A Fast Algorithm for Computing the Truncated Resultant
Guillaume Moroz, Éric Schost

To cite this version:
Guillaume Moroz, Éric Schost. A Fast Algorithm for Computing the Truncated Resultant. ISSAC ’16, Sergei A. Abramov; Eugene V. Zima, Jul 2016, Waterloo, Canada. pp.341-348, 10.1145/2930889.2930931. hal-01366386

HAL Id: hal-01366386
https://hal.science/hal-01366386
Submitted on 14 Sep 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
A Fast Algorithm for Computing the Truncated Resultant

Guillaume Moroz
Inria Nancy Grand Est
guillaume.moroz@inria.fr

Éric Schost
University of Waterloo
eschost@uwaterloo.ca

ABSTRACT
Let $P$ and $Q$ be two polynomials in $\mathbb{K}[x, y]$ with degree at most $d$, where $\mathbb{K}$ is a field. Denoting by $R \in \mathbb{K}[x]$ the resultant of $P$ and $Q$ with respect to $y$, we present an algorithm to compute $R \mod x^k$ in $\mathcal{O}(kd)$ arithmetic operations in $\mathbb{K}$, where the $\mathcal{O}$ notation indicates that we omit polylogarithmic factors. This is an improvement over state-of-the-art algorithms that require to compute $R$ in $\mathcal{O}(d^3)$ operations before computing its first $k$ coefficients.

1. INTRODUCTION
Computing the resultant of two polynomials is an ubiquitous question in symbolic computation, with applications to polynomial system solving [17], computational topology [18, 2, 12, 7, 8, 5, 22, 6], Galois theory [28, 24, 1, 23], computations with algebraic numbers [4], etc.

From the complexity viewpoint, this question admits a satisfactory answer in the simplest case of polynomials $P, Q$ with coefficients in a field $\mathbb{K}$. Euclid’s algorithm can be adapted to compute the resultant $R$ of $P$ and $Q$ in time $\mathcal{O}(d^2)$, assuming arithmetic operations in $\mathbb{K}$ are counted at unit cost (a thorough discussion of resultant algorithms based on Euclid’s algorithm is in [14]). Using fast polynomial multiplication and divide-and-conquer techniques, the Knuth-Schönhage half-gcd algorithm [21, 27] allows one to compute $R$ in time $\mathcal{O}(d)$. This is optimal, up to logarithmic factors, since the input has size $\Theta(d)$.

However, no such quasi-linear result is known in the important case of bivariate polynomials over $\mathbb{K}$, or in the very similar case of univariate polynomial with integer coefficients (in which case one would be interested in bit complexity estimates). In the former situation, suppose we consider two polynomials $P$ and $Q$ in $\mathbb{K}[x, y]$, with degree at most $d$, and we want to compute their resultant $R$ with respect to $y$, so that $R$ is in $\mathbb{K}[x]$. The polynomial $R$ has degree at most $d^2$, so both input and output can be represented using $\Theta(d^2)$ elements in $\mathbb{K}$. However, the best known algorithms to compute $R$ take $\mathcal{O}(d^3)$ operations in $\mathbb{K}$, either by means of evaluation / interpolation techniques, or in a direct manner [26].

In this paper, we are interested in the computation of the resultant $R$ of such bivariate polynomials truncated at order $k$, that is of $R \mod x^k$ for some given parameter $k$. This kind of question appears for instance in the algorithms of [17, 23], where we want two terms in the expansion, so that $k = 2$. A related example, in a slightly more involved setting, involves the evaluation of the second derivative of some subresultants, for input polynomials in $\mathbb{K}[x, y, z]$ [19].

Of course, one could simply compute $R$ itself and truncate it afterwards; however, it seems wasteful to compute all $d^2$ terms of $R$, incurring a cost of $\mathcal{O}(d^3)$, before discarding many of them. Now, for all but finitely many values $a$ in $\mathbb{K}$, it is possible to compute $R \mod (x - a)^k$ using $\mathcal{O}(d k)$ operations in $\mathbb{K}$: indeed, as soon as the non-zero subresultants of $P$ and $Q$ do not vanish at $a$, we can run the Knuth-Schönhage algorithm with coefficients truncated modulo $(x - a)^k$, without attempting to invert an element that would vanish at $a$. The running time claimed above then follows from the fact that arithmetic operations in $\mathbb{K}[x]/(x - a)^k$ can be done using $\mathcal{O}(k)$ operations in $\mathbb{K}$ (for such standard complexity results, our reference is [13]).

If however we cannot choose the expansion point, as is the case here, there is no guarantee that all divisions remain feasible in $\mathbb{K}[x]/(x^k)$; attempting to divide by an element of positive valuation would entail a loss of $x$-adic precision.

An obvious solution is to use a division-free algorithm over $\mathbb{K}[x]/(x^k)$ (by contrast, so-called fraction-free algorithms often require the base ring to be a domain). The best result we are aware of is due to Kaltofen and Villard [20], with a cost of $\mathcal{O}(d^2 698^k)$ ring operations to compute the determinant of a matrix of size $d$ over any ring, and thus $\mathcal{O}(d^2 698^k)$ operations in $\mathbb{K}$ to solve our problem.

In [10], Caruso studies the phenomenon of loss of precision in the iterative version of the subresultant algorithm. He shows that on average, if the base field is finite, this loss of precision grows linearly with the degree of the inputs. In that same reference, he also shows how to modify this algorithm to reduce the loss of precision, resulting in a cost of $\mathcal{O}(d^2(k + \delta))$ operations in $\mathbb{K}$, where $\delta$ is the maximum of the $x$-adic valuations of the non-zero leading subresultants of the input polynomials (under the assumption that these leading subresultants all have valuation less than $k/2$). When $\mathbb{K}$ is finite, the expected value of $\delta$ is $\mathcal{O}(\log(d))$, so that the average running time becomes $\mathcal{O}(d^2 k)$.

Our main result is a complexity estimate for the computation of $R \mod x^k$, using the Knuth-Schönhage divide-and-
conquer algorithm. We show that we can compute $R \mod x^k$ using $O'(dk)$ base field operations, when $K$ has characteristic zero, or at least $k$.

We proceed in three steps. First, we compute cofactors $U, V$ and an integer $t$ such that $UV + PQ = x^t \mod x^{k+1}$ holds in $\mathbb{R}_k = \mathbb{K}[x]/(x^{k+1})$; this is done in Section 3 by a suitable adaptation of the half-gcd algorithm. From this equality, we will be able to deduce a first-order linear differential equation satisfied by the resultant $R$ (Section 5). Solving this differential equation is straightforward, once an initial condition is known; Section 4 shows how to compute the first non-zero term in the resultant.

2. PRELIMINARIES

To a polynomial $P$ of $\mathbb{R}_k$, we associate a valuation $v(P)$ defined as the smallest exponent on $x$ in the monomials of $P$ for $P$ non-zero; by convention, $v(0) = k$. By Gauss’ Lemma, for any polynomials $P$ and $Q$ in $\mathbb{R}_k$, we have $v(PQ) = v(P) + v(Q)$ if the sum is less than $k$ (otherwise, $PQ = 0$).

In the algorithms of the paper, a polynomial $P$ of $\mathbb{R}_k$ is represented by the monomial $x^{v(P)}$ and the polynomial $P/x^{v(P)}$ of valuation 0. We do not take into account the boolean cost of exponent manipulations, in the sense that we assume that we can add two valuations in constant time.

We define the valuation $v(P)$ of a pair of polynomials $(P, Q)$ in $\mathbb{R}_k$ as the minimum of the valuations of $P$ and $Q$. We let $\mathbb{S}_k \subset \mathbb{R}_k$ be the set of all pairs of polynomials $(P, Q)$ with $v(P) < v(Q)$, to which we adjoint $(0, 0)$.

For $n \in \mathbb{N}$, we define $\Pi_n$ as the function from $\mathbb{R}_k$ to itself, and by extension from $\mathbb{S}_k$ to itself, such that:

$$\Pi_n(P) = P \text{ rem } x^{v(P)+n}$$

$$\Pi_n(P, Q) = (P \text{ rem } x^{v(P)} + n, Q \text{ rem } x^{v(P)+n})$$

if $v(P) + n \geq k$ in this definition, we simply replace it by $k$.

We denote by $U(\mathbb{R}_k)$ the set of invertible elements of $\mathbb{R}_k$: they are exactly the polynomials $P$ such that $v(P) = 0$ and $\Pi_1(P) \in \mathbb{K}$.

We call degree of an element of $\mathbb{K}[x]/(x^k)$ the degree of its canonical lift to $\mathbb{K}[x]$. We will often write expressions of the form $Q = P/x^{v(P)}$, for some $P$ in $\mathbb{R}_k$. Such a quotient is not unique; we remove ambiguities by requiring that all coefficients of $Q$ have degree in $x$ less than $k - v(P)$.

If $P$ is a polynomial in $\mathbb{R}_k$, then $x^i | P \in \mathbb{K}[y]$ is the coefficient of $x^i$ in it. Then, we say that $P$ is normal if $\deg_{xi}([x^i]P) > \deg_{xi}(x^iP)$ for all $i > v(P)$. Equivalently, this is the case if the leading coefficient of $P$ has the form $cx^{v(P)}$ for some non-zero $c$ in $\mathbb{K}$.

Being normal is a useful property. For instance, it allows us to define quotient and remainder in a Euclidean division in many cases: if $P \in \mathbb{R}_k$ is normal and $Q \in \mathbb{R}_k$ has valuation at least $v(P)$, then exist $U$ and $R$ in $\mathbb{R}_k$ with $\deg(R) < \deg(P)$, and such that $Q = U P + R$; $U$ is uniquely defined modulo $x^{v(P)}$, whereas $R$ is unique. We will write $Q \div_\mathbb{R}_k P$ for the unique quotient $U$ with degree in $x$ less than $k - v(P)$, and $R = Q \text{ rem }_\mathbb{R}_k P$.

The following normalization property will be essential for our algorithms. Below, $M(n)$ is a bound on the number of arithmetic operations in $K$ for computing the product of two polynomials $f, g$ in $\mathbb{K}[x]$ of degree at most $n$; we assume that $M$ satisfies the super-linearity conditions of [13, Chapter 5]. Using the Cantor-Kaltofen algorithm [9], we can take $M(n)$ in $O(n \log(n) \log \log(n))$. Using Kronecker’s substitution as in [15], multiplication or Euclidean division in degree $d$ in $\mathbb{R}_k$ can then be done in $O(M(kd))$ operations in $\mathbb{K}$.

**Lemma 1** (Normalization). For $P$ non-zero in $\mathbb{R}_k$, there exist polynomials $U$ and $T$ in $\mathbb{R}_k$ such that $T$ is normal, $U$ is a unit, with $\Pi_1(U) = 1$, and $P = UT$.

The polynomial $T$ is unique, whereas $U$ is uniquely defined modulo $x^{v(P)}$. Moreover, if $P$ has degree $d$, for $n \leq k - v(P)$, given $\Pi_n(P)$, $\Pi_n(U)$ and $\Pi_n(T)$ can be computed using $O(M(dn))$ operations in $\mathbb{K}$.

**Proof.** Up to dividing $P$ by $x^{v(P)}$, we may assume that $P$ has valuation $0$. Then, uniqueness and the corresponding algorithm are from Algorithm Q in [25]; the cost analysis is a straightforward extension of that given for Hensel step in [13, Chapter 15] using Kronecker substitution for the arithmetic operations. Once the result is known for $P/x^{v(P)}$, we recover our claim by multiplying $T$ by $x^{v(P)}$.

Given $P$ in $\mathbb{R}_k$, we denote by respectively $\text{Le}(P)$ and $N(P)$ the unique $U$ and $T$ that satisfy the above conditions, with $U$ having degree in $x$ less than $k - v(P)$; these two polynomials have degree in $y$ at most $d$, and $v(N(P)) = v(P)$. We also define $\text{Le}(P) = \Pi_n(\text{Le}(P))$ and $N_n(P) = \Pi_n(N(P))$; by the previous lemma, if $P$ has degree $d$, for any $n$, $\text{Le}(P)$ and $N_n(P)$ can be computed in time $O(M(dn))$.

Using this result, we can reduce the problem of computing the resultant of two polynomials in $\mathbb{R}_k$ to the problem of computing the resultant of two normal polynomials.

**Lemma 2.** Let $P$ and $Q$ be in $\mathbb{R}_k$ of degrees in $y$ at most $d$. Then there exist four monic polynomials $A, B, C, D$ in $\mathbb{R}_k$, with degrees at most $d$, and $u$ in $\mathbb{K}[x]/(x^k)$ such that $\text{Res}(P, Q) = u \text{Res}(A, B) \text{Res}(C, D)$. Moreover, $A, B, C, D, u$ can be computed in time $O(M(dk))$.

**Proof.** By the multiplicative property of the resultant, $\text{Res}(P, Q) = \text{Res}(\text{Le}(P), Q) \text{Res}(N(P), Q)$. Let $d_c, d_a, d_q$ be the degrees in $y$ of $\text{Le}(P), N(P)$ and $Q$ respectively.

Taking the reciprocical polynomial of $\text{Le}(P)$ and $Q$ changes at most the sign of their resultant. Let us thus define $\hat{P} = y^d \text{Le}(P)(1/y)$ and $Q = y^{-d}Q(1/y)$ and let $c_0 \in \mathbb{K}[x]/(x^k)$ be the leading coefficient of $\hat{P}$. By construction, $c_0$ is a unit, so we can define $A = \hat{P}/c_0 \in \mathbb{R}_k$, and we have $\text{Res}(P, Q) = (-1)^{d_c + d_a} c_0^{d_q} \text{Res}(A, Q)$, where $A$ is monic in $y$. Let $R$ be the remainder of the Euclidean division of $\hat{Q}$ by $A$ and let $B = R + A$; since $A$ is monic, $\text{Res}(A, Q) = \text{Res}(A, B)$.

Similarly, because $N(P)$ is normal, we can write it as $N(P) = n_0 C$, where $n_0 \in \mathbb{K}[x]/(x^k)$ is its leading coefficient and $C$ is monic in $y$; then, we have $\text{Res}(N(P), Q) = n_0^{d_q} \text{Res}(C, Q)$. Defining as above $D = (Q \text{ rem }_\mathbb{R}_k C) + C$, we deduce $\text{Res}(N(P), Q) = n_0^{d_q} \text{Res}(C, D)$, and finally

$$\text{Res}(P, Q) = (-1)^{d_c + d_q} (c_0 n_0)^{d_q} \text{Res}(A, B) \text{Res}(C, D).$$

$A$ is monic of degree at most $d$, since $\text{Le}(P)$ has degree at most $d$. The remainder $R$ has degree less than the degree of $A$, so that $B = R + A$ is monic of the same degree as $A$. The same holds for $C$ and $D$.

In terms of complexity, after computing $\text{Le}(P)$ and $N(P)$, all other operations are $O(d)$ inversions or multiplications of power series in $\mathbb{K}[x]/(x^k)$ and $O(1)$ Euclidean divisions by monic polynomials of degree at most $d$ in $\mathbb{R}_k$. Their total cost is $O(M(dk))$ operations in $\mathbb{K}$. □
3. COMPUTING PSEUDO INVERSES

In this section, we show that two polynomials $P, Q$ in $R_k$, we can compute a matrix $M = \left( \begin{smallmatrix} P & Q \\ \end{smallmatrix} \right) \in M_2(\mathbb{Z}_2)$ such that $UP + VQ$ is of the form $x^t \mod x^{t+1}$, for some integer $t$; we call $U$ and $V$ pseudo-inverses of $P$ and $Q$.

To simplify notation, for $s = (P, Q)$ and $M$ as above, we simply write $M \cdot s$ for the matrix-vector product $M \left( \begin{smallmatrix} P \\ Q \end{smallmatrix} \right)$.

3.1 The pseudo-division operator $Q$

In this subsection, we define an operator $Q : S_k \to S_k$ and study its properties. For $s = (0, 0)$, we define $Q(s) = Id$; otherwise, the construction involves three stages.

For a non-zero pair of polynomials $s = (P, Q)$ in $S_k$, define

$$\eta(s) := v(Q \text{ rem}_y N(P)) - v(P).$$

This is well-defined, as $N(P)$ has the same value as $P$.

**Lemma 3.** For an integer $n$ and in $S_k$, given $\Pi_n(s)$, we can compute $\min(\eta(s), n)$ using $O(M(dn))$ operations in $K$.

**Proof.** We start by dividing both $P$ and $Q$ by $x^P(Q)$; this does not involve any arithmetic operations. We can then compute the normalization $N_n(P) / x^P(P)$, and do the Euclidean division of $Q / x^P(P)$ by this polynomial, using coefficients taken modulo $x^P$, both in time $O(M(dn))$. The valuation of the remainder is precisely $\min(\eta(s), n)$. □

The next lemma shows a more intrinsic characterization of $\eta$. We denote by $\sigma_0 : R_k \to K[y]$ the evaluation morphism that sends $x$ as $P$.

**Lemma 4.** For any integer $t \geq 0$, with $J_t(s) = (P, Q) : x^t \oplus \Pi_t(s)$, and $I_t(s) = \sigma_0(J_1(s))$, we have

$$\begin{cases} I_t(s) = I_0(s) & \text{if } 0 \leq t < \eta(s) \\ I_t(s) \supseteq I_0(s) & \text{if } t \geq \eta(s). \end{cases}$$

**Proof.** For $t \geq v(s)$, we have $I_0(s) \subseteq I_t(s)$. Then, let $Q' = Q \text{ rem}_y N(P)$. If $0 \leq t < \eta(s)$ let $W$ be such that $x^t W = UP + VQ' \in (P, Q)$. Since $x^t Q' \mid x^t W = UP \mod x^{t+1}$, with $s$ divides divides $U$ and $x^t W = UP \mod x^{t+1}$. Thus $I_t(s) \subseteq I_0(s)$ and $I_t(s) = I_0(s)$. If $t \geq \eta(s)$, $I_t(s)$ contains the residue class of the remainder of $\Pi_t(Q) / x^t(P)$, that is a non-zero polynomial of degree less than $\deg_\nu(\Pi_1(P))$, and is not included in $I_0(s)$. □

For $s = (P, Q)$ as above, perform a Euclidean division of $Q$ by the normal polynomial $N_\nu(s)(P)$, and define the matrix

$$D_s := \left( \begin{array}{cc} x^\nu(s) & 0 \\ \text{rem}_y x^\nu(s) & \text{LC}_\nu(s)(P) \end{array} \right).$$

Then, the polynomial $Q$ defined by

$$Q := \left( \begin{array}{c} P \\ Q \end{array} \right) \cdot \left( \begin{array}{c} x^\nu(P) \\ \text{LC}_\nu(P) \end{array} \right) \text{rem}_y x^\nu(s) \text{LC}_\nu(s)(P).$$

has valuation $v(P) + \eta(s)$ and we have the inequality $\deg_\nu(\Pi_1(Q)) < \deg_\nu(\Pi_1(P)).$

Given $P, Q \in \mathbb{Z}_2[y]$ of degree at most $d$, and denoting by $G$ their monic gcd, there exists an invertible matrix $G_{(P, Q)}$ with entries in $\mathbb{Z}_2[y]$ of degree less than $d$ such that

$$G_{(P, Q)} \cdot \left( \begin{array}{c} P \\ Q \end{array} \right) = \left( \begin{array}{c} G \\ 0 \end{array} \right).$$

If $Q = 0$ then $G_{(P, Q)}$ is the identity matrix. By extension, for $s = (P, Q) \in \mathbb{Z}_2[y]$ with $v(P) = v(Q)$, we define $G_s$ as $G_{(P, Q)/(P, Q)} = G_{(P, Q)} / x^{v(Q)}$. By construction, the entries $\langle P, Q \rangle$ of $G_s$ satisfying $v(P) < v(Q)$, or $P = Q = 0$; in other words, the pairs $G_s \cdot s$ belong to $S_k$.

For a pair $s = (P, Q)$ in $S_k$ with $P \neq 0$ we define $N_s$ by

$$N_s = \left( \begin{array}{cc} \text{LC}(P)^{-1} & 0 \\ -\text{LC}(P)^{-1} & \text{LC}(P)^{-1} \text{div}_y N(P) \end{array} \right) .$$

If $P = 0$, set $N_s = Id.$ Else, $v(Q) > v(P)$ and there exists a matrix $R$ such that $N_s = Id + zR$; then, $N_s$ is invertible.

**Lemma 5.** For $s = (P, Q) \in S_k^2$, define

$$Q(s) := N_G_{D_s} \cdot D_s \cdot D_s \cdot D_s.$$

Then, if $Q(s) \cdot s$ is in $S_k$,

1. the following equality between ideals holds:

$$\langle Q(s) \cdot s \rangle = \langle s \rangle \cap \langle x^{v(s) + \eta(s)} \rangle ;$$

2. $\Pi_1(Q(s) \cdot s) = \left( \begin{array}{c} x^{\nu(s)} \cdot \text{LC}(s) \\ 0 \end{array} \right)$, where $G$ is the g.c.d. $\Pi_1(Q) / x^{\nu(s)}$ and $\Pi_1(Q) / x^{\nu(s)}$;

3. $\Pi_1(Q(s) \cdot s) = \left( \begin{array}{c} x^{\nu(s)} \cdot \text{LC}(s) \\ 0 \end{array} \right)$, where $G$ is the g.c.d. $\Pi_1(Q) / x^{\nu(s)}$ and $\Pi_1(Q) / x^{\nu(s)}$;

4. $Q(s) \cdot s$ and $Q(s)' \cdot s'$ generate the same ideal, for any pair $s'$ that generates the same ideal as $s$;

5. the entries of $Q(s) \cdot s$ have degree less than $\deg_\nu(\Pi_1(P))$.

For $s = (P, Q) \in S_k$, where $P$ and $Q$ have degree at most $d$, given $\Pi_n(s)$ for some $n \geq \eta(s)$, $\Pi_{\eta(s)}(s) / s$ can be computed using $O(M(dn)) + M(d\log(d))$ operations in $K$.

**Proof.** We saw just above the lemma that $G_{D_s} \cdot D_s \cdot D_s$ belongs to $S_k$. Since applying the matrix $N_G_{D_s} \cdot D_s$ to this vector does not change the valuations of its entries, we deduce that $Q(s) \cdot s$ is in $S_k$.

In order to prove the relation $\langle Q(s) \rangle \cdot s = \langle s \rangle \cap \langle x^{\nu(s) + \eta(s)} \rangle$, it is sufficient to prove that $\langle D_s \cdot s \rangle = \langle s \rangle \cap \langle x^{\nu(s) + \eta(s)} \rangle$, since the matrices $N_G_{D_s} \cdot D_s$ and $G_{D_s} \cdot D_s$ are invertible over $R_k$. The two polynomials in the vector $D_s \cdot s = \langle x^{\nu(s)} \rangle , Q \rangle$ are divisible by $x^{\nu(s)} + \eta(s)$, thus $\langle D_s \cdot s \rangle \subseteq \langle s \rangle \cap \langle x^{\nu(s)} + \eta(s) \rangle$.

For the other inclusion, let $W$ be a polynomial in $\langle s \rangle \cap \langle x^{\nu(s)} + \eta(s) \rangle$. Since we have $s = (P, Q) = (P, Q)$, we can write $W = UP + VQ$, for some $U, V \in R_k$. On the other hand, we know that $x^{\nu(s)} + \eta(s)$ divides $W$, and since it also divides $Q$, it divides $UP$. This implies that $x^{\nu(s)}$ divides $U$, which means that $W$ is in $\langle D_s \cdot s \rangle$. This allows us to conclude for the equality of ideals, noting that $v(s) = v(P) = v(Q)$.

The third point comes from the fact that the matrix $N_G_{D_s} \cdot D_s$ can be written $Id + zR$, so that $\Pi_1(Q(s) \cdot s) = \Pi_1(G_{D_s} \cdot D_s \cdot s) = \langle x^{\nu(s) + \eta(s)} \rangle$.

If $s$ and $s'$ generate the same ideal, $\eta(s) = \eta(s')$ (Lemma 4); using the second item, this proves point 4.

The degree property follows from the fact that in the pair $(A, B) = (Q(s) \cdot s, A)$ is normal of degree $\deg_\nu(G) < \deg_\nu(\Pi_1(P))$, and $B$ is a remainder modulo $N(A)$.

Computing $D_s$ can be done in time $O(M(d\log d(s)))$ using Lemma 1 to compute $N_\nu(s)(P)$ and $\text{LC}_\nu(s)(P)$ in time $O(M(d\log d(s)))$. The matrix-vector product that gives $Q$ is done in degree $n$ in $x$ and $d$ in $y$, in time $O(M(dn))$. Then,
we saw that $\Pi_1(\bar{Q})$ has degree less than $d$, so computing $G_{D,s}$ is an extended gcd calculation in $\mathbb{K}[y]$ that can be done in time $O(M(d)\log(d))$. Applying $G_{D,s}$ to $D_s$, $s$ takes $O(nM(d))$, and results in a matrix of degree $O(d)$ in $y$. Finally, applying $N_{G_{D,s},D_s}$ to $G_{D,s}$, $D_s$ is again a normalization at precision $n$ along with arithmetic operations that can all be done in time $O(M(dn))$.

### 3.2 An extension of the half-gcd

Our goal is now to iterate the pseudo-division $Q(s)$ until we reach an integer $t$ such that $(s) \cap (x^t) = (x^t)$. We use a divide-and-conquer algorithm, inspired by the half-gcd algorithm. If we applied this idea directly, the increase in degree in $y$ of the transition matrices would prevent us from getting a softly linear bound in the degree of the input; we will thus work modulo an equivalence relation, to control the size of the intermediate polynomials.

Consider the following equivalence relation on $S_k$: for any two pairs $(P,Q)$ and $(P',Q')$ in $S_k$, we say that $(P,Q) \sim (P',Q')$ if and only if the ideals they generate are the same. In particular, this implies that $v(P, Q) = v(P', Q')$ and that $\Pi_1(P)$ and $\Pi_1(P')$ are equal up to a constant.

Let further $H, H'$ be two functions from $S_k$ to $M_2(\mathbb{R}_k)$. Extending the equivalence property to the set of functions, we say that $H$ and $H'$ are equivalent if for all $s \in S_k$ we have $H(s) \sim H'(s)$ · $s$. We still write in this case $H \sim H'$.

**Definition 1.** (*Euclidean function*). We say that $H : S_k \rightarrow M_2(\mathbb{R}_k)$ is Euclidean if:

- for all $s, s' \in S_k$, if $s \sim s'$, then $H(s) \cdot s \sim H(s') \cdot s'$
- for all $s \in S_k$, if $s$ is non-zero, $v(H(s)) > v(s)$.

We denote $v(H(s)) - v(s)$ by $\eta_H(s)$, and we say that $H$ is online if for all $s \in S_k$:

- $\eta_H(\Pi_1(s)) \geq i$ for all $i \leq \eta_H(s)$
- $H(s) \cdot s \sim H(\Pi_1(s)) \cdot s$ for all $i \geq \eta_H(s) + 1$.

**Lemma 6.** The function $Q$ introduced in Lemma 5 is an online Euclidean function.

**Proof.** The first point was proved in Lemma 5. Now, let $s = (P,Q)$ be a non-zero element of $S_k$. By construction, we have $v(Q(s)) = v(s) + \eta_H(s)$. In particular, $\eta_H(s) = v(Q \text{ rem}_s N(P)) - v(P) \geq v(Q) - v(P) > 0$. Thus, $Q$ is a Euclidean function.

To prove that $Q$ is online, notice that $Q(s \text{ rem}_s x^{v(s)+1}) \text{ rem}_s x^t = Q(s) \text{ rem}_s x^t$. In particular for $i \leq \eta_H(s)$, $Q(\Pi_1(s)) \cdot \Pi_1(s)$ is given by

$$Q(s) \text{ rem}_s x^{v(s)+1} \text{ rem}_s x^t = Q(s) \text{ rem}_s x^t \mod x^{v(s)+i} = Q(s) \cdot s \mod x^{v(s)+i} = (0,0) \mod x^{v(s)+i},$$

so that $\eta_H(\Pi_1(s)) \geq i$.

If $i > \eta_H(s)$, we prove that there exists a matrix $R$ such that $Q(\Pi_1(s)) = (\text{Id} + zR) \cdot Q(s)$. Indeed, if holds, $(\text{Id} + zR)$ is an invertible matrix over $R_k$, so that $Q(\Pi_1(s)) = (\text{Id} + zR) \cdot Q(s)$. Moreover $\Pi_1(\Pi(\Pi_1(s)) \cdot s) = \Pi_1(Q(s)) \cdot s$, which will be sufficient to conclude. Let $M = G_{D,s,d} \cdot D_s$. If $i > \eta_H(s)$, we note by construction that $M \text{ rem}_s x^t = M$.

Moreover there exists a matrix $R$ such that $N_{G_{D,s},D_s} = (\text{Id} + zR) \cdot N_{G_{D,s},D_s}$, so that $Q(\Pi_1(s)) = (\text{Id} + zR) \cdot Q(s)$.}

Let $s$ be an element of $S_k$ and define by recursion the following sequence of elements of $M_2(\mathbb{R}_k)$:

$$
\begin{align*}
Q_0 &= \text{Id} \\
Q_{n+1} &= Q(Q_n \cdot s) \cdot Q_n.
\end{align*}
$$

For $s \in S_k$, the sequence $(v(Q(s) \cdot s))_{n \in \mathbb{N}}$ is increasing, until it reaches $v(Q(s) \cdot s) = k$. Thus given an integer $n$, we can define the function $Q_n$ from $S_k$ to $M_2(\mathbb{R}_k)$ by

$$Q_n : S_k \rightarrow M_2(\mathbb{R}_k)$$

$$s \mapsto Q_{n_0}, \text{ where } n_0 = \max\{i \in \mathbb{N} | v(Q(s) \cdot s) - v(s) \leq n\}$$

for $s$ non-zero; for $s = (0,0)$, we set $Q(\text{Id})$. In particular, for any $s$, $Q(s) = \text{Id}$ (since for $s$ non-zero, $Q(s) = s$). Now, let $j$ be the minimal integer such that $I_j(s) = I_0(s)$; then, $(Q_n(s) \cdot s) = (s) \cap (x^{v(s)+1})^i$.

**Lemma 7.** For $n \geq 0$, if $s \sim s'$ in $S_k$ then $Q_n(s) \cdot s \sim Q_n(s') \cdot s'$. Moreover, let $j$ be the minimal integer such that $I_j(s) = I_0(s)$; then, $Q_n(s) \cdot s \sim I_j(s)$.

**Proof.** We use a recurrence on $n$. For $n = 0$, it is clear that $Q_0$ satisfies the desired properties. Then, given $n \geq 1$ and $s \sim s'$ two equivalent elements of $S_k$ we know by recurrence assumption that $I_{n+1}(s) \sim I_{n+1}(s')$.

If $v(Q(t) \cdot t) > v(s) + n$, then $Q(t) \cdot t \sim v(s) + n$. Thus $Q_n(s) = Q(t) \cdot I_{n+1}(s)$ and $Q(s) = Q(t) \cdot I_{n+1}(s')$, since $Q_n(s) \cdot s \sim I_{n+1}(s) \cdot s'$. Moreover, in this case, let $j$ be the minimal integer such that $I_j(s) = I_0(s)$. For all $k < \eta_H(t)$ we have

$$I_k(t) = I_0(t).$$

In particular, $k = n - v(t) - v(s) < \eta_H(t)$, and $I_k(t) = I_0(s)$ and $I_0(t) = I_0(s)$. Hence, $(Q_n(s) \cdot s) = (s) \cap (x^{v(s)+1})$ and $j$ is the smallest integer such that $I_j(s) = I_0(s)$.

Otherwise, $v(s) + n - 1 < v(Q(t) \cdot t) \leq v(s) + n$, so that $v(Q(t) \cdot t - v(s) + n$. Then $Q(t) \cdot t - v(s) + n$. Thus $Q_n(s) = Q(t) \cdot I_{n+1}(s)$ and $Q_n(s) = Q(t) \cdot I_{n+1}(s')$. Since $Q$ is a Euclidean function, $Q(t) \cdot t - Q(t') \cdot t'$ and this leads to $Q_n(s) \cdot s \sim Q_n(s') \cdot s'$. Moreover, let $\ell$ be the smallest integer such that $I_\ell(t) \neq I_0(t)$. We have $(Q(t) \cdot t) = (t) \cap (x^{v(s)+1})$. According to Lemma 4 we have $\ell = v(t) = v(s) + n - v(t)$. In particular $n = v(t) + \ell - v(s)$, and $I_\ell(t) = I_0(t)$ and for all $i < n$ we have $I_i(t) \neq I_0(t)$. Thus, $n$ is the smallest integer $j$ such that $I_j(s) = I_0(s)$ and $(Q_n(s) \cdot s) = (Q(t) \cdot t) \cap (x^{v(s)+1})$. Since $(t) = (Q_n(s) \cdot s) = (s) \cap (x^{v(s)+1})$ for some $i \leq n - 1$, this leads to $(Q_n(s) \cdot s) = (s) \cap (x^{v(s)+n})$.

Our goal is to compute efficiently a mapping equivalent to $Q_n$. To this effect, we introduce in Algorithm 1 a generalized version of the half-gcd algorithm [13, p. 320]. In order to control the degree in $y$ of the intermediate elements in Algorithm 1, we use a function $\varphi_{\nu,n} : M_2(\mathbb{R}_k) \rightarrow M_2(\mathbb{R}_k)$, that depends on an element $s \in S_k$ and an integer $n$.

Let $s = (P,Q)$ be a pair of polynomials in $S_k$, $n$ be an integer, and let $M = \left(\begin{smallmatrix} P \nu \\ Q \end{smallmatrix}\right)$ be a matrix of $M_2(\mathbb{R}_k)$. If $Q = 0$ or $v(M \cdot s) = v(s)$ then we let $\varphi_{\nu,n}(M) = \text{Id}$. Otherwise, neither $P$ nor $Q$ are zero and we define $\varphi_{\nu,n}(M)$ as the remainder of

$$
\begin{align*}
L_Q(L_P A \text{ rem}_s N_\nu(Q)) &\ L_P(L_Q B \text{ rem}_s N_\nu(P)) \\
L_Q(L_P C \text{ rem}_s N_\nu(Q)) &\ L_P(L_Q D \text{ rem}_s N_\nu(P))
\end{align*}
$$

by $x^{n+1}$, where $L_P$, $L_Q$ are $L(P), L(Q)$ and $N_\nu(P), N_\nu(Q)$ are two polynomials with constant leading coefficients such that $N(P) = x^{v(P)}N(P)$ and $N(Q) = x^{v(Q)}N(Q)$. The
Algorithm 1 Generalized version of the half-gcd

\textbf{function} GenericHalfGcd(n, s)

\hspace{1em} \textbf{if} n = 0 or \( s = (0, 0) \) \textbf{then}

\hspace{2em} \textbf{return} 1d

\hspace{1em} \textbf{end if}

\hspace{1em} R \leftarrow \text{GenericHalfGcd}\left(\lfloor \frac{n}{2}\rfloor, s\right)

\hspace{1em} n' \leftarrow n - (v(R)) \text{ mod } (\lfloor \frac{n}{2}\rfloor)

\hspace{1em} \tilde{s} \leftarrow \Pi_{n'+1}(R, s)

\hspace{1em} \eta \leftarrow \min(\eta(s), n'+1)

\hspace{1em} \textbf{if} \eta > n' \textbf{then}

\hspace{2em} \textbf{return} R

\hspace{1em} \textbf{end if}

\hspace{1em} \tilde{t} \leftarrow \Pi_{n'+\eta}(Q(\tilde{u}), \tilde{u})

\hspace{1em} S \leftarrow \text{GenericHalfGcd}(n - (v(\tilde{t}) - v(\tilde{u})), \tilde{t})

\hspace{1em} M \leftarrow \varphi_{s,\omega}(S(\tilde{u}) - v(\tilde{u}))(S \cdot Q(\tilde{u}) \cdot R)

\hspace{1em} \textbf{return} M

\textbf{end function}

For any \( s, s' \), if \( \Pi_{n+1}(s) = \Pi_{n+1}(s') \), then \( \mathcal{H}(n, s) = \mathcal{H}(n, s') \), since then \( \tilde{s} = \tilde{s}' \). In particular, with the notation of the algorithm, \( \Pi_{n+1}(s) = \Pi_{n+1}(s') \), which implies that \( \mathcal{H}(n, s) = \mathcal{H}(n, s') \). Hence, by recurrence assumption, we get that \( R, s \sim R(n, s) \sim Q(n, s) \cdot s \).

The definition of \( Q(n, s) \) implies that \( v(Q(n, s)) < v(s) \leq [n/2] \), and by the claim above, we get that \( v(R) < v(s) \leq [n/2] \). This implies that \( v(R) < v(R') \), and, since \( n' = n - (v(R) - v(\tilde{s})) \), \( \Pi_{n'+1}(R) \sim \Pi_{n'+1}(R') \sim \tilde{u} \).

To continue, we distinguish two cases. If \( s \geq \eta(R) + 1 \), then the first part of \( Q \) being online shows that \( \eta(\Pi_{n'+1}(R) \sim s \sim Q(n, s) \cdot \eta \).

In this case, the algorithm returns \( R \). On the other hand, we will prove that in this case, \( Q(n, s) = Q(n, s) \); one this is established, this implies that \( Q(n, s) \sim R, s \sim R, s \) holds in this case. Indeed, \( \eta(R) \sim \eta(Q(n, s) \cdot s) \), which is equivalent to the statement above, and of Lemma 4), which can be rewritten as \( v(Q(n, s)) \sim Q(n, s) \cdot s \sim v(Q(n, s)) \cdot s \).

On the other hand, \( n' = n - (v(R) - v(\tilde{s})) \) gives \( n' \geq n - (v(Q(n, s))) + 1 \), and \( v(R) \sim Q(n, s) \cdot s \) holds in this case, which proves our claim.

Using \( R \sim Q(n, s) \cdot s \), \( Q \) being Euclidean, we get \( Q(n, s) \cdot R \sim Q(n, s) \cdot Q(n, s) \cdot s \cdot Q(n, s) \cdot s \). We claim that the right-hand side has valuation \( v(\tilde{t}) = v(Q(\tilde{u})) \). Indeed, the proof of Lemma 6 establishes the existence of a matrix \( K \) such that \( Q(\tilde{u}) = (\text{Id} + K)Q(\tilde{u}) \cdot R \); this implies that \( v(Q(\tilde{u})) - v(\tilde{u}) \sim Q(\tilde{u}) \cdot s \). On the other hand, the inequality \( n' + 1 > \eta(R) \) also implies that \( v(Q(\tilde{u})) \cdot s \sim v(Q(n, s) \cdot s) \sim v(Q(n, s) \cdot s) \).

The last element in the sequence \( Q(n, s) \cdot s \) from (1) having valuation at most \( v(\tilde{t}) \). On the other hand, the previous paragraph proves that \( Q(n, s) \cdot s \sim Q(n, s) \cdot s \), which is included in the sequence \( Q(n, s) \cdot s \) from (1) having valuation at most \( v(\tilde{t}) \). This is enough to conclude, since the valuations \( Q(n, s) \cdot s \) increase.

Theorem 8: If \( n \equiv 0 \mod 2 \), then \( H(n, s) \sim Q(n, s) \cdot s \).

Moreover, \( \tilde{t} = \Pi_{n+1}(Q(\tilde{u})) \sim \Pi_{n+1}(Q(\tilde{u}) \cdot R, s) \).
and by construction $H(n_1, t) = H(n_1, Q(\bar{u}) \cdot R \cdot s)$, so that

$$S \cdot Q(\bar{u}) \cdot R \cdot s \sim Q_{n_1}(Q(\bar{u}) \cdot R \cdot s) \cdot Q(\bar{u}) \cdot R \cdot s \sim Q_{n_1}(Q_{n-n_1}(s)) \cdot Q_{n-n_1}(s) \cdot s.$$ 

The definition of the sequence $(Q_n)$ implies that the latter expression is equivalent to $Q_n(s) \cdot s$.

Finally, since the function $\varphi_{s,n}$ satisfies $\varphi_{s,n}(S \cdot Q(\bar{u}) \cdot R \cdot s) \sim S \cdot Q(\bar{u}) \cdot R \cdot s$, we conclude that $H(n, s) \sim Q_n$. □

**Lemma 10.** Let $s = (P, Q)$ be in $\mathcal{S}_k$, of degrees at most $d$. For $n > 0$, Algorithm 1 computes $H(n, s)$ in time $O(M(d)\log(n) + M(d)n\log(d))$.

**Proof.** For $n = 0$, for any $s$, computing $H(0, s)$ takes constant time. For a higher value of $n$, remark that the recursive calls are made with arguments $n_0, n_1$ that are at most $\lceil \frac{n}{2} \rceil$: this is clear for $n_0$: for $n_1 = n - (\nu, n) - (\nu, s)$, this is because $v(t) - v(s) = v(Q(\bar{u}) \cdot u) - v(s)$ must be greater than $\lfloor n/2 \rfloor$, by definition of $Q_{n_0}$.

The matrix $R$ has entries of degree at most $d$ (because $\bar{s}$ does), so computing $\bar{u}$ takes time $O(M(d))$, and its entries have degree $O(d)$. More precisely, we saw in the proof of the previous lemma that $R \cdot s \sim Q_{n_0}(s) \cdot s$. Because $Q_{n_0}$ is obtained by iterating $Q$, the last item in Lemma 5 shows that the first entry of $Q_{n_0}(s) \cdot s$ has degree less than $d$; as pointed out before, this implies the same property for $R \cdot s$, and thus for $\bar{u}$ (which is a truncation of it).

Computing $\eta$ takes time $O(M(d))$ by Lemma 3, and the same holds for $\bar{I}$ by Lemma 5, up to an extra term $O(M(d)\log(d))$. In addition, that lemma shows that the entries of $\bar{I}$ have degree less than that of the first entry of $\bar{u}$, and thus less than $d$.

After the last recursive call, it remains to compute $M$. We first compute $Q_1 = Q(\bar{u}) \mod (N_0(P), x^d)$ and $Q_2 = Q(\bar{u}) \mod (N_0(Q), x^d)$, where $d = \nu(s) - \nu(\bar{u}) \leq n$. This can be done in time $O(M(d))$: $Q(\bar{u})$ is a product of three matrices, called $G, D, N$ in Subsection 3.1. The first two have polynomial entries of degree at most $d$, and can be computed in time $O(M(d))$; the last one involves a denominator of the form $\text{Le}(U)^{-1}$, for some polynomial $U$ of degree at most $d$. The direct use of $\text{Le}(U)$ may have degree $O(dk)$, but one can directly compute it modulo $N_0(P)$ or $N_0(Q)$ using Newton iteration on $x$ in time $O(M(d))$.

Then we compute $M_1 = \varphi_{s,n}(S \cdot Q_1 \cdot R)$ and $M_2 = \varphi_{s,n}(S \cdot Q_2 \cdot R)$ and we let $M$ be the concatenation of the first column of $M_0$ and the second column of $M_1$. Thus $M$ can be computed in time $O(M(d))$.

Overall, the time spent on input $(n, s)$ is $O(M(d) + M(d)\log(d))$, plus two recursive calls with parameter at most $\lceil n/2 \rceil$, in degree at most $d$. The total is thus $O(M(d)\log(n) + M(d)n\log(d))$. □

**3.3 Computing pseudo-inverses**

Given two polynomials $P$ and $Q$ in $\mathcal{R}_k$, we will use Algorithm 1 to compute $U$ and $V$ such that $UP + VQ = x^t \mod x^{t+1}$, where $t$ is the smallest integer such that $\langle P, Q \rangle \cap \langle x^t \rangle = \langle x^t \rangle$. In particular, the valuation of the resultant of $P$ and $Q$ is greater than or equal to $t$.

**Corollary 1.** Assume that $P, Q \in \mathcal{R}_k$ have degree at most $d$, and let $t$ be the minimal integer such that $\langle P, Q \rangle \cap \langle x^t \rangle = \langle x^t \rangle$. It is possible to compute in time $O(M(d)\log(n) + M(d)n\log(d))$ two polynomials $U, V$ in $\mathcal{R}_k$ such that $UP + VQ = x^t \mod x^{t+1}$, with $\deg_u(U) < \deg_u(Q)$ and $\deg_v(V) < \deg_v(P)$.

**Proof.** Without loss of generality, assume that $v(P) < v(Q)$. Suppose that we actually have $v(P) < v(Q)$, and define $s = (P, Q) \in \mathcal{S}_k$. In the following we let $t' = t - v(s)$. In particular we have $I_1(s) = \{1\}$ and since for all $i < t'$, $I_1(s) \subseteq I_2(s)$, the properties of $Q_0$, given in Lemma 7 ensure that $Q_0(s) \cdot s = (s) \cap (x^{t'})$. Moreover, for any integer $i \geq t'$, we have $Q_i(s) = Q_{i}(s)$. Thus, by the equivalence property of $\mathcal{H}$ in Lemma 9, we have for any $i \geq t'$:

$$H(i, s) = \left( x^{(a + 2)W} \right),$$

where $s = (P, Q)$ and $W, H$ are in $\mathcal{R}_k$ and $a$ is a non-zero constant. Thus it is enough to compute $\mathcal{H}(i, s)$ for any $i \geq t'$ to recover $U$ and $V$ from the first row of the matrix $\mathcal{H}(i, s)$. On the other hand for any $i < t'$, the first coordinate of $\Pi_1(\mathcal{H}(i, s) \cdot s)$ has a degree greater or equal to 1. Thus we can apply Algorithm 1 to the input $(2^i, s)$, for $i$ from 1 to $\lceil \log k \rceil$ until the first polynomial of $\Pi_1(\mathcal{H}(2^i, s) \cdot s)$ has degree 0 in $y$. We will find $2^a \geq t'$ while calling $\mathcal{H}$ at most $\lceil \log t' \rceil$ times, and this will allow us to conclude.

Suppose now that $v(P) = v(Q)$, let $s = (P, Q)$ and $s' = G \cdot s$, where $G$ is the gcd matrix defined in Subsection 3.1. Then $s'$ satisfies the assumptions of the previous paragraph, we can compute polynomials $U', V'$ such that $U'P + V'Q = x^t \mod x^{t+1}$, with $s' = (P', Q')$. Remark that the value of $t$ is indeed the minimal possible one for $s$ as well, since $G$ is a unit; note also that the degrees of $P', Q'$ are $O(d)$, and thus so are those of $U'$ and $V'$. Multiplying $U', V'$ by $G$, we obtain polynomials $U'', V''$ of degree $O(d)$ such that $U''P + V''Q = x^t \mod x^{t+1}$. Then, define $U = LQ(LP'U'' \mod N_0(Q))$ and $V = LP(LQV'' \mod N_0(P))$, with $LQ, LQ_N, N_0, \Pi_1$ defined as in Lemma 8. These polynomials have prescribed degrees, can be computed modulo $x^{t+1}$ in time $O(M(d))$, and the same proof as in Lemma 8 shows that $UP + VQ = x^t \mod x^{t+1}$. □

**4. THE FIRST NON-ZERO COEFFICIENT**

Before computing the first $k$ coefficients of the resultant $R(x) \in \mathcal{K}[x]/(x^s)$ of two polynomials $P, Q \in \mathcal{R}_k$, we focus on computing the first non-zero coefficient of $R$. The following lemma will allow us to compute it by recurrence.

**Lemma 11.** Let $M, N \in \mathcal{R}_k$ and $U = \mathcal{U}(\mathcal{R}_k)$ and an integer $t \leq k$ be such that $MP + NQ = x^U$ with $\deg_u(N) < \deg_u(P)$, and such that $P$ and $N$ are normal and $\Pi_1(U) = 1$.

Furthermore, assume that $v(u) = v(N) = 0$.

Denote by $d_p, d_{Q}, d_{M}, d_{N}$ the degrees in $y$ of $P, Q, M, N$ respectively, by $p_0 \in \mathcal{K}$ the coefficient of $y^{d_p}$ in $N$. Then, there exists $V$ in $\mathcal{K}[x]/(x^s)$, with $\Pi_1(V) = 1$, such that:

$$\text{Res}(P, Q) = x^{d_{P}d_{Q}d_{N}d_{M}}(-1)^{d_{N}d_{P}}p_0^{d_{N}d_{M}d_{Q}}V \text{ Res}(N, M).$$


Replacing in the first equality $NQ + MP = x^U$, we get $\text{Res}(P, NQ) = p_0^{d_{N}d_{Q}d_{P}}\text{ Res}(P, x^U)$ (see for example [16, 11]). Finally, $\text{Res}(P, x^U) = x^{d_{P}}\text{ Res}(P, U)$. Then since $P$ is normal of valuation zero and $\Pi_1(U) = 1$, there exists $W = 1 + xW \in \mathcal{K}[x]/(x^s)$ such that $\text{Res}(P, U) = p_0^{d_{U}}W$. 


Applying these arguments to $\text{Res}(N, MP)$, we conclude:

$$\text{Res}(P, N) \text{Res}(P, Q) = p_0^{d_P + d_Q} x^{d_P d_Q} W$$

$$\text{Res}(N, P) \text{Res}(N, M) = p_0^{d_N + d_P + d_Q + d_M} W'$$

Note that the symmetry formula for the resultant implies that $\text{Res}(P, N) = (−1)^{d_P d_N} \text{Res}(N, P)$. Then dividing the two equalities, we recover the desired result. ☐

**Lemma 12 (First non-zero coefficient).** Let $d$ be a bound on the degrees in $y$ of $P$ and $Q$. One can determine whether $R = \text{Res}(P, Q)$ vanishes in $K[x]/(x^k)$, and if not compute its first non-zero coefficient and its valuation in $O(M(dk)\log(k) + M(dk)\log(d))$ arithmetic operations.

**Proof.** Lemma 2 allows us to reduce to the case where $P$ and $Q$ are normal polynomials; the cost of this reduction is $O(M(dk))$. Starting from normal $P$ and $Q$, we prove the result by induction; the proof actually only uses the fact that one polynomial, say $P$, is normal. We use an integer argument $\tau$, which gives us a (strict) upper bound on the valuation of the resultant; initially, it is set to $k$.

Dividing by powers of $x$, we can assume that $v(P) = v(Q) = 0$; the upper bound $\tau$ remains valid.

Then we can compute the resultant of $\Pi_1(P)$ and $\Pi_1(Q)$ in $K[y]$, in time $O(M(dk))$. If it is non-zero, we are done.

Else, let $t$ be the smallest integer such that $(P, Q) \cap (x^t) = (x^t)$; hence, $t \leq \tau$, but we also have $t > 0$. Define function $F(d, n) = M(dn)\log(n) + M(d)\log(d)$. Using Corollary 1, we see that there exists a universal constant $c_1$ such that we can compute in $c_1 F(d, t)$ operations two polynomials $U$ and $V$ in $K_{df}$ of degree less than $d$ such that $UP + VQ = x^tW$ with $W \in U \langle R_d \rangle$, with more precisely $\deg(U) < \deg(P)$. Since $t$ is minimal, this implies $v(U, V) = 0$ and since $v(P) = 0$, this implies that $v(V) = 0$.

If $t = \tau$, we are done. Else, let $A = \Pi_{\tau+1}(LC(V))$, $N = \Pi_{\tau+1}(N(V))$ and $M = \Pi_{\tau+1}(U/A \rem_n N(Q))$. Since $A$ is a unit, these definitions imply the equality $M + NQ = x^t(1 + xY) + ZN_a(Q)P$, for some polynomials $Y$ and $Z$. The degree of the left-hand side is less than $\deg(P) + \deg(Q)$, whereas $N(Q)P$ is monic of degree $\deg(P) + \deg(Q)$. Hence, the previous equality shows that $ZN_a(Q)P$ vanishes modulo $x^{t+1}$. The assumptions of Lemma 11 are satisfied; we can thus do a recursive call on $N$ and $M$, with upper bound $\tau−t$, from which we can recover our output using the formula in that lemma.

In terms of complexity, all calculations giving $A, N, M$ can be done in $c_2M(dk)$ operations (the only non-trivial point is the computation of $1/A \rem_n N(Q)$, which is done by Newton iteration on $x$). Hence, the runtime $G(d, \tau)$ satisfies $G(d, \tau) \leq c_0M(d)\log(d) + c_1F(d, t) + c_2M(d) + G(d, \tau−t)$. Using the super-linearity of $F$ in $t$ and of $M$, and the definition of $F$, we deduce the overall cost $G(d, k) = O(F(d, k))$. ☐

5. A DIFFERENTIAL EQUATION

Let $P$ and $Q$ be in $K[x, y]$, and let $R \in K[x]$ be their resultant with respect to $y$. We now prove our main result:

**Theorem 1.** If $P$ and $Q$ have degree at most $d$ and $K$ has characteristic zero, or at least $k$, one can compute $R \rem_x x^k$ using $O(M(dk)\log(k) + M(dk)\log(d))$ operations in $K$.

**First reduction:** If the degree in $x$ of $P$ or $Q$ is greater than or equal to $k$, let $P_k = P \rem_x x^k$ and $Q_k \rem_x x^k$ and let $d_P, d_Q$ be the degrees in $y$ of $P$ and $Q$ respectively. Then,

$$\text{Res}_{d_P, d_Q}(P, Q_k) = \text{Res}(P, Q) \mod x^k,$$

where $\text{Res}_{d_P, d_Q}$ denotes the determinant of the Sylvester matrix associated to the degrees $(d_P, d_Q)$. If both leading coefficients of $P$ and $Q$ have a valuation less than $k$, then $\text{Res}_{d_P, d_Q}(P_k, Q_k) = \text{Res}(P_k, Q_k)$. If both leading coefficients have a valuation greater or equal to $k$, then we have $\text{Res}_{d_P, d_Q}(P_k, Q_k) = p_0^{d_P − \deg_y(Q_k)} \text{Res}(P_k, Q_k)$, where $p_0$ is the leading coefficient of $P$ in $y$. Thus, in any case, we can recover the resultant of $P$ and $Q$ modulo $x^k$ from that of $P_k$ and $Q_k$, in a time that fits in our runtime bound.

**Second reduction:** Assume that $P$ and $Q$ have degree at most $d$ in $y$, with coefficients of degree less than $k$. Using Lemma 2, in time $O(M(dk))$, we can reduce the problem of computing $\text{Res}(P, Q)$ rem $x^k$ to a similar problem with $P$ and $Q$ both monic in $y$, reduced modulo $x^k$, and with no degree increase in $y$ (that lemma proves the existence of suitable polynomials in $R_k$, so we take their canonical lifts to $K[x, y]$). Hence, below, we suppose we are in this case.

With the results of the previous section, we can test if $R = \text{Res}(P, Q)$ vanishes modulo $x^k$, and if not, find its valuation $\mu \leq k$ and the coefficient $c$ of $x^\mu$, in time $O(M(dk)\log(k) + M(dk)\log(d))$. We thus assume that $R \rem_x x^k$ is non-zero, as otherwise we are done. The key to our algorithm is the following differential equation satisfied by $R$ over $K(x)[y]$; below, we write $dy = \deg_y(P)$ and $dz = \deg_z(Q)$.

**Lemma 13.** The following equality holds:

$$\frac{dR}{dx} = R \left( \begin{array}{cc} 1 & \frac{dP}{P} \frac{dQ}{Q} \rem_y Q, y^{d_Q-1} \\ \frac{dQ}{Q} & \frac{dP}{P} \rem_y P, y^{d_P-1} \end{array} \right).$$

**Proof.** Let $A$ be the Sylvester matrix of $P$ and $Q$ with respect to the variable $y$, so that $R = \det(A)$. Since $\rem_x x^k$ is non-zero, $A$ is a unit over $K(x)$.

Differentiating this equality with respect to $x$, we obtain $\frac{d^2R}{dx^2} = R \text{trace}(A^{-1} \frac{dA}{dx})$. Since $A$ is the matrix of the mapping $(F, G) \mapsto FP + GQ$ (in the canonical monomial bases), $\frac{dA}{dx}$ represents $(F, G) \mapsto F \frac{dP}{P} + G \frac{dQ}{Q}$ in the same bases. Similarly, the inverse $A^{-1}$ represents the mapping $S \mapsto (S/P \rem_y Q, S/Q \rem_y P)$. Due to the block structure of the matrix $A^{-1} \frac{dA}{dx}$ its trace is the trace of the block-diagonal operator

$$(F, G) \mapsto \left( F \frac{1}{P} \frac{dP}{P} \rem_y Q, G \frac{dQ}{Q} \rem_y P \right).$$

This mapping being block-diagonal, its trace is the sum of the traces of its two components, $F \mapsto F \frac{dP}{P} \rem_y Q$ and $G \mapsto G \frac{dQ}{Q} \rem_y P$. To conclude, remark that these traces are respectively $\text{coeff}(\frac{dP}{P}, dx, dy, Q, y^{d_Q-1})$ and $\text{coeff}(\frac{dQ}{Q}, dx, dy, P, y^{d_P-1})$. ☐

Using Corollary 1, we can compute in time $O(M(dk)\log(k) + M(dk)\log(d))$ we know two cofactors $U$ and $V$ in $K[x, y]$ of degree less than $k$ in $x$ such that $\deg(U, y) < d_Q$, $\deg(V, y) < d_P$ and...
\[ UP + VQ = x^t \mod x^{k+1}, \text{ with } t \text{ chosen minimal, so that } t \leq \mu. \text{ Then, } t \leq k, \text{ since otherwise } R = 0 \mod x^k. \]

From this, we deduce \( U'V + VQ = x^t \mod x^{2t+k-1} \):

compute \( W \) such that \( UP + VQ = x^t(1 + xW) \mod x^{2t+k-1} \), then the inverses \( J \) (resp. \( B \)) of \( 1 + xW \) modulo \( Q, x^{2t+k-1} \), (resp. modulo \( P, x^{2t+k-1} \)), and let \( U' = U \rem P, Q \) and \( V' = VB \rem P \). The only non-trivial point is the inversions, which are done by Newton iteration with respect to \( x \); overall, the cost of this step is \( O(M(dk)) \).

The defining equality for \( U' \) and \( V' \) can be rewritten as

\[ U'P + V'Q = x^t(1 + x^{k+1} - S), \]

for some \( S \) in \( \mathbb{K}[x, y] \). Their degree constraints then show that the inverse of \( P \) modulo \( Q \) is \( U'/(x^t(1 + x^{k+1} - S)) \rem P, Q \); similarly, the inverse of \( Q \) modulo \( P \) is \( V'/(x^t(1 + x^{k+1} - S)) \rem P, Q \). This further implies that \( x^{2t+k-1} \) is has the form

\[ R = \left( \begin{array}{c}
\frac{U'}{1 + x^{k+1} - S} \\
\frac{V'}{1 + x^{k+1} - S}
\end{array} \right) \rem P, Q, y^q - 1. \]

Taking this equality modulo \( x^{k+1} \), we obtain a relation of the form \( x^{2t+k-1} = RF \mod x^{k+1} \), with

\[ F = \left( \begin{array}{c}
\frac{dP}{dx} \\
\frac{dQ}{dx}
\end{array} \right) \rem P, Q, y^{q-1} \]

\[ \left( \begin{array}{c}
\frac{dP}{dx} \\
\frac{dQ}{dx}
\end{array} \right) \rem P, Q, y^{q-1} \]

Because \( P \) and \( Q \) are both monic in \( y \), once \( U' \) and \( V' \) are known, we can compute \( F \) using \( O(M((t+k)d)) \) operations in \( \mathbb{K} \), which is \( O(M(kd)) \).

Recall that \( R \) is has the form \( c x^p + \cdots \), for some \( p < k \), so that \( x^{2t+k-1} \) has has the form \( \mu x^{k+1} + \cdots \). Thus, the Laurent series \( \frac{dP}{dx} \) has a value at least \(-1\), so that \( F \) has value at least \(-1\). Dividing by \( x^t \) on both sides, we obtain \( x^t = RF \mod x^k \), with \( F = F/x^t \). From now on, let us assume that the characteristic \( p \) of the base field is at least \( k \), or zero. Then, this relation determines \( R \) modulo \( x^k \) up to a constant factor, and knowing the initial condition \( c \) allows us to deduce \( R \mod x^k \) unambiguously. Given \( F \), this is done by means of [3, Theorem 2], which allows us to compute \( R \mod x^k \) in \( O(M(k)) \) operations in \( \mathbb{K} \). Summing all the costs seen so far concludes the proof of our theorem.

Acknowledgements. The authors sincerely thank the reviewers for their careful reading and suggestions. Moroz is supported by project Singcasi (ANR-13-JS02-0006); Schost is supported by NSERC.

6. REFERENCES