Journal of Symbolic Computation 36 (2003) 513–533

Journal of Symbolic Computation

www.elsevier.com/locate/jsc

Towards faster real algebraic numbers

Renaud Rioboo

Laboratoire d'Informatique de Paris 6 (LIP6), Université Pierre et Marie Curie (Paris 6), 8, Rue du Capitaine Scott, 75015, Paris, France

Received 15 November 2002; accepted 10 April 2003

Abstract

This paper presents a new encoding scheme for real algebraic number manipulations which enhances current Axiom's real closure. Algebraic manipulations are performed using different instantiations of sub-resultant-like algorithms instead of Euclidean-like algorithms. We use these algorithms to compute polynomial gcds and Bezout relations, to compute the roots and the signs of algebraic numbers. This allows us to work in the ring of real algebraic integers instead of the field of real algebraic numbers avoiding many denominators. © 2003 Elsevier Ltd. All rights reserved.

Keywords: Algebraic numbers; Algebraic integers; Fractions; Real closed fields; Real closure; Sub-resultants; Fractions

1. Introduction

Real algebraic numbers are relevant for symbolic computations since they are the natural frame where computer algebra users expect solutions of polynomial systems to lie. Exact computations with real algebraic numbers are however hard to achieve and few end user packages (such as Ligatsikas et al., 1996; Strzeboński, 1997) exist for this purpose inside general purpose computer algebra systems. The real closure of Axiom which is based on algorithms of Rioboo (1992) and Ligatsikas et al. (1996) is one of the few packages that can perform non-trivial examples. This is because we avoid primitive elements and costly polynomial factorizations.

For instance, Ramanujan's example of Davenport et al. (1987):

$$\sqrt[3]{-\sqrt[5]{\frac{27}{5}}+\sqrt[5]{\frac{32}{5}}} = (-\sqrt[5]{3}^2+\sqrt[5]{3}+1)\sqrt[5]{\frac{1}{25}} \qquad (1)$$

is, to our knowledge, impossible to solve by any package but Axiom's real closure.

E-mail address: Renaud.Rioboo@lip6.fr (R. Rioboo).
URL: http://www-calfor.lip6.fr/~rr/.

Other techniques described in Duval and Gonzalez Vega (1996) are implemented in Lecerf (1996) and also avoid primitive elements. They provide real root functionalities using dynamic evaluation (see Della Dora et al., 1985) ideas and algorithms based on Coste and Roy (1988), Basu et al. (1996) and Duval and Gonzalez Vega (1996). However the `RealRoot` functionality of this package does not offer the usual arithmetic. It cannot be used as a back-end for triangular systems resolution nor as a tool for cylindrical decomposition and we did not compare with it.

This paper presents the basics for faster versions of Axiom's real closure.

1.1. Real closed fields

We recall (see Lang, 1969; Bochnak et al., 1988) that a real field is a field $\mathbf{K}$ where (-1) is not a sum of squares. An ordered field is a field $\mathbf{K}$ with a total ordering which is compatible with addition ($\forall x, y, z \in \mathbf{K} x \leq y \Rightarrow x + z \leq y + z$) and multiplication ($\forall x, y \in \mathbf{K} x \geq 0, y \geq 0 \Rightarrow xy \geq 0, \forall x x^2 \geq 0$). An ordered field is a real field and a real field admits at least an ordering turning it into an ordered field.

A real closed field is a real field which admits no strict algebraic extension which is real, it is uniquely ordered and this is equivalent to saying that this is a field where every positive number has a square root and where every odd degree polynomial has at least a root. From an effective point of view (see Ligatsikas et al., 1996) we model these properties into those of an ordered field together with an `allRootsOf` function taking a univariate polynomial and returning all its distinct roots.

1.2. Real closure

Given a computable ordered field $\mathbf{Q}$ the real closure $\tilde{\mathbf{Q}}$ of $\mathbf{Q}$ is the smallest extension field of $\mathbf{Q}$ which is real closed. It is computable (see Lombardi and Roy, 1991; Zassenhauss, 1970; Hollcott, 1941) and we use here the same scheme of towers of extensions which is described in Ligatsikas et al. (1996). This scheme allows us to manipulate real algebraic numbers encoded as pairs (γ, Q) where γ is a real algebraic variable and where Q is a univariate polynomial. In this scheme γ is a member of an external structure with its own data representation. This structure is in charge of creating new algebraic variables and computes basic operations such as checking if a univariate polynomial is zero at a real algebraic variable. This external structure is also responsible for computing the sign and the inverse of a univariate polynomial when evaluated at γ. In this scheme the only requirements for the univariate polynomials is that their coefficients are simpler (i.e. already defined). Thus their coefficients belong to the closure itself. That is, if we denote by $\tilde{\mathbf{Q}}$ the real closure of an ordered field $\mathbf{Q}$, the polynomials involved lie in $\tilde{\mathbf{Q}}[X]$.

Roughly speaking we may view an element a of $\tilde{\mathbf{Q}}$ as a tree whose leaves are elements of $\mathbf{Q}$ and whose nodes contain two elements $\mathcal{C}, \mathcal{V}$. Here $\mathcal{C}$ is interpreted as a root γ of a univariate polynomial $P_\gamma(X) \in \tilde{\mathbf{Q}}[X]$ and $\mathcal{V}$ is interpreted as a univariate polynomial $A \in \tilde{\mathbf{Q}}[X]$ representing the equation $a = A(\gamma)$. See Ligatsikas et al. (1996) for details.

$$\begin{tikzcd}
\mathbf{R} \arrow[r] \arrow[d] & \mathbf{A} \arrow[r] \arrow[d] & \tilde{\mathbf{R}} \arrow[d] \\
\mathbf{R}^{*-1}\mathbf{R} \arrow[r] \arrow[d, equal] & \mathbf{R}^{*-1}\mathbf{A} \arrow[r] \arrow[d, equal] & \mathbf{R}^{*-1}\tilde{\mathbf{R}} \arrow[d, equal] \\
\mathbf{Q} \arrow[r] & \mathbf{F} \arrow[r] & \tilde{\mathbf{Q}}
\end{tikzcd}$$

Fig. 1. The new scheme of encoding.

1.3. Algebraic integers

We recall (see Lang, 1964, for instance) that if **R** is an integral domain and if **F** is an algebraic extension of the fraction field of **R**, the algebraic integers of **F** are those elements of **F** which verify a monic polynomial relation of $\mathbf{R}[X]$.

For a sub-ring **R** of a field **K** we denote by $\mathbf{R}^*$ the set of regular (i.e. non-null) elements of **R** and if **A** is an extension ring of **R** contained in **K**. We also denote by $\mathbf{R}^{*-1}\mathbf{A}$ the sub-ring of **K** with numerators in **A** and denominators in $\mathbf{R}^*$. If **A** is an algebraic integral extension of **R** then $\mathbf{R}^{*-1}\mathbf{A}$ is a sub-field of **K**.

When managing polynomial gcds, the main advantage of algebraic integers is that they have no denominator (see however Section 4). In this paper, we propose the following scheme inspired by the real closure of Axiom. We start from a ring **R** and work over the algebraic integers $\tilde{\mathbf{R}}$ of the real closure $\tilde{\mathbf{Q}}$ of the fraction field **Q** of **R**. It is summarized in Fig. 1.

Throughout this paper, unless otherwise noted, **R** is a gcd domain which is called the base ring and **Q** is its fraction field. The ring **A** is an integral finite algebraic extension of **R** and **F** is the field $\mathbf{R}^{*-1}\mathbf{A}$ of fractions with numerators in **A** and denominators in $\mathbf{R}^*$. $\tilde{\mathbf{Q}}$ will be the real closure of **Q**, and $\tilde{\mathbf{R}}$ will be the algebraic integers of $\tilde{\mathbf{Q}}$ which is also the field of fractions with numerators in $\tilde{\mathbf{R}}$ and denominators in $\mathbf{R}^*$.

In Section 2 we introduce weak sub-resultants which enable us to compute univariate polynomial gcds. We describe the quasi sub-resultant algorithm (Algorithm 4) which extends algorithms in Moreno Maza and Rioboo (1996) and algorithms in Loos (1982).

Section 3 extends the quasi sub-resultant algorithm in order to compute the real roots of a univariate polynomial and the sign of univariate expressions depending on one root of this polynomial. We introduce quasi Sylvester sequences (Algorithm 5) which are related to algorithms in Collins and Loos (1982), Gonzalez Vega et al. (1998b,a), Basu et al. (1996) and Lickteig and Roy (2001).

Section 4 adapts the real closure construction of Ligatsikas et al. (1996) and explains the necessary localization process which is needed to compute with real algebraic integers.

Finally Section 5 gives some practical behaviour of the algorithms presented.

This paper is an extended version of a presentation at the ISSAC 2002 conference.

2. Quasi sub-resultants

In this section we present weak sub-resultant algorithms which are efficient when working over algebraic extensions of rings. These algorithms enable us to compute gcds, Bezout relations, or real roots of polynomials.

2.1. Quasi remainders

Let P and Q be two non-constant polynomials of $\mathbf{A}[X]$ and f be a function from $\mathbf{A}^*$ to $\mathbf{A}^*$. For $a \in \mathbf{A}^*$ we denote $f(a)$ by $\overline{a}^f$ and the product $f(a)a$ by $\|a\|_f$. We often call $\overline{a}^f$ the f-pseudo inverse of a and $\|a\|_f$ the f-pseudo norm of a. We call the f-normalized of P the polynomial $\|P\|_f = \overline{\mathrm{lc}(P)}^f P$ where $\mathrm{lc}(P)$ denotes the leading coefficient of P. We thus have $\|P\|_f = \|p\|_f X^{|P|} + \overline{p}^f P'$ for $P = pX^{|P|} + P'$ with $|P'| < |P|$ denoting by $|P|$ the degree of a polynomial P.

We define $\mathrm{qrem}(P, Q, f)$ the f-quasi remainder of P by Q to be the pseudo remainder of P by $\|Q\|_f$. The f-quasi remainder of P by Q thus verifies:

$$\mathrm{qrem}(P, Q, f) = \mathrm{prem}(P, \overline{Q}^f Q) = (\overline{\mathrm{lc}(Q)}^f)^{\max(1+|P|-|Q|,0)}\mathrm{prem}(P, Q).$$

The main advantage of quasi remainders over pseudo remainders is that the relation between quasi remainders can be kept with (experimentally) “smaller” coefficients. Let R and K be the pseudo remainder and the pseudo quotient of P by Q, and R' and K' be the f-quasi remainder and quasi quotient of P by Q, we have

$$q^{\delta+1}P = KQ + R$$

when $|P| \geq |Q|$ with $q = \mathrm{lc}(Q)$ and $\delta = |P| - |Q|$. Whereas we have under the same assumptions

$$\|q\|_f^{\delta+1}P = K'Q + R'$$

and if $\|q\|_f$ is “simpler” than q the division is easier to perform in practice. This is the scheme of Moreno Maza and Rioboo (1996) and we see that f-pseudo inverses enable us to compute f-quasi remainders as pseudo remainders by f-normalized polynomials.

Example 1. Let $P = X^3 + 1$ and $Q = \sqrt{2}X + 1$, the pseudo remainder $\mathrm{prem}(P, Q)$ is $2\sqrt{2} - 1$. For $f(x) = \sqrt{2}x$, the quasi remainder $\mathrm{qrem}(P, Q, f)$ is $8 - 2\sqrt{2}$.

2.2. Weak sub-resultants

Sub-resultants are widely discussed in computer algebra literature. For instance, Loos (1982), Basu et al. (1996), Ducos (2000), Lombardi et al. (2000) and Lickteig and Roy (2001) give their definitions and properties. We are more interested in computing univariate polynomial gcds, Bezout relations and Sturm-like sequences than in algorithms which compute the resultant. Our motivation is to obtain efficient algorithms for manipulating real algebraic numbers. We thus concentrate on the different values produced during computations and we want them to be easy to compute. We concentrate on sub-resultant algorithms because they have the advantage to introduce simplifications by predicting divisors. For the case of integer coefficients this entails that coefficients are made shorter without computing gcds.

We analyse here the inner loop of the sub-resultant algorithm to compute the resultant.

Algorithm 1. We can define the sub-resultant algorithm in terms of three operations: `nextAlpha`, `nextDefective` and `nextNonDefective`. Using Axiom-like syntax we have:

```
generalResultant ($P, Q, \alpha, \psi$) =
 $Q = 0 \Rightarrow$
  $|P| > 0 \Rightarrow 0$
  $\psi$
 $Q' \leftarrow$ nextNonDefective ($Q, \alpha, \psi, |P| - |Q|$)
 $R \leftarrow$ nextDefective ($P, Q, \alpha, \psi$)
 generalResultant ($Q, R,$ nextAlpha ($\mathrm{lc}(Q)$), $\mathrm{lc}(Q')$)
```

Of course, this algorithm must be modified appropriately if one wants to compute polynomial gcds, all sub-resultants or an extended version of the algorithm which computes the cofactors of Bezout relation.

Algorithm 2. We obtain the classical sub-resultant algorithm of Loos (1982) by taking:

$$\texttt{nextAlpha}\,(q) = q,$$

$$\texttt{nextNonDefective}\,(Q, \alpha, \psi, \delta) = \frac{\alpha^{\delta-1}Q}{\psi^{\delta-1}},$$

$$\texttt{nextDefective}\,(P, Q, \alpha, \psi) = \frac{\mathrm{prem}(P, Q)}{-\alpha(-\psi)^{|P|-|Q|}}$$

and initializing α and ψ to 1 in Algorithm 1.

Indeed, let us denote by F_{i-1}, F_i, α_{i-1}, ψ_{i-1} the values passed to the function of Algorithm 1 for the parameters P, Q, α and ψ. We denote by δ_i the difference of degrees $|F_{i-1}| - |F_i|$. Let us assume that F_i is not zero, by the definition of parameter R in Algorithm 1 and the definition of Algorithm 2 we have a pseudo division:

$$\mathrm{lc}(F_i)^{\delta_i+1}F_{i-1} = K_i F_i - \alpha_{i-1}(-\psi_{i-1})^{\delta_i} F_{i+1},$$

which is the sub-resultant pseudo division relation (2).

From Algorithm 2 we obviously see that $\alpha_i = \mathrm{lc}(F_i)$ and by the definition of parameter Q' in Algorithm 1, we see that:

$$S_i = \frac{\alpha_{i-1}^{\delta_i-1}F_i}{\psi_{i-1}^{\delta_i-1}},$$

from which we can deduce sub-resultant relation (3). Now when Q is null the function of Algorithm 1 returns the value ψ_{i-1} and for two polynomials P and Q with $|P| \geq |Q|$, the call `generalResultant` $(P, Q, 1, 1)$ returns the resultant of P and Q.

Example 2. Let P be $X^3 + (\sqrt{2} + \sqrt{3})X^2 + (3\sqrt{2} + 2\sqrt{3})X + 1$ and Q be its derivative in X, the sub-resultant sequence of P and Q is:

$$\begin{aligned}
F_0 &= X^3 + (\sqrt{3} + \sqrt{2})X^2 + (2\sqrt{3} + 3\sqrt{2})X + 1\\
F_1 &= 3X^2 + (2\sqrt{3} + 2\sqrt{2})X + 2\sqrt{3} + 3\sqrt{2}\\
F_2 &= ((-4\sqrt{2} + 12)\sqrt{3} + 18\sqrt{2} - 10)X - 5\sqrt{2}\sqrt{3} - 3\\
F_3 &= (-210\sqrt{2} + 564)\sqrt{3} + 692\sqrt{2} - 483.
\end{aligned}$$

Remark 1. In Example 2, the last remainder computed is:

$$(-210\sqrt{2} + 564)\sqrt{3} + 692\sqrt{2} - 483.$$

Algorithm 3. For any function f from $\mathbf{A}^*$ to $\mathbf{A}^*$ we obtain the algorithm `NewSubResGcd` of Moreno Maza and Rioboo (1996) by taking:

$$\begin{aligned}
&\texttt{nextAlpha}\,(q) = \|q\|_f,\\
&\texttt{nextNonDefective}\,(Q, \alpha, \psi, \delta) = \frac{\alpha^{\delta-1}\|Q\|_f}{\psi^{\delta-1}},\\
&\texttt{nextDefective}\,(P, Q, \alpha, \psi) = \frac{\mathrm{prem}(\|P\|_f, \|Q\|_f)}{-\alpha(-\psi)^{|P|-|Q|}}
\end{aligned}$$

and initializing α and ψ to 1 in Algorithm 1. Again denoting $f(a) = \overline{a}^f$, $f(a)a = \|a\|_f$ and $\|Q\|_f = \overline{\mathrm{lc}(Q)}^f Q$.

This algorithm specializes to the sub-resultant algorithm when taking $\overline{a}^f = 1$ (and thus $\|a\|_f = a$ and $\|Q\|_f = Q$). When $\mathbf{A}$ is a field the algorithm specializes to the Euclidean primitive gcd algorithm by taking $\overline{a}^f = 1/a$ (and thus $\|a\|_f = 1$ and $\|Q\|_f$ is monic and simliar to Q).

Example 3. For the polynomials of Example 2, the primitive Euclidean gcd algorithm computes the following terms:

$$\begin{aligned}
F_0 &= X^3 + (\sqrt{3} + \sqrt{2})X^2 + (2\sqrt{3} + 3\sqrt{2})X + 1\\
F_1 &= X^2 + \left(\frac{2}{3}\sqrt{3} + \frac{2}{3}\sqrt{2}\right) X + \frac{2}{3}\sqrt{3} + \sqrt{2}\\
F_2 &= X + \left(-\frac{251}{4754}\sqrt{2} - \frac{1638}{2377}\right)\sqrt{3} + \frac{2655}{4754}\sqrt{2} - \frac{207}{4754}\\
F_3 &= 1.
\end{aligned}$$

Remark 2. In this example the last remainder computed is:

$$\left(-\frac{7661\,745}{11\,300\,258}\sqrt{2} + \frac{4001\,178}{5650\,129}\right)\sqrt{3} + \frac{14\,738\,781}{11\,300\,258}\sqrt{2} + \frac{61\,034\,481}{22\,600\,516}.$$

2.3. Weak quotients and divisors

In Example 2 the leading coefficients are algebraic numbers. So, pseudo divisions in Algorithm 2 involve multiplication by algebraic numbers. Subsequent simplifications involve division of two algebraic numbers.

We can constrain Algorithm 1 to make "simple" divisions by providing a function f that maps $a \in \mathbf{A}$ to $\overline{a}^f$ in $\mathbf{A}$, such that the product $a\overline{a}^f = \|a\|_f$ lies in a sub-ring $\mathbf{R}$ of $\mathbf{A}$. This is always possible if $\mathbf{A}$ is an integral finite algebraic extension of $\mathbf{R}$. Indeed, let $a \in \mathbf{A}$ and P_a be the minimal polynomial of a. If a is not zero, P_a can be written as $p_0 + X Q_a(X)$, where p_0 and Q_a are both non-zero and relation $p_0 = -a Q_a(a)$ holds in $\mathbf{A}$.

In practice, the function f will remain fixed through-out the process. For $a \in \mathbf{A}$, we let $\overline{a}$ be $\overline{a}^f$ be the pseudo inverse of a and $\|a\|$ be $\|a\|_f$ be the pseudo norm of a. In general we will rely on a function `conjNorm` that returns both terms $(\overline{a}, \|a\|) \in \mathbf{A} \times \mathbf{R}$ for an element a of $\mathbf{A}$.

Example 4. As in Moreno Maza and Rioboo (1996), Algorithm 3 uses pseudo inverses and computes the following terms:

$$\begin{aligned}
F_0 &= X^3 + (\sqrt{3} + \sqrt{2})X^2 + (2\sqrt{3} + 3\sqrt{2})X + 1\\
F_1 &= 3X^2 + (2\sqrt{3} + 2\sqrt{2})X + 2\sqrt{3} + 3\sqrt{2}\\
F_2 &= 4754X + \left(-251\sqrt{2} - 3276\right)\sqrt{3} + 2655\sqrt{2} - 207\\
F_3 &= 506\,595\,634\,305\,713.
\end{aligned}$$

for the polynomials P and Q of Example 2.

Remark 3. In this example, the last remainder computed is:

$$(-5107\,830\sqrt{2} + 5334\,904)\sqrt{3} + 9825\,854\sqrt{2} + 20\,344\,827.$$

Of course, pairs verifying $a\overline{a} = \|a\|$ are not unique and we want to maintain both terms of the pair as simple as possible. Thus, beyond the possibility to divide an element of $\mathbf{A}$ by an element of $\mathbf{R}$, we need a `gcd` function taking as argument a pair of $\mathbf{A} \times \mathbf{R}$ and returning an element of $\mathbf{R}$ which divides both of its arguments. We will require the base ring R to be a gcd domain.

We now recall classical sub-resultant relations in the sequence computed by Algorithm 2.

For two polynomials P and Q, we will denote by F_i the polynomials of the sub-resultant sequence of P and Q as computed by Algorithm 2. We will let f_i be the leading coefficient of F_i. Other successive parameters in sub-resultant Algorithm 2 will be denoted by α_i, and ψ_i.

We have a pseudo division relation:

$$f_i^{\delta_i+1} F_{i-1} = K_i F_i - \alpha_{i-1}(-\psi_{i-1})^{\delta_i} F_{i+1}. \tag{2}$$

Here K_i is the pseudo quotient and $-\alpha_{i-1}(-\psi_{i-1})^{\delta_i} F_{i+1}$ is the pseudo remainder of the pseudo division of F_{i-1} by F_i. We start with $\alpha_0 = \psi_0 = 1$ and have:

$$\begin{cases} \alpha_{i+1} = f_{i+1} \\ \psi_{i+1} = \frac{\alpha_{i+1}^{\delta_i}}{\psi_i^{\delta_i - 1}}. \end{cases} \tag{3}$$

We are now ready to generalize Algorithm 3.

2.4. Quasi sub-resultants

We include here full proofs of the algorithms. This is both for completeness and since these cannot be easily deduced from Moreno Maza and Rioboo (1996). The formulation of Algorithm 1 is a direct consequence of this section.

A simple remark is that we can take quasi remainders instead of pseudo remainders in Algorithm 1 when f is a multiplicative morphism from $\mathbf{A}^*$ to $\mathbf{A}^*$.

Algorithm 4. For a multiplicative morphism f of $\mathbf{A}^*$ and for any function g from $\mathbf{A}^*$ to $\mathbf{A}^*$, returning a divisor of its argument. We define the `QuasiSubResultant` algorithm by selecting:

$$\begin{aligned}
&\texttt{nextAlpha}\,(q) = g(q),\\
&\texttt{nextNonDefective}\,(Q, \alpha, \psi, \delta) = \frac{\alpha^{\delta-1}\overline{\mathrm{lc}(Q)}^{f} Q}{\psi^{\delta-1}},\\
&\texttt{nextDefective}\,(P, Q, \alpha, \psi) = \frac{\mathrm{qrem}(P, Q, f)}{-\alpha(-\psi)^{|P|-|Q|}}
\end{aligned}$$

and initializing α and ψ to 1 in Algorithm 1. As usual $f(a) = \overline{a}^{f}$ and $f(a)a = \|a\|_f$.

This algorithm also specializes to the sub-resultant algorithm when taking $\overline{a}^{f} = 1$ (and thus $\|a\|_f = a$ and $\|Q\|_f = Q$) and $g(a) = a$. But when $\mathbf{A}$ is a field, the quasi sub-resultant algorithm specializes to the Euclidean gcd algorithm by taking $\overline{a}^{f} = 1/a$ (and thus $\|a\|_f = 1$ and $\|Q\|_f$ is the monic polynomial similar to Q) and by taking $g(a) = 1$. If we let g be the identity function, Algorithm 4 specializes to Algorithm 3.

In practice, this means only that we need to compute $\|Q\|_f$ to perform the pseudo division and do not need to remember it after.

Example 5. For the polynomials P and Q of Example 2, the Euclidean remainder sequence of P and Q is:

$$\begin{aligned}
F_0 &= X^3 + (\sqrt{3} + \sqrt{2})X^2 + (2\sqrt{3} + 3\sqrt{2})X + 1\\
F_1 &= 3X^2 + (2\sqrt{3} + 2\sqrt{2})X + 2\sqrt{3} + 3\sqrt{2}\\
F_2 &= \left(\left(-\frac{4}{9}\sqrt{2} + \frac{4}{3}\right)\sqrt{3} + 2\sqrt{2} - \frac{10}{9}\right) X - \frac{5}{9}\sqrt{2}\sqrt{3} - \frac{1}{3}\\
F_3 &= \left(-\frac{22\,985\,235}{11\,300\,258}\sqrt{2} + \frac{12\,003\,534}{5650\,129}\right)\sqrt{3} + \frac{44\,216\,343}{11\,300\,258}\sqrt{2} + \frac{183\,103\,443}{22\,600\,516}.
\end{aligned}$$

Proposition 1. *Let $\mathbf{A}$ be an integral domain. Let f be a multiplicative morphism from $\mathbf{A}^*$ to $\mathbf{A}^*$ and g be any function from $\mathbf{A}^*$ to $\mathbf{A}^*$, returning a divisor of its argument. Let F_i' be* *the f, g-quasi sub-resultant sequence of two polynomials P and Q of $\mathbf{A}[X]$ as computed by Algorithm 4. The coefficients of F_i' remain in $\mathbf{A}[X]$.*

We denote by f_i' the leading coefficient of F_i'. Let α_i', and ψ_i' be the successive parameters in quasi sub-resultant Algorithm 4. For simplicity, we denote by $\overline{a} = f(a)$ and $\|a\| = f(a)a$.

We have a pseudo division relation:

$$\| f_i' \|^{\delta_i+1} F_{i-1}' = K \overline{f_i'} F_i' - \alpha_{i-1}' (-\psi_{i-1}')^{\delta_i} F_{i+1}' \tag{4}$$

with

$$F_{i+1}' = \text{qrem}(F_{i-1}', F_i', f) = \text{prem}(F_{i-1}', \overline{f_i'} F_i').$$

We start with $\alpha_0' = 1$, $\psi_0' = 1$ and

$$\begin{cases} \alpha_{i+1}' = g(f_{i+1}') \\ \psi_{i+1}' = \frac{\|\alpha_{i+1}'\|^{\delta_i}}{\psi_i'^{\delta_i - 1}}. \end{cases} \tag{5}$$

Following Moreno Maza and Rioboo (1996), we write $F_i' = \mu_i F_i$ and relations are to be stated in the fraction field of $\mathbf{A}$. We have to prove that μ_i remains in $\mathbf{A}$.

Let us examine the first terms, since $\alpha_0' = 1$ and $\psi_0' = 1$ we have:

$$F_2' = -\text{qrem}\,(F_0, F_1) = -\text{prem}\,(F_0', \overline{f_1'} F_1')$$
$$F_2' = -\overline{f_1'}^{\delta_1+1} \text{prem}\,(F_0', F_1') = -\overline{f_1'}^{\delta_1+1} \text{prem}(F_0, F_1)$$

since $F_0 = F_0'$ and $F_1 = F_1'$ and thus $\mu_0 = \mu_1 = 1$. Now,

$$\mu_2 F_2 = \overline{f_1}^{\delta_1+1} \alpha_0 \psi_0^{\delta_1} F_2 = \overline{f_1}^{\delta_1+1} F_2.$$

We thus see that μ_2 lies in $\mathbf{A}$. Let us now examine further terms, we have:

$$\begin{aligned} \alpha_{i-1}' \psi_{i-1}'^{\delta_i} F_{i+1}' &= (-1)^{\delta_i+1} \text{qrem}(F_{i-1}', F_i') \\ &= (-1)^{\delta_i+1} \text{prem}(\mu_{i-1} F_{i-1}, \overline{f_i'} \mu_i F_i) \\ \alpha_{i-1}' \psi_{i-1}'^{\delta_i} \mu_{i+1} F_{i+1} &= (-1)^{\delta_i+1} \mu_{i-1} [\overline{f_i'} \mu_i]^{\delta_i+1} \text{prem}(F_{i-1}, F_i) \\ &= \mu_{i-1} \overline{f_i'} \mu_i \alpha_{i-1} [\overline{f_i'} \mu_i \psi_{i-1}]^{\delta_i} F_{i+1} \end{aligned}$$

and thus

$$\alpha_{i-1}' \psi_{i-1}'^{\delta_i} \mu_{i+1} = \mu_{i-1} \overline{f_i'} \mu_i \alpha_{i-1} [\overline{f_i'} \mu_i \psi_{i-1}]^{\delta_i}. \tag{6}$$

But, $f_i' = \mu_i f_i$ and thus $\overline{f_i'} = \overline{\mu_i}\,\overline{f_i}$. Now (6) becomes:

$$\alpha_{i-1}' \psi_{i-1}'^{\delta_i} \mu_{i+1} = \mu_{i-1} \overline{\mu_i}\,\overline{f_i} \mu_i \alpha_{i-1} [\overline{\mu_i}\,\overline{f_i} \mu_i \psi_{i-1}]^{\delta_i} \tag{7}$$

and, as in Moreno Maza and Rioboo (1996), we write this in the form:

$$\frac{\mu_{i+1}}{\mu_i} = \frac{\mu_{i-1} \alpha_{i-1} \overline{\mu_i}\,\overline{f_i}}{\alpha_{i-1}'} \left[\frac{\overline{\mu_i}\,\overline{f_i} \mu_i \psi_{i-1}}{\psi_{i-1}'} \right]^{\delta_i} \tag{8}$$

and we put $i \to i+1$ to obtain:

$$\frac{\mu_{i+2}}{\mu_{i+1}} = \frac{\mu_i\alpha_i\overline{\mu_{i+1}}\,\overline{f_{i+1}}}{\alpha_i'}\left[\frac{\overline{\mu_{i+1}}\,\overline{f_{i+1}}\mu_{i+1}\psi_i}{\psi_i'}\right]^{\delta_{i+1}}. \tag{9}$$

From sub-resultant relation (3) we have:

$$\psi_{i+1}\psi_i^{\delta_{i+1}-1} = \alpha_{i+1}^{\delta_{i+1}} \tag{10}$$

and from (9) multiplied by (10), we have:

$$\frac{\mu_{i+2}\psi_{i+1}\psi_i^{\delta_{i+1}-1}}{\mu_{i+1}} = \frac{\mu_i\alpha_i\overline{\mu_{i+1}}\,\overline{f_{i+1}}}{\alpha_i'}\left[\frac{\overline{\mu_{i+1}}\,\overline{f_{i+1}}\mu_{i+1}\alpha_{i+1}}{\psi_i'}\right]^{\delta_{i+1}}\psi_i^{\delta_{i+1}}$$

$$\frac{\mu_{i+2}\psi_{i+1}}{\mu_{i+1}\psi_i} = \frac{\mu_i\alpha_i\overline{\mu_{i+1}}\,\overline{f_{i+1}}}{\alpha_i'}\left[\frac{\|\mu_{i+1}\|\,\overline{f_{i+1}}\alpha_{i+1}}{\psi_i'}\right]^{\delta_{i+1}}$$

dividing both sides by ψ'_{i+1} gives:

$$\frac{\mu_{i+2}\psi_{i+1}}{\psi'_{i+1}} = \frac{\mu_{i+1}\psi_i}{\psi_i'}\frac{\mu_i\alpha_i}{\alpha_i'}\overline{\mu_{i+1}}\,\overline{f_{i+1}}\frac{(\|\mu_{i+1}\|\,\overline{f_{i+1}}\alpha_{i+1})^{\delta_{i+1}}}{\psi_i'^{\delta_{i+1}-1}\psi'_{i+1}}.$$

From quasi sub-resultant relation (5), we see that the latter rewrites in:

$$\frac{\mu_{i+2}\psi_{i+1}}{\psi'_{i+1}} = \frac{\mu_{i+1}\psi_i}{\psi_i'}\frac{\mu_i\alpha_i}{\alpha_i'}(\overline{\mu_{i+1}}\,\overline{f_{i+1}})^{\delta_{i+1}+1}\left[\frac{\mu_{i+1}\alpha_{i+1}}{\alpha'_{i+1}}\right]^{\delta_{i+1}}. \tag{11}$$

We now, re-induce relation $\mu_i f_i = f_i'$ and $\alpha_i = f_i$ to obtain:

$$\frac{\mu_{i+2}\psi_{i+1}}{\psi'_{i+1}} = \overline{f'_{i+1}}\frac{f_i'}{\alpha_i'}\left[\frac{f'_{i+1}}{\alpha'_{i+1}}\right]^{\delta_{i+1}}\frac{\mu_{i+1}\psi_i}{\psi_i'}.$$

Since f_i'/α_i' is in the ring **A**, we see that the sequence $(\mu_{i+1}\psi_i)/\psi_i'$ has coefficients in **A**. Relation (8) shows that μ_{i+1}/μ_i is in **A**.

This shows that the sequence μ_i has coefficients in **A** and thus that $f_i' = \mu_i f_i$ also has coefficients in **A**.

Example 6. As in this paper, the pseudo inverse function is a multiplicative morphism. We take for $g(q)$ the function that returns a common divisor (in **R**) of q and $\|q\|$. For the polynomials P and Q of Example 2, Algorithm 4 computes the following terms:

$$F_0 = X^3 + (\sqrt{3}+\sqrt{2})X^2 + (2\sqrt{3}+3\sqrt{2})X + 1$$
$$F_1 = 3X^2 + (2\sqrt{3}+2\sqrt{2})X + 2\sqrt{3} + 3\sqrt{2}$$
$$F_2 = ((-4\sqrt{2}+12)\sqrt{3} + 18\sqrt{2} - 10)X - 5\sqrt{2}\sqrt{3} - 3$$
$$F_3 = (-5107\,830\sqrt{2} + 5334\,904)\sqrt{3} + 9825\,854\sqrt{2} + 20\,344\,827.$$

Remark 4. In this example the last remainder is the same as in Remark 3.

3. Real algebraic integers

In this section we present generalizations of previous algorithms which have good properties for the interpretation of their result in an ordered ring.

3.1. Sign computations

One advantage of Algorithm 4 is that we have been able to keep the relation between terms rather simple in Eq. (4):

$$\| f_i' \|^{\delta_i+1} F_{i-1}' = K\overline{f_i'} F_i' - \alpha_i' \psi_i^{\delta_i} F_{i+1}'$$

and the $\| f_i' \|$, α_i' and the ψ_i' remain in the base ring **R**. We can define a Sylvester-like sequence by simply requiring that the coefficients in Eq. (4) are of opposite signs. This has the advantage that signs in **R** are easier to compute than signs in **A**. We can state an analogy of Algorithm 4.

Algorithm 5. Let P and Q be two polynomials of $\mathbf{A}[X]$. Let f be a morphism taking an element a of $\mathbf{A}^*$ and returning an element $\overline{a}^f$ of $\mathbf{A}^*$ such that $\overline{a}^f a = \|a\|_f$ with $\|a\|_f > 0$ (in **R**). Let g be a function taking an element a of $\mathbf{A}^*$ and returning a positive (in **R**) divisor of a. The f, g-quasi Sylvester sequence of P and Q is obtained by selecting

$$\texttt{nextAlpha}\,(q) = g(q),$$
$$\texttt{nextDefective}\,(P, Q, \alpha, \psi) = \frac{-\mathrm{qrem}(P, Q, f)}{\alpha \psi^{|P|-|Q|}},$$
$$\texttt{nextNonDefective}\,(P, \alpha, \psi, \delta) = \frac{(\|\mathrm{lc}(Q)\|_f)^{\delta-1} \|Q\|_f}{\psi^{\delta-1}}$$

and initializing α and ψ to 1 in Algorithm 1.

This algorithm specializes to the Sylvester algorithm (see Basu et al., 1996) by taking $\overline{a}^f = 1/a$ (and thus $\|a\|_f = 1$ and $\|Q\|_f$ is the primitive part of Q) and $g(a) = 1$ when **A** is a field. Algorithm 5 specializes to the negative remainder sequence (see Loos, 1982) by taking $\overline{a}^f = \mathrm{sign}(a)$ (and thus $\|a\|$ is the absolute value of a) and $g(a) = a$.

Since the quasi Sylvester sequence differs only by signs from the quasi sub-resultant sequence its computation can be carried out in the ring **A** of coefficients of the input polynomials.

Let **K** be a a real closed field, **R** be a sub-ring of **K** and **A** be an integral algebraic extension of **R** contained in **K**. Let $\{F_i\}_{i=0}^{i=k}$ be the quasi Sylvester sequence of P and Q in $\mathbf{A}[X]$. Let us assume that $|F_k| = 0$ and $F_k \neq 0$ and let x be a root of some F_i for $i > 0$ then $F_{i-1}(x)$ and $F_{i+1}(x)$ are of opposite signs and regardless of the sign of F_i in a neighbourhood of x, F_{i-1} and F_{i+1} remain of opposite sign in this neighbourhood.

Let $S = \{F_i\}_{i=0}^{i=k}$ be a sequence of polynomials of $\mathbf{A}[X]$ such that F_k is a non-null constant polynomial. Following Basu et al. (1996) and Rioboo (1992), we define the sign variations of S at a point x of **K** as being the number of sign variations of $S' = F_{i_0}(x), F_{i_1}(x), \ldots, F_{i_l}(x)$ with $i_0 = 0$ and where $F_{i_{j+1}}$ is the first polynomial in the sequence $F_{i_j+1} \ldots F_k$ that does not vanish at x.

(a)	x_{i-}	x_i	x_{i+}	(b)	x_{i-}	x_i	x_{i+}
F_{i-1}	$+$	$+$	$+$	F_{i-1}	$-$	$-$	$-$
F_i	$\pm$	0	$\pm$	F_i	$\pm$	0	$\pm$
F_{i+1}	$-$	$-$	$-$	F_{i+1}	$+$	$+$	$+$

Fig. 2. Local behaviour of Sylvester sequence.

Let P and Q be two polynomials with no common roots in $\mathbf{K}$ and let $\{F_i\}_{i=0}^{i=k}$ be the quasi Sylvester sequence of P and Q. As noted before (see Fig. 2), the sign variation of S does not change when crossing a point x which is a root of some F_i, with $i \geq 1$, but only changes when crossing a root x of P.

Basu et al. (1996) proposes to compute the sign of an expression Q at a root of P by computing the sign variations of the Sylvester sequence of P and $P'Q$ where P' denotes the derivative of P. Even when replacing Sylvester sequences with quasi Sylvester sequences this technique has the drawback that sparsity is almost always lost in the process. For instance if Q is a simple expression of degree 1, the Sylvester sequence of P and $P'Q \bmod P$ usually has $|P|$ terms. On the contrary the Sylvester sequence of P and Q only has three terms.

Example 7. The polynomial P of Example 2 has one single root. In order to compute its sign, Basu et al. (1996) will compute the Sylvester sequence of P and $XQ \bmod P$ which is $(-\sqrt{3} - \sqrt{2})X^2 + (-4\sqrt{3} - 6\sqrt{2})X - 3$ whereas we would compute the Sylvester sequence of P and X.

We propose an alternate method which takes advantage of the fact that, when we distinguish the distinct roots of a square free polynomial P by an interval (a, b) containing one single root of P, the sign of P is known in (a, b). Since the sign variations of the quasi Sylvester sequence can only change at a root x_i of a square free polynomial P we can write the variation table of Fig. 2 at the vicinity of a root x_i of P.

We see that in cases (a) and (b) the sign of Q at x is given by $V(x_-) - V(x_+)$ and in cases (c) and (d) it is given by $V(x_+) - V(x_-)$. If we select the interval (a, b) to be such that $P_\alpha(b) \neq 0$ we are able to include the case where P_α vanishes at a and we can thus work with left closed right open intervals. This is the reason why our definition of sign variations slightly differs from the usual definitions which never count zeros.

Proposition 2. *Let $\mathbf{K}$ be a real closed field, let $\mathbf{R}$ be a sub-ring of $\mathbf{K}$ and let $\mathbf{Q}$ be its fraction field. Let $\mathbf{A}$ be an integral algebraic extension of $\mathbf{R}$ and $\mathbf{F}$ be the field $\mathbf{R}^{*-1}\mathbf{A}$. Let $[a, b[$ be a left open, right closed interval of $\mathbf{K}$ with a and b lying in $\mathbf{Q}$. Let $P(X)$ be a square free polynomial of $\mathbf{A}[X]$ such that $P(b) \neq 0$ (in $\mathbf{K}$) and such that $[a, b[$ contains only one root α of P. Let $Q(X)$ be a polynomial of $\mathbf{A}[X]$ such that P and Q have no common root over $\mathbf{K}$. Let S be the quasi Sylvester sequence of P and Q. Let $V(S, a)$ and $V(S, b)$ be the number of sign variations of S at a and b.*

(a)	x_-	x	x_+
P	$-$	0	$+$
Q	$+$	$+$	$+$
V	v	v	$v-1$

(b)	x_-	x	x_+
P	$-$	0	$+$
Q	$-$	$-$	$-$
V	v	$v+1$	$v+1$

(c)	x_-	x	x_+
P	$+$	0	$-$
Q	$+$	$+$	$+$
V	v	$v+1$	$v+1$

(d)	x_-	x	x_+
P	$+$	0	$-$
Q	$-$	$-$	$-$
V	v	v	$v-1$

Fig. 3. Initial variation of Sylvester sequence.

If $P(b)$ is positive in **K** *then $V(S, a) - V(S, b)$ gives the sign of $Q(\alpha)$ in* **K** *and if $P(b)$ is negative then $V(S, b) - V(S, a)$ gives the sign of $Q(\alpha)$ in* **K**.

We will now summarize the advantages of Proposition 2 over commonly used techniques.

- Many algorithms work by "refining" an isolating interval for an algebraic number. This is the case for instance when using Descartes's rule of sign in Rioboo (1992) or Ligatsikas et al. (1996). This can be a long process in certain special cases, we now completely avoid these refinements.
- The quasi Sylvester sequence is faster to compute than the general negative remainder sequences of Collins and Loos (1982) since in quasi sub-resultant Eq. (4):

  $$\| f_i' \|_f^{\delta_i+1} F'_{i-1} = K\, \overline{f_i'}^{\,f} F'_i - \alpha'_i \psi'^{\delta_i}_i F'_{i+1},$$

  proportionality coefficients are kept in the base ring **R**. Signs in **R** are faster to compute than in the algebraic extension **A** where lie the coefficients of the F_i's.
- Sturm Habicht sequences of Basu et al. (1996), Gonzalez Vega et al. (1998a) and Lickteig and Roy (2001) do not require to compute signs in **A** but are still based on straight sub-resultant Eq. (2):

  $$f_i^{\delta_i+1} F_{i-1} = K_i F_i - \alpha_i \psi_i^{\delta_i} F_{i+1}.$$

  Simplifications of pseudo remainders in sub-resultant computations involve divisions performed in $\mathbf{A} \times \mathbf{A}$. We perform those simplifications using divisions in $\mathbf{A} \times \mathbf{R}$ and they are easier to perform in practice.

Remark 5. If we look again at Fig. 2, we see that we can avoid sign computations in some cases. In particular for a list of length 3, we can sometimes compute its sign variation using only two sign computations. We thus can design a method to compute the sign variations of a list of numbers which steps elements by two instead of by one. Our experiments show that we avoid 20–30% of the sign computations.

3.2. Root finding

Since quasi Sylvester sequences have the same properties as Sylvester sequences, one can define the quasi Sturm sequence of a polynomial P of $\mathbf{A}[X]$ to be the quasi Sylvester sequence of P and its derivative P'. Cases (a) and (d) of Eq. (3) can then be used to count the number of roots of a square free polynomial. Starting with an interval containing all the roots of P, it is possible to refine it into subintervals containing one single root of P. Since we choose to count the first 0 of a sequence as a sign, we are able to count the number of roots in a left closed right open interval.

4. Localization

4.1. Internal localization

In previous sections intervals have bounds in the fraction field $\mathbf{Q}$ of the base ring $\mathbf{R}$ and we need to evaluate a polynomial of $\mathbf{A}[X]$ at points of $\mathbf{Q}$ with result in $\mathbf{F}$. In practice we return the result as a fraction

$$P(a_n/a_d) = \frac{P^{\tilde{a_d}}(a_n)}{a_d^{|P|}},$$

where $P^{\tilde{a}} = a^{|P|}P(X/a)$ can be obtained remaining in $\mathbf{A}$:

$$\begin{cases} 0^{\tilde{a}} = 0 \\ (p_n X^n + P_r(X))^{\tilde{a}} = p_n X^n + a^{n-|P_r|} P_r^{\tilde{a}}(X). \end{cases} \tag{12}$$

Note that these fractions can better be stored as triples (a_n, a_d, k) for the fraction a_n/a_d^k. We thus store the denominator a_d^k in a compact form. Since very little arithmetic is done with these fractions we can assume denominators are positive and we simply return the denominator since we only need its sign.

However there are cases where more complicated fractions must be used, in particular the quasi sub-resultant algorithm returns a pseudo divisor of its input polynomials, and though both of its arguments may be monic polynomials, there is no reason for the result to be a monic polynomial. We cannot even assume that it is similar to a monic polynomial.

Example 8. One can consider the polynomials $X^2 - X - 1$ and $X^2 - \sqrt{5}X + 1$ with the coefficient ring being $\mathbf{A} = \mathbb{Z}\left[\sqrt{\ntrianglelefteq}\right]$, the quasi gcd for these polynomials is $2X - 1 - \sqrt{5}$ which cannot be made monic in $\mathbf{A}[X]$, since it is not true that $1 + \sqrt{5}$ is a multiple of 2 in $\mathbb{Z}\left[\sqrt{\ntrianglelefteq}\right]$.

To tackle this problem, in Moreno Maza and Rioboo (1996), we proposed a global localization process. We want here to take advantage that we are building an integral extension of **R** and that defining polynomials can be kept monic with real algebraic coefficients as shown in Fig. 1.

We will encode algebraic expressions as triples:

$$(\gamma, Q, d),$$

where γ/d is an algebraic integer, the univariate polynomial Q will be taken to have coefficients in the closure and the value (in $\tilde{\mathbf{Q}}$) which is encoded by this triple is $Q(\gamma/d)$.

The $(r, Q) \to \tilde{Q^r}$ operation has obvious properties in particular we have:

$$\tilde{Q^{r_1 r_2}} = (\tilde{Q^{r2}})^{\tilde{r1}}. \quad (13)$$

We now have to describe the arithmetic of these triples. We assume some knowledge of the tower management process of Ligatsikas et al. (1996) and we do not describe the cases when algebraic variables are not equal. We use the same techniques here.

4.2. Zero check

The triple (γ, Q, d) encodes $Q(\gamma/d) = \frac{\tilde{Q^d}(\gamma)}{d^{|Q|}}$, thus checking the triple to zero is equivalent to checking to zero $\tilde{Q^d}(\gamma)$. That is asking the coding domain which holds an encoding for γ if the polynomial $\tilde{Q^d}(X)$ is zero when evaluated at γ.

Since the elements we manipulate are algebraic integers, we will encode a particular root γ of a polynomial P_γ by P_γ together with an isolating interval $[a_\gamma, b_\gamma[$. Here the endpoints lie in **Q** encoded as pairs of elements of **R**. We will always take P_γ monic and square free and work with reduced (mod P_γ) polynomials.

For an expression Q to be null at γ it is necessary that P_γ and Q have a non-trivial quasi gcd G. We require that the zero check returns either a boolean valued result or a new encoding of some root γ' related to γ when P_γ and Q have a non-trivial gcd. In Axiom, the type of the zero check of the root coding domain becomes:

```
zero?: (UP,%) -> Union(Boolean,Record(new:%,local:R))
```

This returns either an answer or a pair (γ', r_γ) where γ' is a simpler encoding for the real algebraic $\gamma' = r_\gamma \gamma$. This is the case when a non-trivial quasi divisor of P_γ is encountered. This scheme is simpler than the discussion process of Moreno Maza and Rioboo (1996) or Della Dora et al. (1985) since it returns a new encoding for the same number whereas Moreno Maza and Rioboo (1996) or Della Dora et al. (1985) would return a list of encodings (a split in Moreno Maza and Rioboo, 1996).

Whenever the zero check returns a new algebraic variable, we have a pseudo factorization $gP_\gamma = G_\gamma G_\beta$. Here, G_γ is a polynomial which changes signs between a_γ and b_γ. G_γ is thus a non-monic defining polynomial for γ and we want to only work with monic polynomials. We will return an algebraic integer γ' and a localization information r_γ expressing the rule that $r_\gamma \gamma = \gamma'$. In $\tilde{\mathbf{Q}}$, G_γ is a defining polynomial for γ and if we let g_γ be the leading coefficient of G_γ we have:

$$g_\gamma^{|G_\gamma|-1} G_\gamma(X) = G_{\gamma'}(X/g_\gamma).$$

Here, $G_{\gamma'}$ is a monic polynomial with coefficients in **A** which defines the algebraic number $\gamma' = g_\gamma \gamma$. If we select g_γ to be in the base ring **R** and positive we can derive bounds for γ':

$a_{\gamma'} = g_\gamma a_\gamma$ and $b_{\gamma'} = g_\gamma b_\gamma$. We thus have completed an encoding for the new algebraic number γ'.

In order to compute $G_{\gamma'}$ we use the classical Tschirnhauss transformation from G_γ:

$$G_{\gamma'}[X] = g_\gamma^{|G_\gamma|-1} G_\gamma(X/g_\gamma),$$

which can easily be computed. If $P(X) = p_n X^n + P_r(X)$ with $p_n \neq 0$ and $n \geq 1$. We have:

$$p_n^{n-1} P(X/p_n) = X^n + p_n^{n-1-|P_r|} \tilde{P_r^{p_n}}(X).$$

Now, when asking if a triple (γ, Q, r) is zero we may receive the information that $\gamma = \gamma'/g_\gamma$. We thus need to produce a new triple (γ', Q', r') with the same value in $\tilde{\mathbf{Q}}$. We have:

$$Q\left(\frac{\gamma}{r}\right) = \frac{\tilde{Q^r}(\gamma)}{r^{|Q|}} = Q\left(\frac{\gamma'}{g_\gamma r}\right)$$

and the encoding (γ, Q, d) can be replaced by the encoding $(\gamma', Q', r') = (\gamma', Q, g_\gamma r)$ provided that $\tilde{Q^{g_\gamma r}}$ is reduced modulo P'_γ.

The Tschirnhauss transformation increases the size of the coefficients of the polynomial relations verified by the algebraic variable but in practice the case is very rare.

4.3. Reduction

Whenever we have a real algebraic variable γ which gets simplified in a localized real algebraic variable (γ', g_γ), expressions must be expressed in terms of γ' instead of γ.

We know that $\gamma' = g_\gamma \gamma$ and thus a defining polynomial for γ' is $\tilde{P_\gamma^{g_\gamma}}$ where P_γ is a defining polynomial for γ. If $P_{\gamma'}$ is the defining polynomial for γ' we know that the pseudo division of $P_\gamma(X)$ by the polynomial $\frac{P_{\gamma'}(g_\gamma X)}{g_\gamma^{|P'_{\gamma}|-1}} = P'_{\gamma'}(X)$ is exact and that the leading coefficient of $P'_{\gamma'}$ is g_γ. For an expression $Q(\gamma)$ we can write down the pseudo division of $\tilde{Q}(X) = \tilde{Q^{g_\gamma}}(X)$ by $P'_{\gamma'}(X)$. Assuming $|Q| \geq |P'_{\gamma'}|$, we have:

$$g_\gamma^{|Q|-|P'_{\gamma'}|+1} \tilde{Q} = K P'_{\gamma'} + R,$$

that we multiply by $g_\gamma^{|P'_{\gamma'}|-1}$ to obtain:

$$g_\gamma^{|Q|} \tilde{Q} = K g_\gamma^{|P'_{\gamma'}|-1} P'_{\gamma'} + g_\gamma^{|P'_{\gamma'}|-1-|R|} [g_\gamma^{|R|} R],$$

which is:

$$\begin{aligned} g_\gamma^{|Q|} \tilde{Q}(X) &= K(X) P_{\gamma'}(g_\gamma X) + g_\gamma^{|P'_{\gamma'}|-1-|R|} [g_\gamma^{|R|} R(X)] \\ Q(g_\gamma X) &= K(X) P_{\gamma'}(g_\gamma X) + g_\gamma^{|P'_{\gamma'}|-1-|R|} [\tilde{R^{g_\gamma}}(g_\gamma X)]. \end{aligned}$$

When we evaluate this relation at $X = \gamma'$ we see that:

$$Q(\gamma) = g_\gamma^{|P'_{\gamma'}|-1-|R|} \tilde{R^{g_\gamma}}(\gamma)$$

and that we are able to further reduce Q.

4.4. *Addition*

We want here to add two triples (γ, Q_1, r_1) and (γ, Q_2, r_2). We have:

$$\begin{aligned}(\gamma, Q_1, r_1) \oplus (\gamma, Q_2, r_2) &= Q_1\left(\frac{\gamma}{r_1}\right) + Q_2\left(\frac{\gamma}{r_2}\right)\\ &= \frac{\tilde{Q_1^{r_1}}(\gamma)}{r_1^{|Q_1|}} + \frac{\tilde{Q_2^{r_2}}(\gamma)}{r_2^{|Q_2|}}.\end{aligned}$$

We let r be the least common multiple (in **R**) of r_1 and r_2 and d be the maximum of $|Q_1|$ and $|Q_2|$. We now can express the fraction:

$$\begin{aligned}\frac{\tilde{Q^r}(\gamma)}{r^{|Q|}} &= \frac{r^{d-|Q_1|}\left(\frac{r}{r_1}\right)^{|Q_1|}\tilde{Q_1^{r_1}}(\gamma)}{r^d} + \frac{r^{d-|Q_2|}\left(\frac{r}{r_2}\right)^{|Q_2|}\tilde{Q_2^{r_2}}(\gamma)}{r^d}\\ &= \left(\begin{matrix} r^{d-|Q_1|}\left(\frac{r}{r_1}\right)^{|Q_1|}\tilde{Q_1^{r_1}}(\gamma)\\ + \\ r^{d-|Q_2|}\left(\frac{r}{r_2}\right)^{|Q_2|}\tilde{Q_2^{r_2}}(\gamma)\end{matrix}\right)\Bigg/ r^d.\end{aligned}$$

Now the coefficient of degree i of $\tilde{Q^r}$ is always a multiple of $r^{|Q|-i}$ as can be seen in Eq. (12). Thus for j being 1 or 2, the coefficient of degree i of $r^{d-|Q_j|}(\frac{r}{r_j})^{|Q_j|}\tilde{Q_j^{r_j}}(\gamma)$ is a multiple of $r^{d-|Q_j|}(\frac{r}{r_j})^{|Q_j|}r_j^{|Q_j|-i}$ that is $r^{d-i}(\frac{r}{r_j})^i$ which is a multiple of r^{d-i}.

We thus see that if we let Q_c be the numerator of the above fraction, Q_c can be divided by $r^{d-|Q_c|}$ and that we can express the result as the triple $(\gamma, Q_c/r^{d-|Q_c|}, d)$.

4.5. *Multiplication*

We now multiply two triples (γ, Q_1, r_1) and (γ, Q_2, r_2), we know that:

$$(\gamma, Q_1, r_1) \otimes (\gamma, Q_2, r_2) = \frac{\tilde{Q_1^{r_1}}(\gamma)}{r_1^{|Q_1|}}\frac{\tilde{Q_2^{r_2}}(\gamma)}{r_2^{|Q_2|}}$$

and we let Q_c be the polynomial $\tilde{Q_1^{r_1}}\tilde{Q_2^{r_2}}$. The coefficient of degree i of Q_c is

$$\sum_{j=0}^{j=i} q_{1,j}q_{2,i-j},$$

where $q_{1,j}$ and $q_{2,j}$ are the coefficients of degree j of $\tilde{Q_1^{r_1}}$ and $\tilde{Q_2^{r_2}}$. We take $q_{1,j}$ and $q_{2,j}$ to be zero whenever j exceeds the degree. Since $q_{1,j}$ is a multiple of $r_1^{|Q_1|-j}$ and $q_{2,i-j}$ is a multiple of $r_2^{|Q_2|+j-i}$, we know that $q_{1,j}q_{2,i-j}$ is a multiple of $r_1^{|Q_1|-j}r_2^{|Q_2|+j-i}$.

We let r be the lcm of r_1 and r_2 and d be the sum $|Q_1| + |Q_2|$. We re-express:

$$\frac{Q_1^{\tilde{r_1}}(\gamma)}{r_1^{|Q_1|}}\frac{Q_2^{\tilde{r_2}}(\gamma)}{r_2^{|Q_2|}} = \frac{(r/r_1)^{|Q_1|}Q_1(\gamma)}{d^{|Q_1|}}\frac{(r/r_2)^{|Q_2|}Q_2(\gamma)}{d^{|Q_2|}},$$
$$\frac{Q_1^{\tilde{r_1}}(\gamma)}{r_1^{|Q_1|}}\frac{Q_2^{\tilde{r_2}}(\gamma)}{r_2^{|Q_2|}} = \frac{(r/r_1)^{|Q_1|}Q_1(\gamma)(r/r_2)^{|Q_2|}Q_2(\gamma)}{r^{|Q_1|}d^{|Q_2|}}.$$

Let δ_γ be the degree of the defining polynomial P_γ of γ and let Q_c be the denominator of the above fraction. The coefficient of degree i of Q_c is:

$$(r/r_1)^{|Q_1|}(r/r_2)^{|Q_2|}\sum_{j=0}^{j=i} q_{1,j}q_{2,i-j}. \tag{14}$$

Since $q_{1,j}q_{2,i-j}$ is a multiple of $r_1^{|Q_1|-j}r_2^{|Q_2|+j-i}$, we see that:

$$(r/r_1)^{|Q_1|}(r/r_2)^{|Q_2|}q_{1,j}q_{2,i-j}$$

is a multiple of:

$$(r/r_1)^{|Q_1|}(r/r_2)^{|Q_2|}r_1^{|Q_1|-j}r_2^{|Q_2|+i-j},$$

which is:

$$(r/r_1)^{j}(r/r_2)^{i-j}d^{|Q_1|+|Q_2|-i}.$$

The coefficient of degree i of Q_c is thus a multiple of r^{d-i}.

We can now reduce Q_c and express the final result as a triple (γ, Q, r) as in Section 4.3.

4.6. Pseudo inversion

For an algebraic expression $q = (\gamma, Q, r)$ with $Q \in \mathbf{A}[X]$ we compute a pseudo inverse $\overline{q}$ and a pseudo norm $\|q\|$ by first computing a pair $(\overline{\tilde{Q}}, p)$ with an extended version of Algorithm 4 with input P_γ and $Q^{\tilde{r}}$. We select the function f in Algorithm 4 to be the pseudo inverse function itself. The function g is such that $g(a) = \gcd(a, \|a\|)$. Here $\overline{\tilde{Q}}$ lies in $\mathbf{A}[X]$ and p lies in $\mathbf{A}$. We then recursively compute the pseudo inverse $\overline{p}$ and pseudo norm $\|p\|$ of p which lie respectively in $\mathbf{A}$ and $\mathbf{R}$. We thus have:

$$Q^{\tilde{r}}(X)\overline{\tilde{Q}}(X) + P(X)P_\gamma(X) = p$$

and

$$(\overline{p}\,\overline{\tilde{Q}}(X))Q^{\tilde{r}}(X) + (\overline{p}P(X))P_\gamma(X) = \|p\|.$$

Evaluating at γ gives:

$$\overline{p}\,\overline{\tilde{Q}}(\gamma)Q^{\tilde{r}}(\gamma) = \|p\|$$
$$\overline{p}r^{|Q|}\overline{\tilde{Q}}(\gamma)Q(\gamma/r) = \|p\|$$
$$\overline{p}r^{|P_\gamma|-1}\overline{\tilde{Q}}(\gamma)Q(\gamma/r) = r^{|P_\gamma|-|Q|-1}\|p\|.$$

Since $|\overline{\tilde{Q}}| < |p_\gamma|$ we can write $r^{|P_\gamma|-1}\overline{\tilde{Q}}$ as a polynomial $\overline{Q}(\gamma/r)$.

d	Ran	Rat-mr	Rat-rr	Int-mr	Int-rr
3	0.04	0.05	0.06	0.05	0.06
4	0.14	0.15	0.15	0.13	0.11
5	0.82	0.66	0.76	0.39	0.37
6	22.85	8.47	9.79	4.49	4.58
7	34.99	12.25	19.88	9.55	10.88
8	1035.74	245.12	333.90	193.47	203.70

Fig. 4. `allRootsOf` $(\prod_{i=1}^{i=d}(X-i)+\sqrt[d]{2}X^{d-1})$.

We can now return the pseudo inverse $\overline{q} = (\gamma, \overline{p}\,\overline{Q}, r)$ and the pseudo norm $\|q\| = r^{|P_\gamma|-|Q|-1}\|p\|$

Other operations proceed as in Moreno Maza and Rioboo (1996) or Ligatsikas et al. (1996) managing towers of extensions when we operate on two triples (γ_1, Q_1, r_1) and (γ_2, Q_2, r_2) with $\gamma_1 \neq \gamma_2$.

5. Conclusion

Very recently, thanks to the initiative of Tim Daly, NAG Ltd has released the copyright of Axiom. A new version will soon be available under a free and open source license. There is thus no practical objection on using Axiom to develop algorithms anymore.

We give here some examples which give an experimental validation of the utility of the algorithms presented here. These are the computation of the roots of a polynomial of degree d with coefficients over an extension of degree d. All calculations are done in an extension of degree d and the roots produced enable us to work in an extension of degree d^d.

Running times given in Fig. 4 are in seconds of Axiom-2.3 time when running on a Linux 400 MHz machine with 64 MB of physical memory. Columns are to be interpreted as follows:

Ran is the standard Axiom algorithm which uses the primitive Euclidean algorithms inside zero checks and its extended version inside inversions. Sign computations use refinements and Descartes rule of sign. Roots production use Sylvester sequences.

Rat-mr is the specialization of Algorithm 4 which uses the primitive Euclidean algorithm and its extended version together with Sylvester sequences.

Rat-rr is the specialization of Algorithm 4 which uses the Euclidean algorithm and its extended version together with Sylvester sequences.

Int-mr is the specialization of Algorithm 4 which uses the Moreno Maza and Rioboo (1996) algorithm and its extended version together with quasi Sylvester sequences.

Int-rr is the specialization of Algorithm 4 which uses the weak sub-resultant algorithm and its extended version together with quasi Sylvester sequences.

The cases **Rat-mr**, **Rat-rr**, **Int-mr** and **Int-rr** are computed by different instantiations of the same program. We clearly see that the scheme using rings is faster than the scheme using fields. When working over a field it is faster to compute Sylvester sequences than compute refinements and use Descartes rule of sign. For the scheme of fields the primitive version of the algorithm is faster whereas this is not so obvious for the scheme of rings.

We plan to release our implementation for the forthcoming new version of Axiom. Using an Intel Pentium 400 under Linux, Ramanujan's Eq. (1) of the introduction used to take 15 s to solve with Axiom 2.3 and version 1 of the real closure. Our current development version 2 uses algorithm **Rat-mr** and takes less than 1 s under the same conditions.

Another possible use of quasi Sylvester sequences of Section 2 can be to take advantage of their properties inside a solver dedicated to real solutions.

A natural extension of the techniques presented here would be to use faster sub-resultant algorithms than Collins and Loos (1982). For instance algorithms of Ducos (2000) or Lombardi et al. (2000) could be considered. We think that there is no objection for this generalization. Our main problem in this generalization is that we need to further inspect the reduction case in the algorithm and cannot rely on pseudo remainder properties.

Acknowledgements

The author would like to thank the different referees for their patience and very useful comments and corrections.

References

Basu, S., Pollack, R., Roy, M.F., 1996. On the combinatorial and algebraic complexity of quantifier elimination. Journal of the ACM 43 (6), 1002–1046.

Bochnak, J., Coste, M., Roy, M., 1988. Géométrie Algébrique Réelle. Springer.

Collins, G.E., Loos, R., 1982. Real zeros of polynomials. In: Computer Algebra. Springer, New York, pp. 83–94.

Coste, M., Roy, M., 1988. Thom's lemma, the coding of real algebraic numbers and the computation of the topology of semi-algebraic sets. Journal of Symbolic Computation 5, 121–129.

Davenport, J., Siret, Y., Tournier, E., 1987. Calcul Formel: systèmes et algorithmes de manipulations algébriques. Masson.

Della Dora, J., Discrescenzo, D., Duval, D., 1985. About a new method for computing in algebraic number fields. In: Lecture Notes in Computer Science, vol. 204.

Ducos, L., 2000. Optimizations of the subresultant algorithm. Journal of Pure and Applied Algebra 145, 149–163.

Duval, D., Gonzalez Vega, L., 1996. Dynamic evaluation and real closure. Mathematics and Computers in Simulation 42, 551–560.

Gonzalez Vega, L., Rouillier, F., Roy, M.-F., Trujillo, G., 1998a. Symbolic recipes for real solutions. In: Cohen et al. (1998), pp. 121–167.

Gonzalez Vega, L., Roy, M.-F., Rouillier, F., 1998b. Symbolic recipes for polynomial system solving. In: Cohen et al. (1998), pp. 34–65.

Hollcott, A., 1941. Finite konstruktion geordneter algebraischer erweterungen von geordneten grundkorpern. Ph.D. Thesis, Univ. of Hamburg.

Lang, S., 1964. Algebraic Numbers. Addison-Wesley Pub. Co, New York.

Lang, S., 1969. Algebra. Addison-Wesley Pub. Co, New York.

Lecerf, G., 1996. Dynamic evaluation and real closure. Implementation in Axiom. Available from http://www.medicis.polytechnique.fr/~lecerf/.

Lickteig, T., Roy, M.-F., 2001. Sylvester-Habicht sequences and fast Cauchy index computations. Journal of Symbolic Computation 31, 315–341.

Ligatsikas, Z., Rioboo, R., Roy, M.F., 1996. Generic closure of an ordered field, implementation in Axiom. Mathematics and Computers in Simulation 42, 541–549.

Lombardi, H., Roy, M.F., 1991. Elementary constructive theory of ordered fields. Progress in Mathematics 34, 249–262.

Lombardi, H., Roy, M.F., Safey, M., 2000. New structure theorems for subresultants. Journal of Symbolic Computation 29, 663–690.

Loos, R., 1982. Generalized polynomial remainder sequences. In: Computer Algebra. Springer, New York, pp. 115–137.

Moreno Maza, M., Rioboo, R., 1996. Polynomial gcd computations over towers of algebraic extensions. In: Lecture Notes in Computer Science, vol. 948.

Rioboo, R., 1992. Computation of the real closure of an ordered field. In: ISSAC'92. Academic Press, San Francisco.

Strzeboński, A., 1997. Computing in the field of complex algebraic numbers. Journal of Symbolic Computation 647–656.

Zassenhauss, H., 1970. A real root calculus. In: Computational Problems in Abstract Algebra. Pergamon Press, Oxford, pp. 383–392.

Further reading

Buchberger, B., Collins, G., Loos, R., 1982. Computer Algebra. Springer.

Cohen, A., Cuyper, H., Sterk, H. (Eds.), 1998. Some Tapas of Computer Algebra, In: Algorithms and Computation in Mathematics, vol. 4. Springer.

Loos, R., 1982. Computing in algebraic extensions. In: Computer Algebra, Springer, pp. 173–187.

Rioboo, R., 1991. Quelques aspects du calcul exact avec les nombres réels. Ph.D. Thesis, Laboratoire d'Informatique Théorique et Programmationg.