Polynomial Gcd Computations over Towers of Algebraic Extensions

Marc Moreno Maza and Renaud Rioboo

LITP, Institut Blaise Pascal, Boite 168
Université Pierre et Marie Curie
4, Place Jussieu
75252 Paris Cedex 05
e-mail : m3@posso.ibp.fr, rr@posso.ibp.fr

Abstract. Some methods for polynomial system solving require efficient techniques for computing univariate polynomial gcd over algebraic extensions of a field. Currently used techniques compute *generic* univariate polynomial gcd before *specializing* the result using algebraic relations in the ring of coefficients. This strategy generates very big intermediate data and fails for many problems. We present here a new approach which takes permanently into account those algebraic relations. It is based on a property of subresultant remainder sequences and leads to a great increase of the speed of computation and thus the size of accessible problems.
Keywords. Polynomial gcd, subresultants, algebraic numbers, triangular sets, zero dimensional systems, AXIOM.

1 Introduction

Polynomial gcd computations appear among other problems in algebraic number manipulation and in system solving through triangular systems. This paper presents a new algorithm to compute gcds of univariate polynomial with coefficients satisfying algebraic relations. It is a generalization of the subresultants algorithm ([14]) which enables machine resolution of problems that could not be solved with classical techniques. One of the main reasons is roughly the following. When we compute *generic* univariate polynomial gcd (using the subresultants algorithm) before *specializing* the result using algebraic relations in the ring of coefficients, we deal with univariate polynomials whose coefficients are multivariate polynomials. Thus, when applying the subresultants algorithm, pseudo-divisions increase dramatically the size of the coefficients of the intermediate polynomials and exact quotients are expensive. Our idea is to keep the leading coefficients of the intermediate polynomials in the base field. So we limit the swell of their coefficients and make easy the exact quotients which appear in the subresultants algorithm. To do so, we need to test the invertibility of the leading coefficients. This leads us to use our algorithm recursively, applying a *D5-like process* ([4, 5]). Now, let us introduce the problem more formally. We will state it completely in section 2.

Let $\mathbf{R}$ be an integral domain (a commutative ring with unit element and no zero-divisors), $\mathbf{K}$ its fraction field and $\mathbf{K}[X_1]$ the ring of univariate polynomials with coefficients in $\mathbf{K}$. Let $p_1 \in \mathbf{K}[X_1]$ be a square-free polynomial of positive degree and $\mathbf{A}_1$ be $\mathbf{K}[X_1]/\langle p_1 \rangle$. How to test if $q \in \mathbf{A}_1$ is invertible ? This can be done by interpreting q as an element of $\mathbf{K}[X_1]$ and by computing the gcd g of q and p_1 in $\mathbf{K}[X_1]$. This yields to a *factorization* $p_1 = fg$ together with a decomposition of $\mathbf{K}[X_1]/\langle p_1 \rangle$ into a product $\mathbf{K}[X_1]/\langle f \rangle$ and $\mathbf{K}[X_1]/\langle g \rangle$. In the first component q is invertible and is zero in the second. Thus, with no necessity for p_1 to be irreducible, we have answered the question. This remark of D. Lazard is the basis of the D5 system ([4, 5]) for algebraic number computations.

We see that the question of invertibility in $\mathbf{K}[X_1]/\langle p_1 \rangle$ led us to the question of gcd in $\mathbf{K}[X_1]$ together with a *discussion* process. This is a fundamental problem (over more general rings) in many classes of methods for algebraic solving ([9, 11, 12]). This problem is solved using the subresultant specialization property ([14]) by an algorithm described in [11]. We will briefly recall it here in the next section. However, a first implementation of the lextriangular algorithm of D. Lazard ([12]) using these techniques induced a 400 megabyte memory saturation for certain problems. This motivated us to develop better gcd algorithms. In section 3 we give definitions for the notions of *polynomial gcd* and *discussion* (we will use the term of *splits*) in our context. Then, we give a property of the subresultant remainder sequences algorithm which avoids the swell of memory appearing in the current techniques. In section 4 we revisit the problem of finding gcd described in section 2 in light of the previous remarks. We give explicit and complete algorithms to answer this problem. These algorithms are our AXIOM programs ([7]) up to some syntax details. So we hope that they could be easily adapted in other programming languages. We then show how this approach enables to have an efficient implementation for the lextriangular algorithm. In particular it allows a completely automatic solution for the *cyclic-7 system* ([6]) with results as in [1].

2 Background

Let us assume that we have introduced non-constant polynomials $p_1, \ldots, p_n$ and rings $\mathbf{A}_1, \ldots, \mathbf{A}_n$ such that $p_i \in \mathbf{A}_{i-1}[X_i]$ with leading coefficient in $\mathbf{A}_0 = \mathbf{K}$, the rings $\mathbf{A}_i$ being recursively defined as the quotient ring $\mathbf{A}_i = \mathbf{A}_{i-1}[X_i]/\langle p_i \rangle$. Though the $\mathbf{A}_i$ are not necessarily fields (they may have zero-divisors) we will call each one an *algebraic tower of simple extensions* of $\mathbf{K}$. Since $\mathbf{A}_i$ is a finite dimensional $\mathbf{K}$-algebra, an element of $\mathbf{A}_i$ is either a zero-divisor either invertible. Let us write $\mathbf{P}_i = \mathbf{K}[X_1, \ldots, X_i]$ for $1 \leq i \leq n+1$ and $\mathbf{P}_0 = \mathbf{K}$. For $1 \leq i \leq n$ we consider H_i the surjective canonical homomorphism from $\mathbf{A}_{i-1}[X_i]$ to $\mathbf{A}_i$ and F_i the surjective homomorphism from $\mathbf{P}_{i+1}$ to $\mathbf{A}_i[X_{i+1}]$ defined by

(i) $F_i(X_{i+1}) = X_{i+1}$

(ii) $(\forall p \in \mathbf{P}_i)$ $F_i(p) = \begin{cases} H_1(p) & \text{if } i = 1 \\ H_i(F_{i-1}(p)) & \text{if } i > 1 \end{cases}$

Let $\widehat{p}_1, \ldots, \widehat{p}_n$ be polynomials of $\mathbf{P}_n$ representing the p_i, that is such that $\widehat{p}_i \in \mathbf{P}_i \backslash \mathbf{P}_{i-1}$ and $F_i(\widehat{p}_i) = p_i$ (we can assume that the leading coefficient of $\widehat{p}_i$, viewed as a polynomial in X_i, lies in $\mathbf{K}$). Let I be the ideal generated by $\widehat{p}_1, \dots, \widehat{p}_n$ in $\mathbf{P}_n$, the ring $\mathbf{P}_n/I$ is isomorphic to $\mathbf{A}_n$.

Let a and b in $\mathbf{A}_n[X_{n+1}]$ which have a gcd, let $\widehat{a}$ and $\widehat{b}$ be representing a and b in $\mathbf{P}_n[X_{n+1}]$ (i.e. $F_n(\widehat{a}) = a$ and $F_n(\widehat{b}) = b$). The classical method to compute such a gcd is to first compute a subresultant remainder sequence $(\widehat{r}_0, \widehat{r}_1, \dots, \widehat{r}_m, \widehat{r}_{m+1})$ of $\widehat{a}$ and $\widehat{b}$ in $\mathbf{P}_n[X_{n+1}]$ and to search for possible gcds among the $\widehat{r}_i$. Let j be the biggest integer such that $F_n(\widehat{c}_j) \neq 0$, where $\widehat{c}_j$ is the leading coefficient of $\widehat{r}_j$. We write $r_j = F_n(\widehat{r}_j)$ (using the specialization property of subresultants we know that $r_k = 0$ for $k > j$).

We then determine if $F_n(\widehat{c}_j)$ is invertible in $\mathbf{A}_n$, if $\widehat{c}_j \in \mathbf{K}$ this is the case and r_j is a gcd for a and b. Otherwise there exists an l such that $\widehat{c}_j \in \mathbf{P}_l \setminus \mathbf{P}_{l-1}$. We compute a gcd of $F_{l-1}(\widehat{c}_j)$ and p_l in $\mathbf{A}_{l-1}[X_l]$ by recursively calling the method. If this gcd p'_l is of degree 0 in X_l we know that $F_n(\widehat{c}_j)$ is invertible in $\mathbf{A}_n$ and r_j is a gcd of a and b. Otherwise we have a factorization $p_l = p'_l p''_l$ in $\mathbf{A}_{l-1}[X_l]$, we let $\widehat{p}'_l$ and $\widehat{p}''_l$ be such that $p'_l = F_{l-1}(\widehat{p}'_l)$ and $p''_l = F_{l-1}(\widehat{p}''_l)$ and we consider the ideals I' and I'' of $\mathbf{P}_n$ obtained from I by respectively substituting $\widehat{p}'_l$ and $\widehat{p}''_l$ for $\widehat{p}_l$.

As we did for $\mathbf{A}_n$, we can define the sequences $\mathbf{A}'_l, \dots, \mathbf{A}'_n$ and $\mathbf{A}''_l, \dots, \mathbf{A}''_n$ together with the surjective homomorphisms F'_n and F''_n that will respectively enable to identify $\mathbf{A}'{}_n$ with $\mathbf{P}_n/I'$ and $\mathbf{A}''{}_n$ with $\mathbf{P}_n/I''$. In the tower $\mathbf{P}_n/I''$, $F''_n(\widehat{c}_j)$ is invertible and $F''_n(\widehat{r}_j)$ is a gcd of $F''_n(\widehat{a})$ and $F''_n(\widehat{b})$. On the contrary in $\mathbf{P}_n/I'$ we have $F''_n(\widehat{r}_j) = 0$, and the gcd has to be found among the $(\widehat{r}_0, \widehat{r}_1, \dots, \widehat{r}_{j-1})$ searching again for the last with a non failing leading coefficient.

Despite the problem with the technical management of the discussion and of the recursion a new problem appears which is the prohibitive size of the $\widehat{r}_j$. Leading to the idea to replace computations in $\mathbf{P}_n$ by computations in $\mathbf{A}_n$. If the ideal I is radical, the ring $\mathbf{A}_n$ is a product of fields, we were able to *lift* partial results (propositions 3 and 7).

3 Preliminaries

Notations 1 Let $\mathbf{A}$ be a ring (all rings considered here will be commutative with unit element) and $E \subseteq \mathbf{A}$. We denote by $\mathbf{Id}_{\mathbf{A}}(E)$ the ideal of $\mathbf{A}$ generated by the elements of E, $\mathbf{A}/E$ the quotient ring $\mathbf{A}/\mathbf{Id}_{\mathbf{A}}(E)$ and $\overline{a}^E$ the residue class of an element a of $\mathbf{A}$ in $\mathbf{A}/E$. If E is empty we state $\mathbf{A}/E = \mathbf{A}$. If $E = \{a_1, \dots, a_n\}$ we will denote by $\langle a_1, \dots, a_n \rangle_{\mathbf{A}}$ (or simply $\langle a_1, \dots, a_n \rangle$) the ideal $\mathbf{Id}_{\mathbf{A}}(E)$. We will denote by $\mathbf{nz}(\mathbf{A})$ the multiplicatively closed subset of non-zero-divisors of $\mathbf{A}$ (this contains the group of invertible elements of $\mathbf{A}$) and by $\mathbf{q}(\mathbf{A})$ the ring of fractions with numerators in $\mathbf{A}$ and denominators in $\mathbf{nz}(\mathbf{A})$. For $a \in \mathbf{A}$ and $b \in \mathbf{nz}(\mathbf{A})$, if there exists an element (necessarily unique) c such that $a = bc$ we will denote it by a exquo b. Let $p, q \in \mathbf{A}[X]$ with $p \neq 0$, we write $\mathsf{mcoeff}(p)$ for the coefficient of its highest degree monomial, and $\mathsf{deg}(p)$ this degree. We will say that p is quasi-monic if $\mathsf{mcoeff}(p) \in \mathbf{nz}(\mathbf{A})$. When $\mathsf{deg}(p) > 0$, we denote by $\mathsf{prem}(q, p)$ the pseudo-remainder of q by p.

For a module $\mathbf{M}$ over $\mathbf{A}$ and a multiplicatively closed subset S of $\mathbf{nz}(\mathbf{A})$, we denote by $\mathrm{S}^{-1}\mathbf{M}$ the $\mathbf{A}$-module of fractions with numerator in $\mathbf{M}$ and denominator in S. Every element l of $\mathrm{S}^{-1}\mathbf{M}$ can be (non uniquely) decomposed into the form am/s where $a \in \mathbf{A}$, $m \in \mathbf{M}$ and $s \in S$. We write $\mathsf{numPart}(l) = a$, $\mathsf{polPart}(l) = m$, $\mathsf{numer}(l) = am$ and $\mathsf{denom}(l) = s$. If $\mathbf{M}$ is a polynomial ring over an integral domain, then two different values of $\mathsf{polPart}(l)$ generate the same ideal in the polynomial ring with coefficients over the fraction field of $\mathbf{A}$. Furthermore, if $\mathbf{A}$ is factorial and if S is $\mathbf{nz}(\mathbf{A})$ then a, m and s can be uniquely chosen. Finally if $\mathbf{A}$ is a polynomial ring with coefficients over a field $\mathbf{K}$ and let $\mathbf{L}$ be a field containing $\mathbf{K}$, for an ideal I of $\mathbf{A}$, we will denote by $\mathbf{V}_L(I)$ (or simply $\mathbf{V}(I)$) the affine variety of $\mathbf{L}^n$ associated to I.

3.1 Univariate Polynomial Gcds

Definition 1 *Let $\mathbf{A}$ be a ring and p_1, p_2, g in $\mathbf{A}[X]$. We say that g is a* gcd *of p_1 and p_2 if we have :*

$$\langle p_1, p_2 \rangle_{\mathbf{q}(\mathbf{A})[X]} = \langle g \rangle_{\mathbf{q}(\mathbf{A})[X]}$$

Remark 1 If $\mathbf{A}$ is a field then we recover the usual gcd notion in $\mathbf{A}[X]$. If $\mathbf{A}$ is factorial, g is a (usual) gcd if and only if g is a (definition 1) gcd and a divisor of both p_1 and p_2 in $\mathbf{A}[X]$. If $\mathbf{q}(\mathbf{A})[X]$ is not principal, p_1 and p_2 do not necessarily have a definition 1gcd. The choice of this definition is to generalize usual gcd algorithms which all give a Bézout relation together with a pseudo-divisor. Furthermore, the propositions 2 and 3 will justify this choice.

Remark 2 If p_1 and p_2 have two quasi-monic gcds g and g' then there exists $e, e' \in \mathbf{nz}(\mathbf{A})$ such that $e\,g = e'\,g'$. In practice we will look for quasi-monic gcds as we will do when $\mathbf{A}$ is an algebraic tower of simple extensions of a field $\mathbf{K}$. To do so, we will use the notions of ideal split and quasi-inverse, together with a generalization of the subresultant remainder sequences algorithm that we present now.

Definition 2 *Let $\mathbf{A}$ be an integral domain, f an application from $\mathbf{nz}(\mathbf{A})$ into $\mathbf{nz}(\mathbf{A})$ and p_1, p_2 two polynomials of $\mathbf{A}[X]$. We consider the algorithm* NewSubResGcd *defined by*

$\mathsf{NewSubResGcd}(p_1, p_2, \boxed{f{:}(\mathbf{nz}(\mathbf{A}) \mapsto \mathbf{nz}(\mathbf{A}))}) ==$
  if $\mathsf{deg}(p_2) > \mathsf{deg}(p_1)$ **then** $(p_1, p_2) := (p_2, p_1)$
  $\delta := \mathsf{deg}(p_1) - \mathsf{deg}(p_2)$
  $\psi' : \mathbf{A} := -1\ ;\ \beta' : \mathbf{A} := -1^{\delta+1}$
  $\boxed{p_2 := f(\mathsf{mcoeff}(p_2)) * p_2}$
  $\alpha' : \mathbf{A} := \mathsf{mcoeff}(p_2)$
  while $(\mathsf{deg}(p_2) > 0)$ **repeat**
    $p_3 := \mathsf{prem}(p_1, p_2)$ exquo β'
    $\boxed{p_3 := f(\mathsf{mcoeff}(p_3)) * p_3}$

if $\delta > 0$ **then**
$\psi' := (-\alpha')^\delta$ exquo ${\psi'}^{\delta-1}$
$(p_1\ , p_2) := (p_2\ , p_3)$
if $(\deg(p_2) > 0)$ **then**
$\delta := \deg(p_1)$ - $\deg\ (p_2)$
$\beta' :=$ -$\alpha'\,{\psi'}^\delta$
$\alpha' := \mathsf{mcoeff}(p_2)$
if $p_2 = 0$ **then return** p_1 **else return** p_2

Remark 3 The only difference with the original subresultant remainder sequences algorithm is that at each step we added a multiplication of the polynomial p_i by a constant (which is chosen by the function f). If $f(x) = 1$, we exactly have the original algorithm. Moreover if $\mathbf{A}$ is a field and if $f(x) = 1/x$ we have the monic euclidean algorithm (where the polynomials are kept monic during computations).

Theorem 1 *Let $\mathbf{A}$ be an integral domain. For every function $f : \mathbf{nz}(\mathbf{A}) \to \mathbf{nz}(\mathbf{A})$, the NewSubResGcd terminates and computes a gcd as in definition 1.*

Proof. $\triangleright$ Next proposition will show that the algorithm is done in $\mathbf{A}[X]$, meaning that all divisions are exact divisions. It is clear that the algorithm terminates. It computes a gcd (as in definition 1) since at each step the polynomials p_1 and p_2 on one hand and the polynomials p_2 and p_3 on the other generate the same ideal in $\mathbf{q}(\mathbf{A})[X]$. Moreover, if the last computed p_2 is null, p_1 generates this ideal, else this p_2 does. $\triangleleft$

Proposition 1 *Let F'_k, α'_k, β'_k and ψ'_k be the terms computed by NewSubResGcd and defined for $k \geq 3$ as follow $F'_1 = p_1$, $\lambda_2 = f(\mathsf{mcoeff}(p_2))$, $F'_2 = \lambda_2 p_2$, ${\alpha'}_2 = \mathsf{mcoeff}(F'_2)$, $\delta_{k-2} = \deg(F'_{k-2}) - \deg(F'_{k-1})$, $\psi'_3 = -1$, $\beta'_3 = (-1)^{\delta_1+1}$, $\lambda_k = f(\mathsf{mcoeff}(\mathsf{prem}(F'_{k-2}, F'_{k-1})))$, $\lambda_k \mathsf{prem}(F'_{k-2}, F'_{k-1}) = \beta'_k F'_k$, $\psi'_{k+1} = \dfrac{(-\alpha'_{k-1})^{\delta_{k-2}}}{{\psi'_k}^{\delta_{k-2}-1}}$, $\beta'_{k+1} = -\alpha'_{k-1}{\psi'_{k+1}}^{\delta_{k-1}}$, and ${\alpha'}_k = \mathsf{mcoeff}(F'_k)$. Then, all the F'_k remain in $\mathbf{A}[X]$ and all the ψ'_k remain in $\mathbf{A}$.*

Proof.$\triangleright$ Let $F_1, F_2, \ldots, F_k$ be the polynomials computed by the original subresultant algorithm, α_k the leading coefficient of F_k; the β_k and ψ_k are given by $\beta_3 = (-1)^{\delta_1+1}$ and $\beta_{k+1} = -\alpha_{k-1}\psi_{k+1}^{\delta_{k-1}}$ with $\psi_3 = -1$ and $\psi_{k+1} = \dfrac{(-\alpha_{k-1})^{\delta_{k-2}}}{\psi_k^{\delta_{k-2}-1}}$. The β'_k and ψ'_k verify the same recurrence relation where α'_k (the leading coefficient of F'_k) replaces α_k. Let us first remark that F_k and F'_k are associated when they are interpreted as polynomials of $\mathbf{q}(\mathbf{A})[X]$. Let us examine the beginning of the algorithm; we have $F'_2 = \lambda_2 F_2$, $\alpha'_2 = \lambda_2\alpha_2$, $\psi'_3 = \psi_3$, $\beta'_3 = \beta_3$. We can write $\mathsf{prem}(F'_1, F'_2) = \mathsf{prem}(F_1, \lambda_2 F_2) = \lambda_2^{\delta_1+1}\mathsf{prem}(F_1, F_2)$ and thus:

$$F'_3 = \lambda_3\lambda_2^{\delta_1+1}F_3$$

since $\beta'_3 = \beta_3$. We now consider the terms generally, putting: $F'_i = \mu_i F_i$ and writing relations in the fraction field of $\mathbf{A}$ we have:

$$\begin{aligned}\lambda_{i+2}\mathsf{prem}(F'_i, F'_{i+1}) &= \beta'_{i+2}F'_{i+2} \\ &= \beta'_{i+2}\mu_{i+2}F_{i+2} \\ &= \lambda_{i+2}\mathsf{prem}(\mu_i F_i, \mu_{i+1}F_{i+1}) \\ &= \lambda_{i+2}\mu_i\mu_{i+1}^{\delta_i+1}\beta_{i+2}F_{i+2}\end{aligned}$$

And thus $\beta'_{i+2}\mu_{i+2} = \lambda_{i+2}\mu_i\mu_{i+1}^{\delta_i+1}\beta_{i+2}$ and, replacing the β by the ψ and since $\alpha'_i = \mu_i\alpha_i$, we see that:

$$\mu_{i+2}{\psi'}^{\delta_i}_{i+2} = \lambda_{i+2}\mu_{i+1}^{\delta_i+1}\psi_{i+2}^{\delta_i} \tag{1}$$

Meaning:

$$\frac{\mu_{i+2}}{\mu_{i+1}} = \lambda_{i+2}\left[\frac{\mu_{i+1}\psi_{i+2}}{{\psi'}_{i+2}}\right]^{\delta_i}. \tag{2}$$

On another hand, if we induce recurrence relations on the ψ_i, we have:

$$\begin{aligned}\frac{\mu_{i+1}\psi_{i+2}}{{\psi'}_{i+2}} &= \mu_{i+1}\left[\frac{\alpha_i}{\alpha'_i}\right]^{\delta_{i-1}}\left[\frac{{\psi'}_{i+1}}{\psi_{i+1}}\right]^{\delta_{i-1}-1} \\ &= \left[\frac{{\psi'}_{i+1}}{\psi_{i+1}}\right]^{\delta_{i-1}-1}\frac{\mu_{i+1}}{\mu_i^{\delta_{i-1}}}\end{aligned} \tag{3}$$

but stating $i \to i-1$ in (2) we find $\left[\frac{\psi_{i+1}}{{\psi'}_{i+1}}\right]^{\delta_{i-1}} = \frac{\lambda_{i+1}\mu_i^{\delta_{i-1}+1}}{\mu_{i+1}}$ which we substitute in (3) to obtain:

$$\frac{\mu_{i+1}\psi_{i+2}}{{\psi'}_{i+2}} = \frac{\lambda_{i+1}\mu_i^{\delta_{i-1}+1}}{\mu_{i+1}}\mu_{i+1}\left[\frac{\psi_{i+1}}{\mu_i\psi'_{i+1}}\right]^{\delta_{i-1}} = \lambda_{i+1}\frac{\mu_i\psi_{i+1}}{{\psi'}_{i+1}}$$

and stating $u_i = \frac{\mu_i\psi_{i+1}}{{\psi'}_{i+1}}$ we have $u_i = \prod_{j=2}^{j=i}\lambda_j$ since $\mu_2 = u_2 = \lambda_2$. To show that computations remain in $\mathbf{A}$ it suffices to find an entire value for μ_i. We substitute the value of u_i in (2), obtaining:

$$\frac{\mu_{i+2}}{\mu_{i+1}} = \lambda_{i+2}{u_{i+1}}^{\delta_i} \tag{4}$$

and thus μ_i remain in $\mathbf{A}$ and is divisible by u_i. It follows that ψ'_i is also in $\mathbf{A}$ (since $\psi_i \in \mathbf{A}$). $\triangleleft$

Remark 4 We will adapt this algorithm using a *D5-like process* to the case where $\mathbf{A}$ is an algebraic tower of simple extensions and where we can no more assume that *intermediate* rings remain integral. To do so, we introduce the notions of *quasi-inverse* and *split of an ideal* that we present now.

3.2 Quasi-inverse

Definition 3 *Let* $\mathbf{R}$ *be an integral domain and* $\mathbf{A}$ *an* $\mathbf{R}$*-torsion-free ring, i.e. a ring containing* $\mathbf{A}$ *such that* $\mathbf{nz}(\mathbf{R}) \subseteq \mathbf{nz}(\mathbf{A})$*.. We say that a couple* $(y, r) \in \mathbf{A} \times \mathbf{R}$ *is an* $\mathbf{R}$-quasi-inverse *(or simply quasi-inverse) if we have both*

(i) $xy = r$
(ii) $x \neq 0 \Longrightarrow y \neq 0$

Remark 5 Let $x \in \mathbf{A}$ and (y, r) an $\mathbf{R}$-quasi-inverse of x, we have

$$x \notin \mathbf{nz}(\mathbf{A}) \Longleftrightarrow r = 0.$$

If r is null, x is a zero-divisor in $\mathbf{A}$. Conversely, let $z \in \mathbf{A}, z \neq 0$ be such that $zx = 0$, we have $xyz = zr = 0$. Since $\mathbf{A}$ is $\mathbf{R}$-torsion-free, if $r \neq 0$ then $z = 0$. A contradiction.

Remark 6 It follows that if $x \in \mathbf{nz}(\mathbf{A})$ then $\frac{y}{r}$ is the inverse of x in $\mathbf{q}(\mathbf{A})$. If $x \notin \mathbf{nz}(\mathbf{A})$ a quasi-inverse of x is every couple $(y, 0)$, where $xy = 0$.

In the case of an algebraic tower of simple extensions, we will show how the computation of a quasi-inverse and that of a gcd *resolve mutually.* In section 4 we will return to the problem exposed in section 2. We will apply the *NewSubResGcd* algorithm where $\mathbf{A}$ is an algebraic tower of simple extensions using a *D5-like process.* To avoid fractions we will work with multivariate polynomials with coefficients in the integral domain $\mathbf{R}$ instead of its fraction field $\mathbf{K}$. Thus, the parameter f (of the algorithm *NewSubResGcd*) applied to the leading coefficient c of an intermediate polynomial, if c is invertible, will return a quasi-inverse of c instead of an inverse. To compute $f(c)$ we will use an extended version of the *NewSubResGcd* algorithm. Finally the leading coefficient of the intermediate polynomials i.e. the α'_k, and the β'_k and ψ'_k will remain in $\mathbf{R}$. To understand better the efficiency of that choice, it may be interesting to have an interpretation of our method using determinantal polynomials. We leave it for a future paper.

3.3 Split of an Ideal

Definition 4 *Let* $\mathbf{A}_1, \ldots, \mathbf{A}_n$ *be rings. The (Cartesian) product* $\mathbf{A}_1 \times \cdots \times \mathbf{A}_n$ *has a ring structure, called* product-ring *if addition and multiplication are defined componentwise. From now on, every product of rings (and in particular of fields) will have this structure.*

Proposition 2 *Let* $\mathbf{A}$ *be a ring isomorphic to some product-ring* $\mathbf{A}_1 \times \cdots \times \mathbf{A}_n$*. Let* p*,* q *be in* $\mathbf{A}[X]$*. Let* h_i *be the projection from* $\mathbf{A}$ *into* $\mathbf{A}_i$ *and* f_i *the homomorphism extending* h_i *to an homomorphism of* $\mathbf{A}[X]$ *into* $\mathbf{A}_i[X]$ *(stating* $f_i(X) = X$*).*

If for each i *the polynomials* $f_i(p)$ *and* $f_i(q)$ *have gcds (resp quasi-monic gcds) in* $\mathbf{A}_i[X]$ *then* p *and* q *have gcds (resp quasi-monic gcds) in* $\mathbf{A}[X]$*.*

In particular if all the $\mathbf{A}_i$ *are integral domains then every couple of polynomials* (p, q) *has gcds in* $\mathbf{A}[X]$*, if the* $\mathbf{A}_i$ *are furthermore Noetherian, then all the ideals of* $\mathbf{q}(\mathbf{A})[X]$ *are principal.*

Definition 5 *Let* $\mathbf{A}$ *be a ring and* I *an ideal of* $\mathbf{A}$*. We call* split *every tuple* $S = (S_1, \ldots, S_n)$ *of ideals of* $\mathbf{A}$ *such that:*

$$S_1 \cdots S_n \subseteq I \subseteq S_1 \cap \cdots \cap S_n$$

and we will write $I \longmapsto (S_1, \ldots, S_n)$*. The ideals* S_i *are called the* components *of the split* S*. The split is called* proper *if* I *is a proper subset of each of its components, it is called* minimal *if it is proper and if for every component* S_i *the* $(n-1)$*-uple* $(S_1, \ldots S_{j-1}, S_{j+1}, \ldots, S_n)$ *is not a split of* I*. Finally it is called* unambiguous *if for* $i \neq j$ *we have* $S_i + S_j = \mathbf{A}$*.*

Remark 7 Let S be a proper split of the ideal I, it has at least two components. If S is furthermore minimal then none of its components is $\mathbf{A}$. Let us note that we can *extract* a minimal split from every split of an ideal I and that a minimal split can have components that compare with respect to inclusion. For that we can consider in $\mathbf{K}[X, Y]$, where $\mathbf{K}$ is a field, the split $\langle X^2, X + Y\rangle \longmapsto (\langle X, Y\rangle, \langle X, Y\rangle)$. We can also verify that every minimal split of I has no multiple components if and only if I is radical.

Remark 8 Let us first note that if $I \longmapsto (S_1, S_2)$ and $S_2 \longmapsto (S_{2,1}, S_{2,2})$ are (unambiguous) splits then $I \longmapsto (S_1, S_{2,1}, S_{2,2})$ is a (unambiguous) split. Let $S = (S_1, \ldots, S_n)$ be an unambiguous split of I. We have $\mathbf{A}/I \cong \mathbf{A}/S_1 \times \cdots \times \mathbf{A}/S_n$. Conversely if $\mathbf{A}/I$ is a product-ring (with no factor being $\mathbf{A}$ or the trivial ring) then I has unambiguous splits (which are proper and have no component equal to $\mathbf{A}$). If $\mathbf{A}$ is furthermore Noetherian and if S is an unambiguous and proper split we can verify that S is minimal.

Remark 9 Let $\mathbf{A}$ is a polynomial ring over a field $\mathbf{K}$, let $\mathbf{L}$ be an algebraic closure of $\mathbf{K}$ and let $I \longmapsto (S_1, \ldots, S_n)$ we have:

$$\mathbf{V}_{\mathbf{L}}(I) = \bigcup_{i=1}^{n} \mathbf{V}_{\mathbf{L}}(S_i).$$

Moreover if I is a zero dimensional non primary ideal then every minimal primary decomposition of I is an unambiguous split of I.

Remark 10 We can see that the ideal I is prime if and only if it has no proper splits. Let I be an ideal which is not prime and $a, b \in \mathbf{A} \setminus I$ such that $ab \in I$. The couple $S = (I + \langle a\rangle, I + \langle b\rangle)$ is a proper split of I that we will call *fundamental*. Let $\mathbf{A}$ be a Noetherian ring, replacing in S every component which is not a prime ideal by a fundamental split we obtain a split $P = (P_1, \ldots, P_n)$ where the P_i are prime ideals. Removing every redundant P_i we can assume P minimal. If $\mathbf{A}$ is Artinian then the P_i are maximal. If I is radical the P_i are pairwise different. Thus under the two hypothesis P is unambiguous and $\mathbf{A}/I$ is a product of fields. Combining with proposition 2 we have:

Proposition 3 *Let* $\mathbf{A}$ *a finite type* $\mathbf{K}$*-algebra over a field* $\mathbf{K}$ *and* I *a zero dimensional radical ideal of* $\mathbf{A}$*, every couple of polynomials of* $(\mathbf{A}/I)[X]$ *has quasi-monic gcds.*

4 Quasi-inverse and Gcd Computations over an Algebraic Tower of Simple Extensions

4.1 Triangular Sets

Let $\mathbf{R}$ be an integral domain, $\mathbf{K}$ its fraction field, n a positive integer and V a set of n ordered variables $X_1 < X_2 < \cdots < X_n$. For $1 \leq i \leq n$, we write $\mathbf{R}_i = \mathbf{R}[X_1, \ldots, X_i]$ and $\mathbf{P}_i = \mathbf{K}[X_1, \ldots, X_i]$.

Notations 2 Let $p \in \mathbf{R}_n$, $p \neq 0$, we write $\mathsf{deg}(p, v)$ for the degree of p with respect to the variable v, we denote by $\mathsf{var}(p)$ the subset of variables v of V for which $\mathsf{deg}(p, v) > 0$. For $p \notin \mathbf{R}$, we denote by $\mathsf{mvar}(p)$ the highest variable of $\mathsf{var}(p)$ and call it its *main variable*, we then write $\mathsf{mdeg}(p) = \mathsf{deg}(p, \mathsf{mvar}(p))$. We call *initial* of p and denote by $\mathsf{init}(p)$ the leading coefficient of p as an univariate polynomial in $\mathsf{mvar}(p)$, we call *tail* of p and denote it by $\mathsf{tail}(p)$ the polynomial defined by $p = \mathsf{init}(p)\, \mathsf{mvar}(p)^{\mathsf{mdeg}(p)} + \mathsf{tail}(p)$. For $E \subseteq \mathbf{R}_n$ we write $\mathsf{var}(E) = \{v \in V \mid (\exists p \in E) \mid \mathsf{deg}(p, v) > 0\}$.

Definition 6 *A subset T of $\mathbf{R}_n$ is called a* triangular set *is every polynomial of T is non-constant with initial in $\mathbf{R}$ and if $p \mapsto \mathsf{mvar}(p)$ defines a bijection between T and $\mathsf{var}(T)$. We will say that $v \in V$ is* algebraic *in T if $v \in \mathsf{mvar}(T)$.*

Notations 3 Let T be a non-empty triangular set of $\mathbf{R}_n$. We will call $\mathsf{first}(T)$ the polynomial of T with greatest main variable and $\mathsf{rest}(T) = T \setminus \{\mathsf{first}(T)\}$. Let $v \in \mathsf{mvar}(T)$, T_v will be the the polynomial of T such that $\mathsf{mvar}(p) = v$. We define $T_v^- = \{p \in T \mid \mathsf{mvar}(p) < v\}$ and $T_v^+ = \{p \in T \mid \mathsf{mvar}(p) > v\}$. We will call degree of T the tuple $\mathsf{deg}(T)$ of $\mathbb{N}^n$ defined by $\mathsf{deg}(T) = (d_1, \ldots, d_n)$, where $d_i = 0$ if $X_i \notin var(T)$ and $d_i = \mathsf{mdeg}(\, T_{X_i})$ otherwise. Finally for $\alpha = (\alpha_1, \ldots, \alpha_n)$ in $\mathbb{N}^n$ we write $|\,\alpha\,| = \alpha_1 + \cdots + \alpha_n$.

Definition 7 *Let $a, b \in \mathbf{R}_n$ such that $b \notin \mathbf{R}$, $\mathsf{prem}(a, b)$ and $\mathsf{pquo}(a, b)$ will be respectively the* pseudo-remainder *and* pseudo-quotient *of a by b when interpreting them as univariate polynomials in $\mathsf{mvar}(b)$. Let T be a triangular set of $\mathbf{R}_n$, we call* pseudo-remainder *of a by T the polynomial $\mathsf{prem}(a, T)$ equal to a if T is empty and defined by $\mathsf{prem}(a, T) = \mathsf{prem}(\mathsf{prem}(a, \mathsf{first}(T)), \mathsf{rest}(T))$ otherwise. An element is said to be T-*reduced *if $\mathsf{prem}(a, T) = a$*

Let $a \in \mathbf{R}_n$ and $T = \{p_1, \ldots, p_m\}$ a triangular set of $\mathbf{R}_n$. Let l_i be the initial de p_i, E the set of these initials, S_E the multiplicatively stable part of $\mathbf{nz}(\mathbf{R})$ generated by E and $\mathbf{M}_T$ the $\mathbf{R}$-submodule of $\mathbf{R}_n$ of elements that are T-reduced. There exist integers $e_1, \ldots, e_m$ and polynomials $q_1, \ldots, q_m$ of $\mathbf{R}_n$ such that:

$$l_1^{e_1} \cdots l_m^{e_m} a \;=\; q_1 p_1 + \cdots + q_m p_m + \mathsf{prem}(p, T) \tag{5}$$

Notations 4 $\mathsf{modulo}(a, T)$ will be the element of $\mathrm{S}_E{}^{-1}\ \mathbf{M}_T$ defined by

$$\mathsf{modulo}(a, T) = \mathsf{prem}(a, T) / (l_1^{e_1} \cdots l_m^{e_m})$$

Every triangular set is a Gröbner basis for the pure lexicographical ordering and $\mathsf{prem}(a,T)$ is reduced with respect to T (in the Gröbner basis meaning). So $\mathsf{modulo}(a,T)$ is the normal form of a with respect to T (viewing a and T in $\mathbf{P}_n$). Thus, the $\mathbf{R}$-algebras $\mathbf{R}_n/T$ and $\mathrm{S}_E{}^{-1}\ \mathbf{M}_T$ are isomorphic and $\overline{a}^T$ identifies to $\mathsf{modulo}(a,T)$. Moreover $\mathsf{modulo}(a,T)$ computes more efficiently than using (5).

Notations 5 Let $T = \{p_1,\ldots,p_m\}$ be a triangular set of $\mathbf{R}_n$ with $\mathsf{mvar}(p_i) < \mathsf{mvar}(p_{i+1})$. We define $\mathsf{reduce}(T) = \{p_1{}',\ldots,p_m{}'\}$ with $p_1{}' = p_1$ and $p_{i+1}{}' = \mathsf{numer}(\mathsf{modulo}(p_{i+1},\{p_1{}',\ldots,p_i{}'\}))$. Let $E \subseteq \mathbf{R}_n$ such that $T \cup E$ is a triangular set of $\mathbf{R}_n$, we write $T + E = \mathsf{reduce}(T \cup E)$ and say that T is *auto-reduced* if $\mathsf{reduce}(T) = T$.

4.2 Algorithm Specifications

We have two kinds of algorithms. First, we have those which *try to compute* a gcd or a quasi-inverse modulo a triangular set T (we will define later this terminology) and may return a split of T (their names are suffixed with ElseSplit). Then we have those which compute a gcd or a quasi-inverse using the first ones; they return lists like $[[r_1,T_1],\ldots,[r_m,T_m]]$ where the T_i are triangular sets such that $T \longmapsto (T_1,\ldots,T_m)$ and where r_i is the result modulo T_i.

To describe those algorithms we will first give their input and output specifications. For some algorithm called MethodWhichComputes working with objects $a,b,T,\ldots$, the input specifications are denoted by $H_{mwc}(a,b,T,\ldots)$ and the output ones by $S_{mwc}(a,b,T,\ldots)$. Then, in the following subsections, we will give the corresponding procedures. Up to few modifications they are AXIOM programs. For that reason they may appear rather *technical*. But, for the same reason, as there is no *mysterious routine* they may easily allow an implementation in an other programming language.

Notations 6 Throughout this section T is an auto-reduced triangular set (a "a.r.t.s." for short). We put $\mathsf{var}(T) = \{v_1,\ldots,v_m\}$, $\mathbf{B} = \mathbf{K}[v_1,\ldots,v_m]$ and $\mathbf{A} = \mathbf{B}/T$. Let $v \in V$ and $p,p_1,p_2 \in \mathbf{R}_n$. We will denote by $H_{qres}(p,T)$ the condition $\mathsf{var}(p) \subseteq \mathsf{var}(T)$ and $H_{srges}(p_1,p_2,v,T)$ the conjunction of:

(i) $(\forall w \in \mathsf{var}(T))\ w < v$
(ii) $\mathsf{deg}(p_1,v) \geq \mathsf{deg}(p_2,v) > 0$
(iii) $(\mathsf{var}(p_1) \cup \mathsf{var}(p_2)) \setminus \{v\} \subseteq \mathsf{var}(T)$
(iv) $\mathsf{modulo}(p_1,T) = p_1$ and $\mathsf{modulo}(p_2,T) = p_2$

Definition 8 *We assume that* $H_{srges}(p_1,p_2,v,T)$ *is satisfied. Let* $g \in \mathbf{R}_n$ *be* T*-reduced and satisfying* $\mathsf{var}(g) \subseteq \mathsf{var}(p_1) \cup \mathsf{var}(p_2)$*. Since these polynomials identify to elements of* $\mathbf{A}[v]$*, we will say that* g *is a* gcd modulo T *of* p_1 *and* p_2 *if* g *is a gcd of* p_1 *and* p_2 *as in definition* 1 *with* $\mathsf{mcoeff}(g) \in \mathbf{R}$*. We now assume that* $H_{qres}(p,T)$ *is satisfied. A couple* $(q,r) \in \mathbf{R}_n \times \mathbf{R}$ *is a* quasi-inverse of p modulo T *if* q *is* T*-reduced and* (q,r) *is a quasi-inverse as in definition* 3.

Output specifications are the following:

$(S_{qres}(T))$ **:** For $p \in \mathbf{R}_n$, if $H_{qres}(p, T)$ then quasiRecipElseSplit (p, T) returns

- either $(q, r) \in \mathbf{R}_n \times \mathbf{R}$ a quasi-inverse of p modulo T ,
- either two a.r.t.s. T_1, T_2 such that $T \longmapsto (T_1, T_2)$.

$(S_{srges}(T))$ **:** For every $(p_1, p_2, v) \in {\mathbf{R}_n}^2 \times V$, if $H_{srges}(p_1, p_2, v, T)$ then subResGcdElseSplit$(p_1,\, p_2,\, v,\, T)$ returns

- either a T-reduced $g \in \mathbf{R}_n$ which is a gcd modulo T of p_1 and p_2,
- either two a.r.t.s. T_1, T_2 such that $T \longmapsto (T_1, T_2)$.

$(S_{esrges}(T))$ **:** For every $(p_1, p_2, v) \in {\mathbf{R}_n}^2 \times V$, if $H_{srges}(p_1, p_2, v, T)$ then extendedSubResGcdElseSplit (p_1, p_2, v, T) returns

- either T-reduced $(g, f, k) \in \mathbf{R}_n^3$ such that modulo$(f p_1 + k p_2, T) = g$, with g being a gcd modulo T of p_1 and p_2,
- either two a.r.t.s. T_1, T_2 such that $T \longmapsto (T_1, T_2)$.

$(S_{srg}(T))$ **:** For every $(p_1, p_2, v) \in {\mathbf{R}_n}^2 \times V$, if $H_{srges}(p_1, p_2, v, T)$ then subResGcd (p_1, p_2, v, T) returns a list $[[g_1, T_1], \ldots, [g_m, T_m]]$ where $g_i \in \mathbf{R}_n$ and T_i are a.r.t.s. such that

- subResGcdElseSplit (p_1, p_2, v, T_i) returns g_i
- $T \longmapsto (T_1, \ldots, T_m)$

$(S_c(T))$ **:** For every $p \in \mathbf{R}_n$, if $H_{qres}(p, T)$ then quasiRecip (p, T) returns a list $[[[\widehat{p}_1, \| p \|_1], T_1], \ldots, [\widehat{p}_m, \| p \|_m], T_m]]$, where $\widehat{p}_i \in \mathbf{R}_n$, and T_i is an a.r.t.s. such that

- quasiRecipElseSplit (p, T_i) returns $(\widehat{p}_i, \| p \|_i)$
- $T \longmapsto (T_1, \ldots, T_m)$

We also have an operation called split? which takes as argument any result of the first three operations and returning true if and only if it is a split.

Finally when T is radical as an ideal of $\mathbf{P}_n$, all computed splits will be unambiguous. Thus, in that case, the pairs appearing in the result produced by subResGcd (p_1, p_2, v, T) can be recombined to get a polynomial g which is a gcd modulo T of p_1 and p_2. To do so, use the Chinese remainders algorithm and the extendedSubResGcdElseSplit algorithm. Look at the propositions and at their proofs in the next subsection to understand how this can be done.

4.3 The Algorithm QuasiRecipElseSplit

quasiRecipElseSplit $(p, T) ==$

  $p/s :=$ modulo(p, T)
  $p \in \mathbf{R} \hookrightarrow$ **return** $([s, p])$
  $u :=$ mvar(p)
  $esrges :=$ extendedSubResGcdElseSplit ($T_u, p, u,\ T_u^-$)
  if split?$(esrges)$
  then
    $[T_1, T_2] := esrges$
    return $[T_1 +\ T_u +\ T_u^+, T_2 +\ T_u +\ T_u^+]$
  else
    $[f, k, g] := esrges$ { *we have:* modulo$(f\ \ T_u\ +\ k\, p,\ T_u^-)\ =\ g$ }
    if deg$(g, u) > 0$

then
  $q :=$ pquo(T_u, g)
  return [$T_u^- + \{g\} + T_u^+$, $T_u^- + \{q\} + T_u^+$]
else { *we have:* $g \in \mathbf{nz}(\mathbf{R})$ }
  return [$(k\ s)$, g]

Proposition 4 *If* $S_{qres}(T)$ *and* $S_{esrges}(T_u^-)$ *hold and if* quasiRecipElseSplit (p, T) *is not a split of* T*, putting* $(\widehat{p}, \| p \|) =$ quasiRecipElseSplit(p, T) *we get:*

(i) modulo$(p, T) = 0 \iff \| p \| = 0$
(ii) modulo$(p, T) \neq 0 \iff \overline{p}^T \in \mathbf{nz}(\mathbf{A})$

The proof of the previous proposition is easy. More generally our algorithms induce a constructive proof of the fact that if T is a radical ideal of $\mathbf{P}_n$ an element of $\mathbf{A}$ is either a zero-divisor either invertible. Furthermore, this statement shows that the computation of a quasi-inverse modulo T leads to that of a gcd modulo a triangular set.

Proposition 5 *If the ideal generated by* T *in* $\mathbf{P}_n$ *is radical, if* var$(p) \subseteq$ var(T) *and if* $\overline{p}^T \in \mathbf{nz}(\mathbf{A})$ *then it exists* $q \in \mathbf{P}_n$ *such that* $\overline{pq}^T = 1$*. Moreover the computation of* q *can be reduced to the following problem: for* $a, b \in \mathbf{R}_n$ *and* $w \in$ var(T)*, such that* $H_{srges}(a, b, w, T_w^-)$ *holds, find* $u, v, d \in \mathbf{R}_n$ *with all variables except* v *being algebraic in* T_w^-*,* d *being a gcd modulo* T_w^- *of* a *and* b *and such that* modulo$(au + bv, T_w^-) = d$*.*

Proof. $\triangleright$ By induction on $|$deg$(T)|$. If $|$deg$(T)| = 0$ or if $p \in \mathbf{R}$, we are done. Otherwise let $u =$ mvar(p), $W =$ var(T_u^-) $\mathbf{B'} = \mathbf{K}[W]$ and $\mathbf{A'} = \mathbf{B'}/T_u^-$. Since $\mathbf{Id}_{\mathbf{K}[W]}(T_u^-)$ is radical and zero dimensional there exists $d \in \mathbf{B'}[u]$ with:

$$\langle \overline{p}^{\,T_u^-},\ T_u \rangle_{\mathbf{q}(\mathbf{A'})[u]} \;=\; \langle \overline{d}^{\,T_u^-} \rangle_{\mathbf{q}(\mathbf{A'})[u]} \qquad (6)$$

with init$(\overline{d}^{\,T_u^-}) \in \mathbf{nz}(\mathbf{A'})$ (see prop. 3). Since $\overline{p}^T \in \mathbf{nz}(\mathbf{A})$ we can see that deg$(d, u) = 0$. By writing the Bézout relation induced by (6) and using induction hypothesis for T_u^- we finish the proof $\triangleleft$. From what we deduce (using prop. 3):

Proposition 6 *If the ideal generated by* T *in* $\mathbf{P}_n$ *is radical and if* $H_{srges}(p_1, p_2, v, T)$ *holds, then* p_1 *and* p_2 *have a gcd modulo* T *in* $\mathbf{R}_n$*.*

Proposition 7 *Assume that* $S_{qres}(T)$ *and* $S_{esrges}(T_u^-)$ *hold, that the ideal generated by* T *in* $\mathbf{P}_n$ *is radical and that* quasiRecipElseSplit (p, T) *is a split* (T_1, T_2) *of* T *then, interpreting all ideals in* $\mathbf{P}_n$*, this split is unambiguous.*

Proof. $\triangleright$ We use the algorithm notations and W, $\mathbf{B'}$ and $\mathbf{A'}$ as above. We show that $\mathbf{Id}_{\mathbf{P}_n}(T_u^-) + \langle g, q \rangle_{\mathbf{P}_n} = \mathbf{P}_n$. By prop. 6 there exists $d \in \mathbf{B'}[u] \cap \mathbf{R}_n$ with:

$$\langle \overline{g}^{\,T_u^-}, \overline{q}^{\,T_u^-} \rangle_{\mathbf{q}(\mathbf{A'})[u]} \;=\; \langle \overline{d}^{\,T_u^-} \rangle_{\mathbf{q}(\mathbf{A'})[u]} \qquad (7)$$

and $\mathsf{init}(d) \in \mathbf{nz}(\mathbf{R})$. So there exists $\alpha, \beta \in \mathbf{nz}(\mathbf{A'})$ and $g', q' \in \mathbf{B'}[u]$ with $\alpha\overline{g}^{\,T_u^-} = \overline{dg'}^{\,T_u^-}$ and $\beta\overline{q}^{\,T_u^-} = \overline{dq'}^{\,T_u^-}$. By $S_{esrges}(T_u^-)$ together with the formula defining q, the polynomials g and q have their initials in $\mathbf{nz}(\mathbf{R})$. It follows that the initials of $\overline{g'}^{\,T_u^-}$ and $\overline{q'}^{\,T_u^-}$ belong to $\mathbf{nz}(\mathbf{A'})$.

Since $\mathbf{Id}_{\mathbf{P}_n}(T)$ is radical we can verify that $\mathsf{deg}(g', u) > 0$ and $\mathsf{deg}(q', u) > 0$. Similarly, if we had $\mathsf{deg}(d, u) > 0$, then the polynomial $f = g'q'd$ would not be in $\mathbf{Id}_{\mathbf{P}_n}(T)$ (since $\mathsf{init}(\overline{f}^{\,T_u^-}) \in \mathbf{nz}(\mathbf{A'})$ and $\mathsf{deg}(f, u) < \mathsf{deg}(T_u, u)$) but f^2 would be inside. It follows that $d \in \mathbf{nz}(\mathbf{R})$.

We now finish writing the Bézout relation from (7) and using proposition 5 $\triangleleft$.

4.4 The Algorithms SubResGcdElseSplit and ExtendedSubResGcdElseSplit

Let $v \in V$ and $p' \in \mathbf{R}_n$ be T-reduced with $(\forall u \in \mathsf{var}(T))\, u < v$ and $(\mathsf{var}(p) \setminus \{v\}) \subseteq \mathsf{var}(T)$. We define the normalizeElseSplit operation by :

normalizeElseSplit (p', v, T) ==
    $c := \mathsf{mcoeff}(p', v)$
    $c \in \mathbf{R} \hookrightarrow$ **return** p'
    $qres :=$ quasiRecipElseSplit (c, T)
    split?$(qres) \hookrightarrow$ **return** $qres$
    $(\widehat{c}, \parallel c \parallel) := qres$
    $\mathsf{deg}(p', v) = 0 \hookrightarrow$ **return** $\parallel c \parallel$
    return polPart(modulo($\parallel c \parallel \mathsf{mvar}(p')^{\mathsf{mdeg}(p')} + \widehat{c}\,\mathsf{tail}(p'), T$))

Proposition 8 *Assume $S_{qres}(T)$ holds and* normalizeElseSplit (p', v, T) *is not a split. Writing p'' =* normalizeElseSplit(p', v, T) *we have* $\mathsf{mcoeff}(p'', v) \in \mathbf{R}$ *and* $\mathsf{deg}(p', v) = \mathsf{deg}(p'', v)$.

This follows from proposition 4 and the algorithm subResGcdElseSplit is :

subResGcdElseSplit (p_1, p_2, v, T) ==
    $np2 :=$ normalizeElseSplit (p_2, v, T)
    split?$(np2) \hookrightarrow$ **return** $np2$
    $p_2 := np2$
    $\delta := \mathsf{mdeg}(p_1)$ - $\mathsf{mdeg}(p_2)$; $\psi : \mathbf{R} := -1$; $\beta : \mathbf{R} := (-1)^{(\delta+1)}$
    $p_3 :=$ numer(modulo(prem$(p_1, p_2), T$))
    while $(p_3 \neq 0)$ **repeat**
        $\alpha : \mathbf{R} := \mathsf{mcoeff}(p_2, v)$
        $p_3 := p_3$ exquo β ; $(p_1, p_2) := (p_2, p_3)$
        $np2 :=$ normalizeElseSplit (p_2, v, T)
        split?$(np2) \hookrightarrow$ **return** $np2$
        $p_2 := np2$
        if $\mathsf{deg}(p_2, v) > 0$
            then

if $\delta > 0$ **then**
$newpsi := -\alpha$
for i **in** $2..\delta$ **repeat**
$newpsi := (-\alpha\, newpsi)$ exquo ψ
$\psi := newpsi$
$\delta :=$ mdeg(p_1) - mdeg(p_2) ; $\beta := -\alpha\, \psi^{\delta}$
$p_3 :=$ numer(modulo(prem$(p_1, p_2), T$))
else
$p_3 := 0$
return p_2

Remark 11 The extendedSubResGcdElseSplit algorithm is the *extended version* of the previous algorithm. It uses variables l_1, l_2, u_1, u_2, w_1 and w_2 of $\mathrm{S}_E{}^{-1}\ \mathbf{M}_T$ such that modulo$(u_1 a + w_2 b, T) = l_1$, modulo$(u_2 a + w_2 b, T) = l_2$ and $p_i =$ numer(l_i), where a and b are the input values of the polynomials p_1 and p_2 respectively. During the computations, the variables $l_1, \ldots, w_2$ are updated to maintain those relations. For a practical implementation, let us precise how we manage the fractions of $\mathrm{S}_E{}^{-1}\ \mathbf{M}_T$. Assume that $\mathbf{R}$ is a factorial ring with an effective gcd operation. Thus each element l of $\mathrm{S}_E{}^{-1}\ \mathbf{M}_T$ has a canonical form, i.e. numPart(l) is the content of numer(l) and numPart(l) and denom(l) are relatively prime, using $\mathbf{nz}(\mathbf{R})$ instead of S_E (for efficiency). Then, we manage those fractions in a *lazy way*. More precisely, in the extendedSubResGcdElseSplit algorithm, fractions are not stored in their canonical form. (the exact divisions by β could fail). We only simplify the final fractions l_2, u_2, w_2 before cleaning their denominators (i.e. putting the same denominator in those three fractions and removing it) to get the output value when it is not a split.

normalizeElseSplit (l, u, w, v, T) ==
$c :=$ mcoeff(polPart$(l), v$)
$c \in \mathbf{R} \hookrightarrow$ **return** $[l, u, w]$
$ces :=$ quasiRecipElseSplit (c, T)
split?$(ces) \hookrightarrow$ **return** ces
$(\widehat{c}, \| c \|) := ces$
$(l', u', w') :=$ (modulo$(\bar{c}\, l, T)$, modulo$(\widehat{c}\, u, T)$, modulo$(\bar{c}\, w, T)$)
return $[l', u', w']$

extendedSubResGcdElseSplit (p_1, p_2, v, T) ==
$(l_1, l_2) := (p_1/1, p_2/1)$
$(u_1, u_2) := (1/1, 0/1)$
$(w_1, w_2) := (0/1, 1/1)$
$luw_2 :=$ normalizeElseSplit (l_2, u_2, w_2, v, T)
split?$(luw_2) \hookrightarrow$ **return** luw_2
$[l_2, u_2, w_2] := luw_2$; $p_2 :=$ numer(l_2)
$\psi :=$ -1 ; $\delta :=$ mdeg(p_1) - mdeg(p_2) ; $\beta :=$ (-1)**$(\delta$+1)
$[p_3, q_3] :=$ [prem(p_1, p_2), pquo(p_1, p_2)]
$l_3 :=$ modulo(p_3, T)

while $(l_3 \neq 0)$ **repeat**
$\alpha :=$ mcoeff(p_2, v) ; $\lambda := \alpha^{\delta+1}$
$l_3 :=$ (numer(l_3) exquo β)/denom(l_3)
$l_4 :=$ modulo(q_3, T) ; $p_3 :=$ numer(l_3)
$(s_1, s_2) :=$ (denom(l_1), denom(l_2))
$u_3 :=$ modulo$(\lambda\, s_1\, u_1 \;-\; l_4\, s_2\, u_2, T)$
$u_3 :=$ (numer(u_3) exquo β)/denom(u_3)
$w_3 :=$ modulo$(\lambda\, s_1\, w_1 \;-\; l_4\, s_2\, w_2, T)$
$w_3 :=$ (numer(w_3) exquo β)/denom(w_3)
$(l_1, l_2) := (l_2, l_3)$
$(p_1, p_2) := (p_2, p_3)$
$(u_1, u_2) := (u_2, u_3)$
$(w_1, w_2) := (w_2, w_3)$
$luw_2 :=$ normalizeElseSplit (l_2, u_2, w_2, v, T)
split?$(luw_2) \hookrightarrow$ **return** luw_2
$[l_2, u_2, w_2] := luw_2$; $p_2 :=$ numer(l_2)
if deg$(p_2, v) > 0$
then
if $\delta > 0$ **then**
$newpsi := -\alpha$
for i **in** $2..\delta$ **repeat**
$newpsi := (-\alpha * newpsi)$ exquo ψ
$\psi := newpsi$
$\delta :=$ mdeg(p_1) - mdeg(p_2)
$\beta := -\alpha\, \psi^{\delta}$
$[p_3, q_3] :=$ [prem(p_1, p_2), pquo(p_1, p_2)]
$l_3 :=$ modulo(p_3, T)
else
$l_3 := 0$
return clearDenoms (u_2, w_2, l_2)

4.5 The Algorithms SubResGcd and QuasiRecip

They deduce easily from previous ones managing the discussion. For instance:

subResGcd $(p_1, p_2, v, T) ==$
$toSee := [\,[\,[p_1, p_2], T]\,]$; $toSave := [\,]$
while not empty? $toSee$ **repeat**
$[[a_j, b_j], T_j] :=$ first $toSee$; $toSee :=$ rest $toSee$
$res :=$ subResGcdElseSplit (a_j, b_j, v, T_j)
if not split?(res)
then
$toSave :=$ cons$([res, T_j], toSave)$
else
for T **in** res **repeat**
$a_j :=$ polPart(modulo(a_j, T)) ; $b_j :=$ polPart(modulo(b_j, T))

```
        if (deg(a_j, v) = 0) or (deg(b_j, v) = 0)
          then
            toSave := cons([1, T_j], toSave)
          else
            if mdeg(a_j) < mdeg(b_j) then (a_j, b_j) := (b_j, a_j)
            toSee := cons([[a_j, b_j], T], toSee)
return toSave
```

4.6 Algorithms Correction

We now verify that the algorithms satisfy their output specifications.

Proposition 9 *The algorithms* quasiRecipElseSplit, subResultantGcdElseSplit *and* extendedSubResultantGcdElseSplit *satisfy respectively* $S_{qres}(T)$, $S_{srges}(T)$, *and* $S_{esrges}(T)$ *for all* T.

Proof. $\triangleright$ We saw that whenever $S_{srges}(T)$ holds, $S_{esrges}(T)$ also holds. It suffices to show by induction on $|\,\mathsf{deg}(T)\,|$ that $S_{qres}(T)$ and $S_{srges}(T)$ both hold. Let p, p_1, p_2 in $\mathbf{R}_n$ and v in V such that $H_{srges}(p_1, p_2, v, T)$ and $H_{qres}(p, T)$ hold.

If $|\,\mathsf{deg}(T)\,| = 0$ (T is empty), then from $H_{qres}(p, T)$, $p \in \mathbf{R}$ and $\mathsf{modulo}(p, T)$ is $p/1$. So subResultantGcdElseSplit is the usual subresultant algorithm which satisfies $S_{srges}(T)$.

On the contrary if $|\,\mathsf{deg}(T)\,| > 0$, and using the algorithm notations we write $p/s = \mathsf{modulo}(p, T)$, let us look at quasiRecipElseSplit (p, T). We can verify that $H_{esrges}(\ T_u, p, u,\ T_u^-)$ holds, thus $S_{esrges}(\ T_u^-)$ applies.

If $esrges$ is a "f.s.a.r.t.s." (fundamental split into a.r.t.s.) of T_u, then quasiRecipElseSplit (p, T) is also a f.s.a.r.t.s. of T.

If $\mathsf{deg}(g, u) > 0$, we also obtain a f.s.a.r.t.s. of T since g pseudo-divises T_u modulo T_u^-. Since $\mathsf{init}(g) \in \mathbf{nz}(\mathbf{R})$, there exists $t \in \mathbf{nz}(\mathbf{R})$ with $\mathsf{modulo}(gq,\ T_u^-) = t\ T_u$ and we have $\mathsf{init}(q) \in \mathbf{nz}(\mathbf{R})$. Thus gq belongs to the ideal of $\mathbf{P}_n$ generated by T but none of them does.

Finally, if $\mathsf{deg}(g, u) = 0$ and if a is the original input polynomial we have $\mathsf{modulo}(ask, T) = \mathsf{modulo}(pk, T) = g$ and, since $g \in \mathbf{nz}(\mathbf{R})$ $S_{qres}(T)$ holds.

We now deduce $S_{srges}(T)$ by looking at subResGcdElseSplit (p_1, p_2, v, T).

If at any step normalizeElseSplit (p_2, v, T) is a f.s.a.r.t.s., then the same holds for subResGcdElseSplit (p_1, p_2, v, T). We will assume that this is not the case.

By induction hypothesis and since $S_{qres}(T)$ holds, we are under assumptions of proposition 4 and proposotion 8 applies.

Thus the line p_2 := $np2$ replaces $p2$ with p'_2 having same degree and initial in $\mathbf{R}$ and these are associated as polynomials of $\mathbf{q}(\mathbf{A})[v]$. Since the initals of the polynomials of T are in $\mathbf{R}$, we have $\langle p_1, p_2\rangle_{\mathbf{q}(\mathbf{A})[v]} = \langle p_2, p_3\rangle_{\mathbf{q}(\mathbf{A})[X]}$ and it follows that the returned p_2 is a (definition 1) gcd of the original p_1 and p_2 .

Finally it is reduced modulo T since it is the result of a normalizeElseSplit (p_2, v, T) operation $\triangleleft$.

Let us remark that if the ideal generated by T in $\mathbf{P}_n$ is radical then splits eventually appearing during computations are unambiguous (by proposition 7).

4.7 Application to Zero Dimensional System Solving

Let I be a zero dimensional ideal of $\mathbf{P}_n$ given by a lexicographical Gröbner basis G sorted by increasing leading monomials. The lextriangular algorithm (see [12]) applied to G returns a split of I with every component being a triangular set.

It works managing two lists *toSave* and *toSee*. The first one is made of already built components, the second one is a list of triplets $[T, v, H]$ where T is triangular set (under construction), v is the greatest variable algebraic in T and H is the *sublist* of G of those polynomials g such that $\mathsf{mvar}(g) > v$.

To add the first polynomial p of H to T we determine if p is quasi-monic in $(\mathbf{K}[\mathsf{var}(T)]/T)\ [w]$, where w is its main variable and ip is the initial of p. To do so, we run quasiRecip (ip, T); this leads to a *discussion* i.e. a list of pairs as described in the output specification of that operation.

For each alternative of the discussion if ip is a zero divisor, we forget p and proceed with the next polynomial of the list H. On the contrary, if ip is invertible modulo T, we add p to T and proceed replacing H by its polynomials with main variable bigger than w.

Correctness follows from a theorem of Gianni and Kalkbrenner (see [8, 10, 12]):

```
lextriangular G ==
  toSee := [[[first(G)], mvar(first(G)), rest(G)]]
  toSave := []
  while not empty? toSee repeat
     [T, v, H] := first toSee
     toSee := rest toSee
     p := first H ; ip := init(p)
     lq := quasiRecip (ip, T)
     while not empty? lq repeat
        [[q̂, ‖ q ‖], T'] := first lq
        lq := rest lq
        if ‖ q ‖ = 0
          then
            toSee := cons([T', v, rest(H)], toSee)
          else
            newp := ‖ q ‖ mvar(p)^mdeg(p) + q̂ tail(p)
            newp := polPart(numer(modulo(newp, T')))
            newT := cons(newp, T')
            newv := mvar(newp)
            newH := [g ∈ H | mvar(g) > newv]
            if empty? newH
              then
                toSave := cons(newT, toSave)
              else
                toSee := cons([newT, newv, newH], toSee)
  return toSave
```

Acknowledgments: we wish to thank D. Lazard and the referees for their hepfull suggestions and comments.

References

1. J. Backelin and R. Frberg. How we proved that there are exactly 924 cyclic 7-roots. In S. M. Watt, editor, *Proc. ISSAC'91*, pages 103–111. ACM, 1991.
2. B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* PhD thesis, Innsbruck, 1965.
3. D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms.* Spinger-Verlag, 1991.
4. J. Della Dora, C. Discrescenzo, and D. Duval. About a new method method for computing in algebraic number fields. In *Proc. EUROCAL 85 Vol. 2*, volume 204 of *Lect. Notes in Comp. Sci.*, pages 289–290. Springer-Verlag, 1985.
5. D. Duval. *Questions Relatives au Calcul Formel avec des Nombres Algbriques.* Universit de Grenoble, 1987. Thse d'Etat.
6. J.C. Faugère. *Résolution des systèmes d'équations algébriques.* Université Paris 6, 1994. Thse de Doctorat.
7. Richard D. Jenks and Robert S. Stutor. *AXIOM, The Scientific Computation System.* Springer-Verlag, 1992.
8. M. Kalkbrener. Solving systems of algebraic equations by using Gröbner basis. European Conference on Computer Algebra, Leipzig, GDR, 1987 (J. H. Davenport, ed.). *Lecture Notes in Computer Science* , 378:282–292, 1987.
9. M. Kalkbrener. A generalized euclidean algorithm for computing triangular representations of algebraic varieties. *J. Symb. Comp.*, 15:143–167, 1993.
10. P. Gianni. Properties of Gröbner basis under specializations European Conference on Computer Algebra, Leipzig, GDR, 1987 (J. H. Davenport, ed.). *Lecture Notes in Computer Science* , 378:293–297, 1987.
11. D. Lazard. A new method for solving algebraic systems of positive dimension. *Discr. Appl. Maths*, 33:147–160, 1991.
12. D. Lazard. Solving zero-dimensional algebraic systems. *J. Symb. Comp.*, 13:117–132, 1992.
13. D. Lazard. Systems of algebraic equations (algorithms and complexity). In Eisenbud and Robbiano, editors, *Proceedings of Cortona Conference.* Cambridge University Press, 1993.
14. R. Loos. Generalized polynomial remainder sequences. In *Symbolic and Algebraic Computation*, pages 115–137. Spinger-Verlag, 1982.
15. B.L. van der Waerden. *Algebra.* Springer-Verlag, 1991. seventh edition.