3/8	<0.1	<0.1	0.35	2.9	0.42	2.8	0.34	0.88	1.1	1
Arnborg-Lazard	3/3/8	<0.1	<0.1	0.36	3	0.42	2.7	0.35	0.91	1.3	1.1
Barry	3/3/2	<0.1	<0.1	0.2	0.64	0.22	1.6	0.19	0.19	0.26	0.2
Caprasse-Li	4/4/18	<0.1	<0.1	0.61	1.4	0.77	0.9	0.63	0.35	0.62	0.46
Caprasse	4/4/18	<0.1	<0.1	0.65	1.4	0.8	0.9	0.62	0.37	0.65	0.5
chemical-reaction	4/4/4	<0.1	<0.1	0.23	1.3	0.26	1	0.23	0.29	0.71	0.45
circles	2/2/22	0.6	0.57	0.32	10	0.41	10	0.32	6.4	1.8	1.6
cyclic-5	5/5/10	0.22	0.22	1.1	1.9	1.6	0.77	1.1	0.4	1.5	0.69
Czapor-Geddes-Wang	5/5/2	<0.1	<0.1	0.98	2.5	5.4	3.3	0.99	1.1	1	0.84
fabfaux	3/3/3	<0.1	<0.1	0.65	2.9	14	3.4	0.66	0.89	1.3	1.3
geometric-constraints	3/3/8	<0.1	<0.1	0.16	0.91	0.18	0.92	0.17	0.23	0.41	0.35
GonzalezGonzalez	3/3/2	<0.1	<0.1	0.21	0.45	0.26	0.42	0.22	0.19	0.16	0.14
Katsura-4	5/5/12	<0.1	<0.1	0.41	5.7	0.51	7.8	0.41	1.4	2.6	2.4
lhlp1	3/3/6	<0.1	<0.1	0.19	0.54	0.21	0.76	0.19	0.19	0.2	0.17
lhlp2	3/3/2	<0.1	<0.1	0.2	0.46	0.24	0.65	0.19	0.18	0.19	0.14
lhlp3	3/3/2	<0.1	<0.1	0.15	0.37	0.18	0.4	0.16	0.14	0.15	<0.1
lhlp4	2/2/4	<0.1	<0.1	0.18	1.3	0.22	2.1	0.18	0.3	0.63	0.42
lhlp5	3/3/4	<0.1	<0.1	0.26	0.97	0.3	1.1	0.27	0.27	0.39	0.34
lhlp6	4/4/4	<0.1	<0.1	0.29	1.1	0.33	0.84	0.28	0.26	0.59	0.34
neural-network	4/4/22	0.57	0.57	0.44	7	0.67	6.1	0.42	1.7	3	2.7
nld-3-4	4/4/27	0.72	0.73	1.1	2.9	1.6	1.9	1.2	0.6	1.5	1.3
nld-3-5	5/5/111	47	47	9.1	23	12	14	8.9	4	12	10
nld-4-5	5/5/?	>2000	>2000	>2000	?	>2000	?	>2000	?	?	?
nld-7-3	3/3/7	58	58	1.7	3.8	2.9	3.1	1.6	4.6	<0.1	<0.1
nld-8-3	3/3/8	275	275	2.1	9.9	11	7	2.1	8.7	1.3	1.2
nld-9-3	3/3/7	1078	1083	7.9	14	32	27	7.9	16	<0.1	<0.1
nld-10-3	3/3/8	>2000	>2000	10	45	341	118	10	44	2.4	2.3
nql-5-4	5/5/2	66	62	0.2	0.59	0.22	0.57	0.2	0.26	0.11	0.14
nql-10-2	10/10/2	144	132	0.25	1.4	0.29	1.4	0.25	0.49	0.31	0.44
nql-10-4	10/10/2	>2000	>2000	0.32	1.4	0.38	1.4	0.31	0.53	0.27	0.37
nql-15-2	15/15/2	>2000	>2000	0.39	2.6	0.45	2.5	0.38	1.5	0.52	0.81
p3p-special	5/5/24	0.22	0.27	0.31	9	0.5	13	0.31	2.8	3.2	3.4
PlateForme2d-easy	6/6/0	<0.1	<0.1	0.57	0.16	0.76	0.17	0.56	0.14	<0.1	<0.1
r-5	5/5/1	1.1	1.1	0.32	0.13	0.36	0.13	0.31	0.12	<0.1	<0.1
r-6	6/6/1	>2000	>2000	0.48	0.11	0.56	0.1	0.48	<0.1	<0.1	<0.1
Rose	3/3/18	0.34	0.37	0.46	15	0.54	22	0.45	2.2	8.5	7.3
simple-nql-20-30	20/20/2	>2000	>2000	0.57	12	0.65	12	0.55	28	1.2	0.13
Takeuchi-Lu	4/4/14	<0.1	<0.1	0.27	3	0.31	3.9	0.27	0.46	1.9	1.2
Trinks-2	6/7/0	<0.1	<0.1	0.18	<0.1	0.19	<0.1	0.18	<0.1	<0.1	<0.1
Trinks-difficult	6/6/2	<0.1	<0.1	0.24	1.2	0.29	1.8	0.24	0.25	0.64	0.53
wilkinson20	1/1/21	<0.1	<0.1	0.11	0.49	0.13	0.5	0.11	0.13	0.21	0.17
wilkinsonxy	2/2/25	<0.1	<0.1	0.17	3.2	0.19	3.2	0.17	1.2	1	1

Fig. 1 Benchmark

radical='yes' ensuring these regular chains are squarefree. In order to keep things simple and uniform, the option probability=xx of Triangularize is not used. Thus the modular algorithm of [8] is not applied even though it can solve all our examples that the non-modular version 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 RealRootIsolatealgorithm 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-process 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, for some other examples, such as 5-body-homog, p3p-special and Rose, normalization makes things worse.

Comparing Strategies 1 and 3 shows two things. First, it is usually faster to isolate solutions with an infinite precision rather than with a small precision. Second, 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 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 $2 \le i \le n$ 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 (i.e., 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 $2 \le i \le n$. 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 $1 \le i \le n$. On these 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 15 components where the largest coefficient has size 20 digits. The command EquiprojectableDecomposition (which requires the use of normalized regular chains) produces 3 components for nld-9-3, where most coefficients have more than 500 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 500 digits. This makes it difficult to isolate the real roots of such polynomial. Therefore, the nld-d-n examples show that splitting can help in 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 [24]. Many improvements in 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 some non-equiprojectable varieties, our algorithm and its MAPLE implementation show favorable performances.

References

1. Akritas, A.G.: There is no Uspensky's method. In: proceedings of ISSAC (1986)
2. Akritas, A.G., Strzeboński, A.W., Vigklas, P.S.: Implementations of a new theorem for computing bounds for positive roots of polynomials. Comput. **78**(4), 355–367 (2006)
3. Akritas, A.G., Vigklas, P.S.: A comparison of various methods for computing bounds for positive roots of polynomials. J. Univ. Comput. Sci. **13**(4), 455–467 (2007)
4. Becker, E., Mora, T., Marinari, M.G., Traverso, C.: The shape of the shape lemma. In: Proceedings of the International Symposium on Symbolic and Algebraic Computation, pp. 129–133. ACM Press, New York, USA (1994)
5. Cheng, J.S., Gao, X.S., Yap, C.K.: Complete numerical isolation of real roots in zero-dimensional triangular systems. Journal of Symbolic Computation **44**(7), 768–785 (2009). http://www.sciencedirect.com/science/article/B6WM7-4TKPVBT-1/2/34927cd53d447d69e3b126d3fa93006a
6. Collins, G.E., Akritas, A.G.: Polynomial real root isolation using Descartes'rule of signs. In: proceedings of ISSAC'76, pp. 272–275. Yorktown Heights NY (1976)
7. Collins, G.E., Johnson, J.R., Krandick, W.: Interval arithmetic in cylindrical algebraic decomposition. J. Symb. Comput. **34**(2), 145–157 (2002)
8. Dahan, X., Moreno Maza, M., Schost, É., Wu, W., Xie, Y.: Lifting techniques for triangular decompositions. In: ISSAC'05, pp. 108–115. ACM Press (2005)
9. Dahan, X., Schost, É.: Sharp estimates for triangular sets. In: ISSAC 04, pp. 103–110. ACM (2004)
10. Della Dora, J., Dicrescenzo, C., Duval, D.: About a new method for computing in algebraic number fields. In: Proceedings of EUROCAL 85 Vol. 2, Lect. Notes in Comp. Sci. vol. 204, pp. 289–290. Springer (1985)
11. Descartes, R.: Géométrie (1636)
12. Eigenwillig, A., Kettner, L., Krandick, W., Mehlhorn, K., Schmitt, S., Wolpert, N.: A descartes algorithm for polynomials with bit-stream coefficients. In: CASC, vo. 3718 of LNCS, pp. 138–149. Springer (2005)
13. Faugère, J.C.: A new efficient algorithm to compute Gröbner bases (F_4). J. Pure Appl. Algebra **139**, 61–88 (1999)
14. Johnson, J.R., Krandick, W.: Polynomial real root isolation using approximate arithmetic. In: Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation, pp. 225–232. ACM (1997)
15. Leiserson, C.E., Li, L., Moreno Maza, M., Xie, Y.: Efficient evaluation of large polynomials. In: Proceedings of International Congress of Mathematical Software—ICMS 2010. Springer (2010)

16. Lemaire, F., Moreno Maza, M., Xie, Y.: The RegularChains library. In: Ilias S. Kotsireas (ed.) Maple Conference 2005, pp. 355–368 (2005)
17. Li, X., Moreno Maza, M., Pan, W.: Computations modulo regular chains. In: proceedings of ISSAC'09, pp. 239–246. ACM Press, New York, USA (2009)
18. Li, X., Moreno Maza, M., Rasheed, R., Schost, É.: The modpn library: Bringing fast polynomial arithmetic into MAPLE. In: MICA'08 (2008)
19. Lu, Z., He, B., Luo, Y., Pan, L.: An algorithm of real root isolation for polynomial systems. In: Wang, D., Zhi, L. (eds.) Proceedings of Symbolic Numeric Computation 2005, pp. 94–107 (2005)
20. McCallum, S.: An improved projection operation for cylindrical algebraic decomposition. In: Caviness, B.F., Johnson, J.R. (eds.) Quantifier Elimination and Cylindrical Algebraic Decomposition, pp. 242–268. Springer (1988)
21. Moreno Maza, M.: On triangular decompositions of algebraic varieties. Tech. Rep. TR 4, 99, NAG Ltd, Oxford, UK, : presented at the MEGA-2000 Conference. Bath, England (1999)
22. Mourrain, B., Pavone, J.P.: Subdivision methods for solving polynomial equations. Tech. rep., INRIA (Aug 2005), (number RR-5658)
23. Mourrain, B., Rouillier, F., Roy, M.F.: The bernstein basis and real root isolation. Comb. Comput. Geom., MSRI Publ. **52**, 459–478 (2005)
24. Rioboo, R.: Real algebraic closure of an ordered field, implementation in axiom. In: Proceedings of ISSAC'92, pp. 206–215. ISSAC, ACM Press (1992)
25. Rioboo, R.: Towards faster real algebraic numbers. J. Symb. Comput. **36**(3–4), 513–533 (2003)
26. Rouillier, F.: Solving zero-dimensional systems through the rational univariate representation. AAECC **9**, 433–461 (1999)
27. Rouillier, F., Revol, N.: The multiple precision floating-point interval library (MPFI) library. http://gforge.inria.fr/projects/mpfi/
28. Rouillier, F., Zimmermann, P.: Efficient isolation of polynomial real roots. J. Comput. Appl. Math. **162**(1), 33–50 (2003)
29. Serret, J.A.: Cours d'Algèbre Supérieure, 4th edn. Gauthier-Villars, Paris (1877)
30. Uspensky, J.V.: Theory of Equations. McGraw-Hill Co., New York (1948)
31. von zur Gathen, J., Gerhard, J.: Fast algorithms for taylor shifts and certain difference equations. In: ISSAC, pp. 40–47 (1997).
32. Xia, B., Yang, L.: An algorithm for isolating the real solutions of semi-algebraic systems. J. Symb. Comput. **34**(5), 461–477 (2002)
33. Xia, B., Zhang, T.: Real solution isolation using interval arithmetic. Comput. Math. Appl. **52**(6–7), 853–860 (2006)