Annals of the University of Craiova, Mathematics and Computer Science Series
Volume 49(1), 2022, Pages 52–61, DOI:
ISSN: 1223-6934

A vectorial approach to generalize the remainder theorem

Marcos A. Hidalgo Rosas and Francesco Laudano

Abstract. We propose a new computational proof for the division algorithm that, using vector algebra, generalizes the remainder theorem to divisions for polynomials of any degree over a generic integral domain. Then, we extend this result to calculate the pseudo-divisions. Later, starting from the previous theorems, we obtain some algorithms that calculate the pseudo-remainder and the pseudo-quotient while avoiding long division. Finally, we provide examples and comparisons indicating that these algorithms are efficient in divisions by sparse polynomials and their divisors, as cyclotomic polynomials.

2010 Mathematics Subject Classification. Primary 13B25; Secondary 13F20.
Key words and phrases. polynomial pseudo-division, pseudo-remainder, algorithm, matlab code.

1. Introduction

Many algorithms that perform the polynomials division give priority to the calculation of the quotient ([2, Algorithm 4.3 p. 122], [3, Algorithm D, p. 421 and Algorithm R, p. 425], [5], [1] and reference therein, [7, Algorithm 9.5 p. 261] and [8]). An exception is the remainder theorem, which allows us determining the remainder for divisors of the type $x - c$ through an evaluation, with considerable savings in calculations [Lemma 2.3 p. 123] [6]. In [4, Theorem 2.7 p. 962] the author shows that the remainder theorem can be extended to divisors whose leading coefficient is a unit. This circumstance allows us calculating the remainder without going through the quotient; furthermore, in many cases, we can also calculate the quotient with the same substitution-evaluation technique.

Let, for example, $f = 2x^{16} + 5x^3 + x^2 - 4x + 1$ and $m = x^{10} - 2$. By substituting 2 for x^{10} in $f(x)$, we obtain

$$r = 2 \cdot 2x^6 + 5x^3 + x^2 - 4x + 1 = 4x^6 + 5x^3 + x^2 - 4x + 1,$$

which, as can be easily checked, is the remainder of f on division by m in $Z[x]$. Moreover, by substituting 3 for x^{10} in $f - r = 2x^{16} - 4x^6$, we obtain the polynomial

$$q = 2 \cdot 3x^6 - 4x^6 = 2x^6,$$

which is the quotient we desire.

The first aim of this work is to provide an adequate formalization to the naive computational approach introduced in [4, Theorem 2.7 p. 962] to calculate the remainder of polynomial divisions, framing it in the context of vector algebra. In this way, we obtain a new constructive proof for the so-called Euclidean division algorithm (Theorem 2.2). The second purpose is to applying the above result to calculate the

Received March 10, 2021. Accepted February 26, 2022.

pseudo-remainder and the pseudo-quotient (Theorem 2.3). Besides, we provide some optimized algorithms for calculating the remainder and the pseudo-remainder (Algorithm 1, Algorithm 2, Algorithm 3) that seems to be efficient in divisions by sparse polynomials and their divisors, as cyclotomic polynomials. Finally, we compare the effectiveness of Algorithm 3 with the classical "*deconv*" algorithm.

2. Main theorem and algorithms

In the following D denotes an integral domain, $Frac(D)$ the field of fractions of D and $Frac(D)[x]$ the commutative ring of all polynomials in an indeterminate x with coefficients in $Frac(D)$. As in the ring of integers, we can define the "*modulo-m congruence*" relation in $Frac(D)[x]$, by posing $a \equiv_m b$ if and only if m divides $a-b$ in $Frac(D)[x]$ (i.e., there exists $q \in Frac(D)[x]$ such that $a-b=mq$). As is well known, the modulo-m congruence is an equivalence relation that preserves the operations of $Frac(D)[x]$.

Let $m=\sum_{j=0}^{h} a_j x^j \in D[x]$. The basic idea to determine the remainder of divisions by m, without long division, consist on translate the Pascal divisibility test to the polynomials of $Frac(D)[x]$. In other words, we would determine the remainders of the power of x modulo m.
For this purpose, we will use the following notations:

$$A := -a_h^{-1}\begin{bmatrix} a_{h-1} & a_{h-2} & a_{h-3} & . & . & . & a_1 & a_0 \end{bmatrix},$$

$$X^{(0)} := \begin{bmatrix} x^{h-1} \\ x^{h-2} \\ . \\ . \\ . \\ x^1 \\ x^0 \end{bmatrix}, \text{ and } X^{(k)} := \begin{bmatrix} A\cdot X^{(k-1)} \\ X_1^{(k-1)} \\ X_2^{(k-1)} \\ . \\ . \\ . \\ X_{h-2}^{(k-1)} \\ X_{h-1}^{(k-1)} \end{bmatrix} \text{ for each positive integer } k.$$

Moreover, if B and C are vectors of $Frac(D)[x]^{(h)}$ we say $B \equiv_m C$ if and only if $B_i \equiv_m C_i$, for any $i \in \{1,2,...,h\}$.

At this point, through successive iterations, we can verify that $x^{h+k} \equiv_m A\cdot X^{(k)}$ and $\deg(A\cdot X^{(k)}) < h$, for each $k \in \mathbb{N}$. Then every $A\cdot X^{(k)}$ is the reminder of x^{h+k} on division by m.

In other words, the following lemma holds.

Lemma 2.1. *Let D be an integral domain and let $Frac(D)$ the field of fractions of D. Let $m=\sum_{j=0}^{h} a_j x^j \in D[x]$, with $h := \deg(m) > 0$. Then, for each $k \in \mathbb{N}$ we have*

- $A\cdot X^{(k)} \in Frac(D)[x]$ *and* $x^{h+k} \equiv_m A\cdot X^{(k)}$,
- $X^{(k+1)} \equiv_m X^{(k)} x$,
- $\deg(A\cdot X^{(k)}) < h$.

Proof. We proceed by induction on k. For $k = 0$ we have $A \cdot X^{(0)} = -a_h^{-1} \sum_{j=0}^{h-1} a_j x^j \in Frac(D)[x]$ and $\deg(A \cdot X^{(0)}) < h$. Moreover $x^h \equiv_m A \cdot X^{(0)}$, being $x^h - A \cdot X^{(0)} = x^h + a_h^{-1} \sum_{j=0}^{h-1} a_j x^j = a_h^{-1} m$ a multiple of m.

Then, $X^{(1)} = \begin{bmatrix} A \cdot X^{(0)} \\ x^{h-1} \\ x^{h-2} \\ . \\ . \\ . \\ x^1 \end{bmatrix} \equiv_m \begin{bmatrix} x^h \\ x^{h-1} \\ x^{h-2} \\ . \\ . \\ . \\ x^1 \end{bmatrix} = X^{(0)} x$. In addition, we remark that each power of x appearing in $X^{(1)}$ has a degree less than h.

At this point we can suppose the thesis holds for any $i \in \mathbb{N}$, $i \leq k$, to prove that it holds also for $k + 1$. From the induction hypothesis we have $x^{h+k+1} = x^{h+k} x \equiv_m A \cdot X^{(k)} x \equiv_m A \cdot X^{(k+1)} \in Frac(D)[x]$.

Moreover, $X^{(k+2)} = \begin{bmatrix} A \cdot X^{(k+1)} \\ X_1^{(k+1)} \\ X_2^{(k+1)} \\ . \\ . \\ . \\ X_{h-1}^{(k+1)} \end{bmatrix} \equiv_m \begin{bmatrix} x^{h+k+1} \\ X_1^{(k)} x \\ X_2^{(k)} x \\ . \\ . \\ . \\ X_{h-1}^{(k)} x \end{bmatrix} \equiv_m \begin{pmatrix} A \cdot X^{(k)} x \\ X_1^{(k)} x \\ X_2^{(k)} x \\ . \\ . \\ . \\ X_{h-1}^{(k)} x \end{pmatrix} = X^{(k+1)} x.$

Finally, we can suppose that the powers of x appearing in $X^{(k)}$ have a degree less than h. Indeed, being

$$A \cdot X^{(k+1)} = -a_h^{-1} \begin{bmatrix} a_{h-1} & a_{h-2} & a_{h-3} & . & . & . & a_1 & a_0 \end{bmatrix} \cdot \begin{bmatrix} A \cdot X^{(k)} \\ X_1^{(k)} \\ X_2^{(k)} \\ . \\ . \\ . \\ X_{h-2}^{(k)} \\ X_{h-1}^{(k)} \end{bmatrix},$$

since $\deg(A \cdot X^{(k)}) < h$, we also have $\deg(A \cdot X^{(k+1)}) < h$. □

Example 2.1. With $m = 2x^2 + 3x + 5$ in $\mathbb{Z}[x]$ we have:
$A = \begin{bmatrix} -\frac{3}{2} & -\frac{5}{2} \end{bmatrix}$, $X^{(0)} = \begin{bmatrix} x \\ 1 \end{bmatrix}$, $X^{(1)} = \begin{bmatrix} A \cdot X^{(0)} \\ X_1^{(0)} \end{bmatrix} = \begin{bmatrix} -\frac{3}{2}x - \frac{5}{2} \\ x \end{bmatrix}$, $X^{(2)} = \begin{bmatrix} A \cdot X^{(1)} \\ X_1^{(1)} \end{bmatrix} = \begin{bmatrix} \frac{9}{4}x + \frac{15}{4} - \frac{5}{2}x \\ -\frac{3}{2}x - \frac{5}{2} \end{bmatrix}$, $X^{(3)} = ...$ Then
$x^2 \equiv_m A \cdot X^{(0)} = -\frac{3}{2}x - \frac{5}{2}$, $x^3 \equiv_m A \cdot X^{(1)} = -\frac{1}{4}x + \frac{15}{4}$, $x^4 \equiv_m A \cdot X^{(2)} = \frac{33}{8}x + \frac{5}{8}$, $x^5 \equiv_m A \cdot X^{(3)} = ...$

Let $f = \sum_{j=0}^{n} f_j x^j \in D[x]$, with $a_n \neq 0$ and $n \geq h$. Thus, by the previous considerations, follows that the reminder of f on division by m is

$$r = \sum_{k=0}^{n-h} f_{h+k}(A \cdot X^{(k)}) + \sum_{j=0}^{h-1} f_j x^j,$$

and it can be calculated starting from $r = \sum_{j=0}^{h-1} f_j x^j$, whit the iteration

$$r := r + f_{h+k} A \cdot X^{(k)}.$$

Moreover, we can prove that q is the remainder of $f - r$ on division by $m - 1$, then, in many cases, it can be calculated applying the previous method to these latter polynomials.

In other words, keeping the previous notations for A and $X^{(k)}$, we can state the following Theorem, which provides a formal computational proof for the Euclidean division algorithm.

Theorem 2.2. *Let D be an integral domain and let $Frac(D)$ the field of fractions D. Let $f, m \in D[x]$, with $f = \sum_{j=0}^{n} f_j x^j$, $m = \sum_{j=0}^{h} a_j x^j$ and $h := \deg(m) > 0$. Then, there exist unique polynomials $q,\ r \in D[x]$ with $f = mq + r$, where either $r = 0$ or $\deg(r) < h$.*
Moreover, if $\deg(f) \geq \deg(m)$, then
(1) $r = \sum_{k=0}^{n-h} f_{h+k}(A \cdot X^{(k)}) + \sum_{j=0}^{h-1} f_j x^j$,
(2) *if $\deg(f) < 2 \cdot \deg(m)$ then* $q = \sum_{k=0}^{n-h} b_{h+k}(A_q \cdot X^{(k)}) + \sum_{j=0}^{h-1} b_j x^j$,
with $A_q = -a_h^{-1} \begin{bmatrix} a_{h-1} & a_{h-2} & a_{h-3} & \dots & a_1 & a_0 - 1 \end{bmatrix}$ *and* $f - r = \sum_{j=0}^{n} b_j x^j$.

Proof. We first prove the existence of q and r. We have $f = 0m + f$; then, if $f = 0$ or $\deg(f) < h$, the existence is proved by posing $r = f$ and $q = 0$.
Let us now analyze the case $n := \deg(f) \geq h$. Since a_h is a unit of R, from the above Lemma 2.1 for each $k \in \mathbb{N}$ we have $x^{h+k} \equiv_m A \cdot X^{(k)}$, moreover $\deg(A \cdot X^{(k)}) < h$. Then the polynomial $r = \sum_{k=0}^{n-h} a_{h+k}(A \cdot X^{(k)}) + \sum_{j=0}^{h-1} a_j x^j$ is m-congruent to f and $\deg(r) < h$, i.e there exist a polynomial $q \in R[x]$ such that $f = mq + r$. Moreover, if there are $q', r' \in R[x]$ such that $f = mq' + r'$ with $\deg(r') < h$, follows $m(q - q') = r' - r$. Then $q = q'$, otherwise we would have $\deg(r - r') = \deg(m(q - q')) \geq \deg(m)$, which is false. Thus r and q are unique.
To prove the second item, we just observe that, being $f - r = (m - 1)q + q$, we have $q \equiv_{m-1} f - r$. From the hypotesis $\deg(f) < 2 \cdot \deg(m)$ we have $\deg(q) = \deg(f) - \deg(m) < 2 \cdot \deg(m) - \deg(m) = \deg(m) = \deg(m - 1)$. □

The previous theorem gives rise to the following algorithm.

2.1. Pseudo-Remainder algorithm. In many cases, we can determine the polynomial remainder up to a multiplicative constant, as happens, for example, in the calculation of a polynomials greatest common divisor. This possibility allows us to avoid the use of divisions and, therefore, to obtain more efficient algorithms. In particular, we can perform a new division for which the remainder and the quotient belong to $D[x]$; precisely, the division of $a_h^{n-h+1} f$ by m [2, pp. 54-55]. The remainder and

Algorithm 1 (General Remainder algorithm)

Input: $f, m \in D[x]$, *with* $f = \sum_{j=0}^{n} f_j x^j$, $m = \sum_{j=0}^{h} a_j x^j$ *and* $n \geq h > 0$.
Output: *remainder* $r \in Frac(D)[\mathrm{x}]$ *of* f *on division by* m.
Do: *For* $k = 0$ *to* $(n-h)$, *calculate the remainder* $A \cdot X^{(k)}$ *of* x^{h+k} *mod* m, $(X^{(k)} := A \cdot X^{(k-1)})$,
calculate the remainder r *of* f *mod* m $(r = \sum_{j=0}^{h-1} f_j x^j$, *if* $f_{h+k} \neq 0$ *then* $r := r + f_{h+k} A \cdot X^{(k)})$.

the quotient of this division are obtained by $n - h + 1$ instances of the substitution method provided in Algorithm 1, applied using recursively the rule

$$a_h x^h \to -\sum_{i=0}^{h-1} a_i x^i.$$

Thus, using the previous notations for A, A_q, $f - r$, and $X^{(k)}$, we have the following result.

Theorem 2.3. *Let* D *be an integral domain; let* $f, m \in D[x]$, *with* $f = \sum_{j=0}^{n} f_j x^j$, $m = \sum_{j=0}^{h} a_j x^j$ *and* $h := \deg(m) > 0$. *Then, there exist unique polynomials* $q,\ r \in D[x]$ *with* $a_h^{n-h+1} f = mq + r$, *where either* $r = 0$ *or* $\deg(r) < h$.
Moreover, if $\deg(f) \geq \deg(m)$, *then*
(1) $r = a_h^{n-h+1}\big(\sum_{k=0}^{n-h} f_{h+k}(A \cdot X^{(k)}) + \sum_{j=0}^{h-1} f_j x^j\big)$,
(2) *if* $\deg(f) < 2 \cdot \deg(m)$ *then* $\quad q = a_h^{n-h+1}\big(\sum_{k=0}^{n-h} b_{h+k}(A_q \cdot X^{(k)}) + \sum_{j=0}^{h-1} b_j x^j\big)$.

As is well known, the above polynomials q and r are called the pseudo-quotient and the pseudo-remainder of f on pseudo-division by m. In the following they are denoted by $prem(f, m)$ and $pquot(f, m)$.

The following algorithm can be used to calculate the pseudo-remainder.

Algorithm 2 (Pseudo-Remainder algorithm)

Input: $f, m \in D[x]$, *with* $f = \sum_{j=0}^{n} f_j x^j$, $m = \sum_{j=0}^{h} a_j x^j$ *and* $n \geq h > 0$.
Output: *Pseudo-remainder* $r \in D[\mathrm{x}]$ *of* f *on division by* m.
Do: *For* $k = 0$ *to* $(n-h)$, *calculate the remainder* $(A \cdot X^{(k)})$ *of* $a_h^{k+1} x^{h+k}$ *mod* m, $(X^{(k)} := A \cdot X^{(k-1)})$,
calculate the pseudo-remainder pr *of* f *mod* m $(pr = \sum_{j=0}^{h-1} f_j x^j$, *if* $f_{h+k} \neq 0$ *then* $pr := pr + a_h^{n-h-k} f_{h+k} X^{(k)})$.

2.2. Remainder to divisor of $c_s x^s - c_0$. As a particular case of Algorithm 1, we can determine the remainder in divisions for divisors of polynomials of type $c_s x^s - c_0$. In the following, to simplify, we will consider only primitive polynomials.
Let $m = \sum_{j=0}^{h} a_j x^j$ be such a divisor, then $c_s x^s - c_0 = qm$ for some $q \in D[x]$, i.e. $x^s \equiv_m \frac{c_0}{c_s}$ in $Frac(D)[x]$. Consequently, posing $\frac{c_0}{c_s} := c$, the remainders of the divisions of the powers of x by $c_s x^s - c_0$ will develop as follows:

$$1,\ x,\ \ldots\ x^{s-1},\ c,\ cx,\ \ldots\ cx^{s-1},\ c^2,\ c^2 x \ldots$$

Therefore, posing $f = \sum_{j=0}^{n} f_j x^j \in D[x]$ with $n := \deg(f) \geq h > 0$, $n = ks + t$ with $0 < t < s$ and $f_{sk+u} = 0$ for any integer u with $t < u < s$, we can write

$$f = f_{sk+(s-1)}x^{sk+(s-1)} + \ldots + f_{sk}x^{sk} + f_{s(k-1)+(s-1)}x^{s(k-1)+(s-1)} + \ldots + f_2x^2 + f_1x + f_0,$$

and from $(x^s)^j \equiv_m c^j$, we have $x^{sj+t} \equiv_m c^j x^t$. Substituting c for x^s in f, and grouping the coefficients of terms having an equal degree, we obtain

$$r_0 = \left(\sum_{j=0}^{k} f_{sj+(s-1)}c^j\right)x^{s-1} + \left(\sum_{j=0}^{k} f_{sj+(s-2)}c^j\right)x^{s-2} + \ldots + \left(\sum_{j=0}^{k} f_{sj+1}c^j\right)x + \left(\sum_{j=0}^{k} f_{sj}c^j\right)$$
$$= \sum_{i=0}^{s-1}\left(\sum_{j=0}^{k} f_{sj+i}c^j\right)x^i.$$

Thus, since $\deg(r_0) < h$, follows that r_0 is the remainder of f modulo $c_s x^s - c_0$. At this point, if r is the remainder of f modulo m, we have $r \equiv_m f \equiv_m r_0$. Then, we can calculate the polynomial r applying Algorithm 1 to r_0. Moreover, if $n \geq s > n/2 > 0$, from Theorem 2.2, we can also calculate the quotient using the same algorithm on the polynomials $r - r_0$ and $m - 1$.

Algorithm 3 (Remainder algorithm for divisor of $c_s x^s - c_0$)

Input: $f, m \in D[x]$, *with* $m = \sum_{j=0}^{h} a_j x^j$ *is a divisor of* $c_s x^s - c_0$, $f = \sum_{j=0}^{n} f_j x^j$ *and* $n = sk + t \geq h > 0$ *with* $0 \leq t < s$.
Output: *Remainder* $r \in Frac(D)[x]$ *of* f *on division by* m.
Do: *Calculate* r_0 ($r_0 = \sum_{i=0}^{s-1} f_i x^i$, *For* $j = 1$ *to* k, $r_0 = r_0 + \sum_{i=0}^{s-1} f_{sj+i}x^i$), *if* $s < h$ *then calculate the remainder of* r_0 *modulo* m *using Algorithm 1.*

3. Implementation, examples and efficiency

In this section, we provide some Matlab implementations of the previous algorithms. We also provide examples and compare the effectiveness of Algorithm 3 with the classical algorithm *deconv*.

Matlab Code for General Remainder algorithm

```
function [r] = GRA(f,m)
    n  = size(f,2)-1;
    h = size(m,2)-1;
    AX=zeros(n-h+1,h);
    for i=1:h
        AX(1,i)=-m(i+1)/m(1);
    end
    if n>h
        for k=2:n-h+1
            for i=1:h-1
                AX(k,i)=AX(k-1,1)*AX(1,i)+AX(k-1,i+1);
            end
            AX(k,h)=AX(k-1,1)*AX(1,h);
```

```
        end
    end
    r=zeros(1,h);
    for k=1:n-h+1
        if f(k)≠ 0
            r(1,:)=f(k)*AX(n-h+2-k,:)+r(1,:);
        end
    end
    for i=0:h-1
        r(1,h-i)=f(n-i+1)+r(1,h-i);
    end
end
```

Example 3.1. Using the previous code we can check that the remainder of

$$f = 6x^4 + 2x^3 - 3x^2 + 5x + 2 \quad \text{modulo} \quad m = 2x^3 - x + 2 \quad \text{in } \mathbb{R}[x] \quad \text{is} \quad r = 0.$$

Matlab Code for pseudo-remainder algorithm

```
function [r] = GPseudoRemainder(f,m)
    n  = size(f,2)-1;
    h = size(m,2)-1;
    AX=zeros(n-h+1,h);
    for i=1:h
        AX(1,i)=-m(i+1)/m(1);
    end
    if n>h
        for k=2:n-h+1
            for i=1:h-1
                AX(k,i)=AX(k-1,1)*AX(1,i)+AX(k-1,i+1);
            end
            AX(k,h)=AX(k-1,1)*AX(1,h);
        end
    end
    r=zeros(1,h);
    for k=1:n-h+1
        if f(k)≠ 0
            r(1,:)=f(k)*AX(n-h+2-k,:)+r(1,:);
        end
    end
    for i=0:h-1
        r(1,h-i)=f(n-i+1)+r(1,h-i);
    end
    r(1,:) = r(1,:)*(m(1,1)^(n-h+1));
end
```

Example 3.2. Using the previous code we can check that the pseudo-remainder of

$$f = 6x^4 + 2x^3 - 3x^2 + 5x + 2 \quad \text{modulo} \quad m = 2x^3 + x - 2 \quad \text{in } \mathbb{R}[x].$$

is $Pr = -24x^2 + 40x + 16$.

Matlab Code for divisors of $c_s x^s - c_0$

```
function [r] = Rem_Binomial_Divisors(f,m,p)
    n = size(f,2)-1;
    s = size(p,2)-1;
    h = size(m,2)-1;
    k = fix(n/s);
    t=rem(n,s);

    c=-p(s+1)/p(1);
    r=zeros(1,s);
    for i= 1:(n+1)
        g(i)=f(i);
    endfor
    for i= 1:(s-t)
        f(i)=0;
    endfor
    for i= (s-t+1):(n+s-t+1)
        f(i)=g(i-s+t);
    endfor

    for i=0:s-1
        for j=0:k
            if f(s*j+s-i+1)≠ 0
                r(s-i)=r(s-i)+f(s*j+s-i+1)*c^(k-j);
            end
        end
    end

    if h<s
      r=GRA(r,m)
    end
end
```

Example 3.3. Using the previous code we can check that the remainder of

$$f = 2x^{13} - 3x^{11} + 4x^8 - 2x^4 + x + 1 \quad \text{modulo} \quad m = x^7 + 2 \quad \text{in } \mathbb{Z}[x]$$

is $r = \sum_{l=0}^{6}(a_l + a_{7+l}(-2))x^l = -4x^6 + 4x^4 - 7x + 1$.

Example 3.4. Using the previous code we can check that the remainder of

$$f = 2x^{13} - 3x^{11} + 4x^8 - 2x^4 + x + 1 \text{ modulo } m = \phi_6 = x^2 - x + 1,$$

the 6th cyclotomic polynomial $\in \mathbb{Z}[x]$, is $r = 12x - 6$.

The results show that the execution times of Algorithm 3 remain low (due to the linear trend) and do not change even when increasing the degree of the divisor. On the contrary, the *deconv* algorithm presents a polynomial trend, its execution times are greater and increase considerably as the degree of the divisor increases (FIGURE 1).

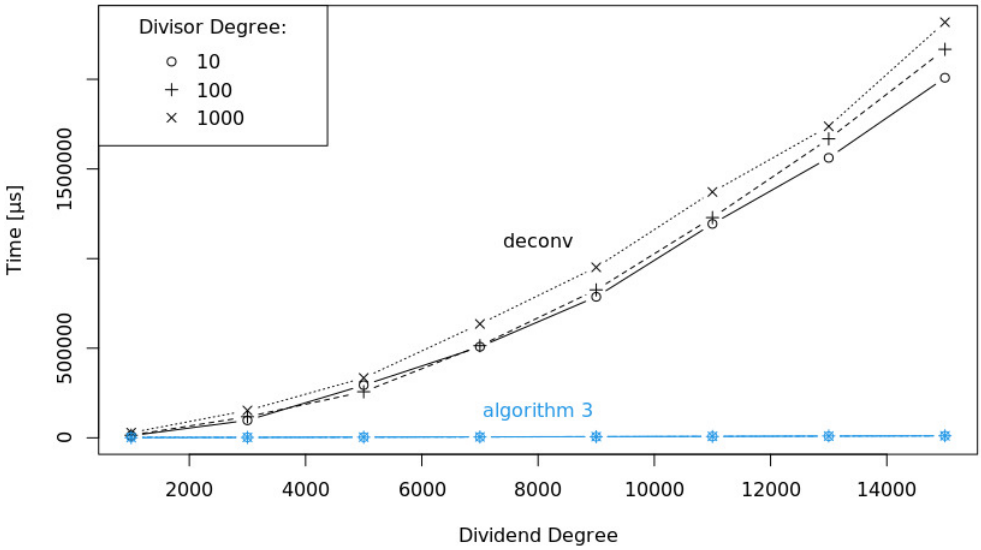

FIGURE 1. Execution time between Algorithm 3 and the classic *deconv* algorithm, for computation of the remainder of sparse polynomials divisions for polynomials of the type $c_s x^s - c_0$. The computer used was equipped with an AMD EPYC 7742 64 Core Processor , 256 cores, and 1.08 TB ram.

4. Conclusion

In this work, using vector algebra, we provided, concurrently, a new computational proof for the Euclidean algorithm and a generalization for the remainder theorem (Theorem 2.2). Then, we proved a theorem that allows calculating the pseudo-remainder and pseudo-quotient in the divisions of polynomials over a generic integral domain (Theorem 2.3). Starting from this theorem, we coded an algorithm that calculates the pseudo-remainder and the pseudo-quotient while avoiding long division (Algorithm 2.1). Finally, we provided examples and comparisons showing that Algorithm 3 seems to be efficient in divisions by polynomials which are divisors of $x^h - a$, such as cyclotomic polynomials.

The results obtained seem to indicate that the framework of vector algebra used in this work could allow us to extend this computational approach to the pseudo-divisions of multivariate polynomials with coefficients on generic rings, possibly even non-commutative ones.

Disclosure Statement

No potential conflict of interest was reported by the authors.

References

[1] D. Bini and V. Pain, Polynomial Division and Its Computational Complexity, *Journal of Complexity* **2** (1986), 179–203.

[2] K.O. Geddes, S.R. Czapor, and G. Labahn, *Algorithms for Computer Algebra (3rd ed)*, Klewer Academic Publisher, 1992.

[3] D.E. Knuth, *The Art of Computer Programming Vol 2 Seminumerical algorithm (3rd ed)*, Addison-Wesley, 1998.

[4] F. Laudano, A generalization of the remainder theorem and factor theorem, *Int J Math Educ Sci Technol.* **50** (2019), no. 6, 960–967. DOI: 10.1080/0020739X.2018.1522676

[5] F. Richman, A division algorithm, *Journal of Algebra and Its Applications* **4** (2005), no. 4, 441–449. DOI: 10.1142/S0219498805001289

[6] J.J. Rotman, *Advanced Modern Algebra*, Prentice-Hall, 2003.

[7] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra (Third edition)*, Cambridge University Press, 2013.

[8] https://it.mathworks.com/help/matlab/ref/deconv.html#bvjpzjj-1

(Marcos A. Hidalgo Rosas) Faculty of Life Sciences, Universidad Regional Amazónica Ikiam, Tena, 150102, Ecuador
E-mail address: marcos.hidalgo@est.ikiam.edu.ec, m.arco.s@outlook.com

(Francesco Laudano) Department of Agricoltura, Università degli Studi del Molise, Via F. De Santis, Campobasso, 86100, Italy. ORCID ID 0000-0003-4489-095X
E-mail address: francesco.laudano@unimol.it, frlaud.fl@gmail.com