Research Square

Preprints are preliminary reports that have not undergone peer review. They should not be considered conclusive, used to inform clinical practice, or referenced by the media as validated information.

Reducing Rational Polynomial: A Proposition of a Strategy to Deal With Floating Points Numbers Using SVD

Ahmad DEEB (ahmaddeeb25@gmail.com)

Université de La Rochelle https://orcid.org/0000-0003-1966-0976

Rafik Belarbi

La Rochelle University: Universite de La Rochelle

Research Article

Keywords: Greatest common divisors, Sylvester's Matrix, Singular Value Decomposition, multiplication, finite field, large integers, floating point operation

Posted Date: May 17th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-494879/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License

Reducing Rational polynomial: A proposition of a strategy to deal with floating points numbers using SVD

Ahmad DEEB[a,*], Rafik BELARBI[a,1]

[a] *University of La Rochelle*
LaSIE UMR CNRS 7356
Avenue Michel Crépeau, 17042 La Rochelle Cedex 01
ORCiD:0000-0003-1966-0976

Abstract

In this article, we present a new strategy to reduce rational polynomial based on the kernel of a linear map defined by the matrix's Sylvester. The strategy does not hold the computation of the Greatest common divisors (GCD) of two polynomials, as other algorithms do, but produce the reduced fraction directly. This strategy was inspired when we consider elements of Padé approximant as a basis is the Proper Generalised Decomposition for solving Partial Differential equation. The algorithm can use the Singular value decomposition technique when dealing with a polynomial with floating-point arithmetic. We compare it with Brown's algorithm in two wedges: multiplication for finite fields and large integers. Results are shown in term of time computation and robustness. The proposed algorithm shows that the time accuracy of computing the reduced fractional is at the same order as the Brown algorithm for finite field and large integers when the GCD of both polynomials has a small degree and an improving when the GCD's degree increase with the degree of polynomials. Also, robustness is more dynamic when arithmetic with the floating-point operation.

Keywords: Greatest common divisors; Sylvester's Matrix; Singular Value Decomposition; multiplication, finite field, large integers, floating point operation

1. Introduction

In numerical calculus addressed for solving problems with mathematics and computers, Proper Generalized Decomposition is an efficient tool to solve inverse problems and identify parameters for partial differential equations. This tool consists of constructing, for each variable in the equation, base elements that will span the tensor space of solutions for each parameter. The type of element could be defined a priori, and therefore this generates a different type of equations. In the case of parametric identification, solutions could be

*Corresponding author
Email addresses: ahmad.deeb@univ-lr.fr (Ahmad DEEB), rafik.belarbi@univ-lr.fr (Rafik BELARBI)

considered to be of exponential's type other function's type as rational functions, and such that variables of these functions are the parameters to be identified. Rational polynomials functions or Padé approximant (relative to Henri Padé's works [1]) are well known for the large spectrum they have to approximate functions. So they are efficient in parametric identification: the constructed abacus can cover a large domain of solution and so identify parameters by optimizing errors with experimental outputs.

Rational polynomials or Padé approximants are widely used in numerical computation. They helps to get better approximation of functions, solutions of equations [2], as for instance integro-differentials equations [3] and others differentials equations [4, 5] obtained in many applications. For example, in mechanics, they perform very well to solve the equation of an an-harmonic oscillator [6]. Other examples of application could be found in strength materials for post-buckling problems [7], in Fluid mechanics to solve heat equation [8] and Blasius-equation which occurs in boundary layers [9]. They are used also in astrophysics and cosmology [10, 11], in Quantum Chromo-Dynamics [12] and other fields [13]. Padé approximants may also help to solve inverse problems. For example, in material science and when we need to model the Paramagnetisim of material governed by a uniform magnetic field, Langevin function was introduced to predict the magnetization. In the inverse case and once we could measure the magnetization and we search to predict the magnetic field, we need to compute the inverse of the Langevin Function. Padé approximants were proposed [14, 15] to solve this issue and get approximations of its inverse.

When dealing with rational polynomials, having the reduced one is essential to prevent over-floating issues in numerical computation. To reduce them, we usually search their Greatest Common divisors (GCD) to simplify it. There are several ways to find the GCD of two polynomials. Two of them are Factorizing and the Euclidean algorithm. The first one finds the factors of each expression, then selects the set of common factors held by all from within each set of factors. This method [16] may be useful only in simple cases, as factoring is usually more complicated than computing the greatest common divisor. Many algorithms in computer algebra are invented as Berlekamp's algorithm [17] and Cantor-Zassenhaus [18] whose both take as input a square-free polynomial and send the factored polynomial.

The Euclidean algorithm, which is the oldest non-trivial algorithm (according to Knuth [19]), consists of constructing a sequence of polynomials to get the Greatest common divisor (GCD). This algorithm uses the Euclidean division for elements in an integral domain. The idea consists of constructing series of polynomials using the remainder sequences of the Euclidean division: For two elements P_1 and P_2 of an integral domain, the euclidean decomposition is carried out ($P_1 = Q_1P_2 + R_1$) and the remainder R_1 is used to establish a new decomposition ($P_2 = Q_2R_1 + R_2$) and so on ($R_{i-1} = Q_{i+1}R_i + R_{i+1}, i = 2...k$) till to find a remainder R_k which is equal to zero or one. If the last remainder is equal to one, both elements are called prime. If not, the R_{k-1} is their GCD. In the domain of polynomials, the complexity of finding the GCD of two univariate polynomials is equivalent to the complexity of multiplying both. An accelerated version is proposed: the semi-GCD algorithm [20]. Many effective algorithms were proposed to compute GCD of two polynomials with coefficients in the integral domain. For more details, we refer the reader to these publications [20, 21, 22, 23, 24, 25]

Nevertheless, Euclid's algorithm presents a difficulty: The coefficients of the remainder sequences increase exponentially. Therefore pseudo-remainder algorithm could be replaced to overcome this problematic [26]. However, this asks for computing GCD of coefficients which requires more other time operations. Furthermore, the establishment of the polynomials remainder sequences is a complex procedure and that why Collins in his work proposes the context of the subresultant[27]: the determinant of the matrix associated to a linear application. In his works, he finds that the remainder of a euclidean division for two polynomials is a polynomial which is equivalent to a pseudo one whose coefficients are determinants of subs-matrix of the Sylvester matrix. To recall, For two polynomials P of order r and Q of order s (with $r > s$), the Sylvester matrix is a $r + s - 1$ square matrix where each row of the first s rows are the coefficients of P and the last r rows are coefficients of Q. If the determinant of this matrix, which is known as Resultant of P and Q (Res(P, Q)), is non-zero, so the GCD of them is 1. If not, The coefficients of the GCD polynomial are in the last non-zero coefficients in the Row echelon form. Brown [24] use these works to simplify the construction of a Polynomial remainder sequence. He also proposes a variant of his algorithm To overcome the problems of increasing coefficient with these sequences by using the leading coefficient of remainders to reduce their coefficients.

Others algorithms which are based on Sylvester's matrix exist to compute the GCD. Mitrouli et al. [28, 29] used this property to proceed on the factorization of the Sylvester matrix and construct a modified QU and LU methods in which it divides by three numbers of floating-point operations needed to obtain the echelon form and then compute GCD. This strategy gives us the GCD but not reduced polynomial. Euclidean division in the $\mathbb{C}[X]$ ring is necessary to get the reduced one which asks for additional floating-point operations.

In this article, we will present an algorithm to compute the reduced rational polynomial based on the property: Fraction has a different representation, and for rational polynomial, we will choose the one with the smallest degree. For a rational polynomial (P, Q), we find the one that verifies the relation

$$\frac{P}{Q} = \frac{P'}{Q'}$$

which is minimal regarding the degree of its polynomials; i.e. we will search two polynomials P' and Q' that verify the equation in which degree of the polynomial equation is minimal:

$$PQ' - P'Q = 0 \tag{1}$$

This equation could be represented in the matrix form when the Singular Value decomposition (SVD), methods which are familiar in numerical computation of reduced orders models, is useful. I present the algorithm in the next section with its strategy and different variants of it. In section 3, numerical tests are conducted for polynomials to show efficiency in time computation, accuracy and robustness for two fields: multiplication in Finite Field and multiplications in arithmetic for large integers.

2. Strategy and algorithm

In this section, we will present the strategy of the proposed algorithm to find reduced rational polynomial. We will set two two variants of the algorihm. The first one is addressed for multiplication in the finite field and large integers and uses Gaussian elimination. The second one is addressed to floating-point operations. Moreover, use Singular Value Decomposition (SVD). SVD methods are used to reduce order models, and here it will be used not just to determine the rank of Sylvester's matrix, which read GCD's degree but to select also, in the orthogonal basis, the reduced element. The main algorithm is detailed, and all process are presented too. We end this section by presenting algorithmic tables of algorithms.

2.1. Strategy

The strategy is based on the idea: For a rational polynomial (P, Q), we search for a rational polynomial (P', Q') that verify the equality:

$$PQ' - P'Q = 0 \tag{2}$$

and with a minimal degree of polynomials. This is represented as finding the kernel of the following application:

$$\phi_{P,Q} : \begin{array}{ccc} K[X] \times K[X] & \mapsto & K[X] \\ (Q', P') & \to & PQ' - P'Q \end{array} \tag{3}$$

While the irreducible one is unique modulo constant, we consider that the denominator Q' is unitary, i.e. we fix the non-zero coefficient, multiplying the first monomial as a unit. Therefore, the strategy to find the degree's lowest one is to start from a fraction with the minimal degree's possibility and test if it verifies the above equality. In a situation where we have a fraction with polynomials having equal degrees, we start with a constant polynomial. In other cases, the difference between degrees of both polynomials are considered; i.e. for polynomials P and Q with degree r and s respectively and such that $|r - s| = K > 0$, we will start with a rational polynomial having K as the degree of the nominator and 0 as the degree of the denominator (if $deg(P) > deg(Q)$).

For a Rational polynomial defined by the couple of vectors

$$(P : (p_0, \ldots, p_r), Q : (q_0, \ldots, q_s))$$

with order $(r = deg(P), s = deg(Q))$, we start to consider that the reduced fraction is composed by two polynomials of order $(r - s, 0)$ [1], i.e. P' and Q' are two polynomials with degree $r - s$ and 0 respectively. We insert the two polynomials in the relation (1) in order to test if they will verify it. In the case they verify it, each monomial of the left hand side must

[1] if $r > s$, if not we choose $(0, s - r)$

have a zero coefficient. Then, a system of $n+1+|r-s|$ equations is obtained ($n = \max(r,s)$, $m = \min(r,s)$).

In the following, we will take the case when $r-s = K > 0$. In the other case, an inversion of the rational polynomial is considered; i.e. in the algorithm, P and Q are mutated.

2.2. First iteration k=0

We consider that the rational polynomial is reduced to a one with $P' = a_0 + \ldots + a_K x^K$ and $Q' = b_0$ [2]. So the relation (1) gives us:

$$(p_0 + p_1 x + \ldots + p_r x^r) b_0 - (q_0 + q_1 x + \ldots + q_s x^s)(a_0 + \ldots + a_K x^K) = 0 \tag{4}$$

Assembling coefficients in each monomial give us the following equation:

$$(p_0 b_0 - q_0 a_0) + (p_1 b_0 - q_1 a_0 - q_0 a_1) x + \ldots + (p_s b_0 - q_s a_0 - \ldots - q_{s-K} a_K) x^s + \ldots = 0 \tag{5}$$

Each monomial gives us an equation of the system. We have $n+1$ equations with $K+2$ unknowns: $b_0, a_0, \ldots, a_K$. Then, the obtained system is given as below:

$$\left\{ \begin{array}{cccccc}
p_0 b_0 & -q_0 a_0 & & & & = 0 \\
p_1 b_0 & -q_1 a_0 & -q_0 a_1 & & & = 0 \\
\vdots & \vdots & \vdots & & & \\
p_K b_0 & -q_K a_0 & -q_{K-1} a_1 & \ldots & -q_0 a_K & = 0 \\
\vdots & \vdots & \vdots & & \vdots & \\
p_s b_0 & -q_s a_0 & -q_{s-1} a_1 & \ldots & -q_{s-K} a_K & = 0 \\
p_{s+1} b_0 & & -q_s a_1 & \ldots & -q_{s-K+1} a_K & = 0 \\
\vdots & & & & & \\
p_{r-1} b_0 & & & -q_s a_{K-1} & -q_{s-1} a_K & = 0 \\
p_r b_0 & & & & -q_s a_K & = 0
\end{array} \right. \tag{6}$$

The associated matrix of the above system is given below:

$$A_0 = \begin{pmatrix}
p_0 & -q_0 & 0 & \ldots & 0 & 0 \\
p_1 & -q_1 & -q_0 & \ldots & 0 & 0 \\
\vdots & \vdots & \vdots & & & \vdots \\
p_K & -q_K & -q_{K-1} & \ldots & -q_1 & -q_0 \\
\vdots & \vdots & \vdots & & & \vdots \\
p_s & -q_s & -q_{s-1} & \ldots & -q_{s-K+1} & -q_{s-K} \\
p_{s+1} & 0 & -q_s & \ldots & -q_{s-K} & -q_{s-K+1} \\
\vdots & \vdots & \vdots & & & \vdots \\
p_{r-1} & 0 & \ldots & & -q_s & -q_{s-1} \\
p_r & 0 & \ldots & & 0 & -q_s
\end{pmatrix}, X = \begin{pmatrix} b_0 \\ a_0 \\ a_1 \\ \vdots \\ a_K \end{pmatrix} \tag{7}$$

[2] when $r < s$ so $0 < K = s - r$ and the first iteration will be a rational polynomial with $P' = a_0$ and $Q' = b_0 + \ldots + b_K x^K$

The matrix could have two possibilities: its rank is equal to $K+1$ or $K+2$. We can verify one of them thanks to its construction and the guarantee that coefficients are non-zero; As we can see, the last $K+1$ columns are linearly independents. If the rank of the matrix A_0 is equal to $K+1$, the above system will have a solution with $b_0 = 1$: we have $K+1$ unknown so we need a system of rank $K+1$. The solution is the non trivial element in the Kernel of the matrix A_0 while the Gaussian elimination can produce the solution. Therefore, the rational polynomials could be simplified to the reduced one whose coefficients are located in the element of the kernel.

$$(a_0 + \ldots + a_K x^K, 1) = (p_0 + \ldots + p_r x^r, q_0 + \ldots + q_s x^s)$$

If not, the Rational polynomial (P, Q) can not be reduced to a rational polynomial with the considered degrees. In this instance, we increase the order of the two supposed polynomials, so the reduced fraction could be

$$(P' : (a_0, a_1, \ldots, a_{K+1}), Q' : (b_0, b_1))$$

and we will repeat the previous step to verify if this couple is the reduced one of the initial fractions (P, Q).

2.3. kth iteration

To present the iteration in general, let's suppose that we are at the iteration k, i.e., that the order of polynomials in the reduced fraction is at maximum k for Q' and $k+K$ for P' [3] (P' has $k+K+1$ unknowns and Q' has k [4] unknowns as b_0 is considered to be unitary).

Let insert the new couple in the relation (1). We obtain the following relation:

$$(p_0 + ... + p_r x^r)(b_0 + ... + b_k x^k) - (q_0 + ... + q_s x^s)(a_0 + ... + a_k x^k + \ldots + a_{k+K} x^{k+K}) = 0$$

Expanding this equation and assembling coefficients of each monomial will lead us to the equation that generates the system. And that it is true in case when the couple (P', Q') verify the relation (1):

$$(p_0 b_0 - q_0 a_0) + (p_1 b_0 + p_0 b_1 - q_1 a_0 - q_0 a_1)x + \ldots = 0$$

As the equation is so long, it will be decomposed in a system of equations that each one is the result of each monomial in this relation. We note that equality is not ensured; i.e. we asked to be equal and we will verify it. The following matrix is the matrix' form of the system of the equation:

$$A_k X_k = 0 \tag{8}$$

[3] This is the case when $r - s = K > 0$.

[4] if $s - r = K > 0$, P will have $k+1$ unknowns and $Q\prime$ has $k+K$ unknowns

A_k is a matrix of size $(n+k+1, 2k+K+2)$. We should note that $n = \max(r,s)$[5].

$$A_k = \begin{pmatrix} p_0 & 0 & & \dots & -q_0 & 0 & & & \dots \\ p_1 & p_0 & & \dots & -q_1 & -q_0 & & & \dots \\ p_2 & p_1 & 0 & \dots & -q_2 & -q_1 & 0 & & \dots \\ \vdots & \vdots & & & \vdots & \vdots & & & \\ p_K & \dots & p_0 & \dots & -q_K & \dots & -q_0 & 0 & \dots \\ \vdots & \vdots & & & \vdots & \vdots & & & \\ p_k & p_{k-1} & \dots & p_0 & -q_k & -q_{k-1} & \dots & -q_0 & 0 \\ \vdots & \vdots & & & \vdots & \vdots & & & \\ p_s & p_{s-1} & \dots & p_{s-k} & -q_s & -q_{s-1} & \dots & -q_{s-k} & q_{s-k-K} \\ p_{s+1} & p_s & & p_{s-k+1} & 0 & -q_s & \dots & -q_{s-k+1} & q_{s-k-K+1} \\ \vdots & \vdots & & & \vdots & \vdots & & & \\ p_r & p_{r-1} & \dots & p_{r-k} & 0 & 0 & -q_s & \dots & -q_{s-k} \\ 0 & p_r & \dots & p_{r-k+1} & 0 & 0 & & \dots & -q_{s-k+1} \\ 0 & 0 & \dots & p_{r-k+2} & 0 & 0 & & \dots & -q_{s-k+2} \\ \vdots & \vdots & & & \vdots & \vdots & & & \\ 0 & 0 & \dots & p_r & 0 & 0 & \dots & 0 & -q_s \end{pmatrix} \tag{9}$$

The vector of unknowns X_k with $2k+2+K$ elements is presented.

$$X_k = (b_0, \dots, b_k, a_0, \dots, a_k, \dots, a_{k+K})^T \tag{10}$$

relation with the subresultant. A_k is the submatrix used by Collins [27] where its determinant is the $0th$ coefficients of the remainder in kth step in the euclidean division.

2.3.1. Construction the A_k matrix

To simplify the issue, the matrix A_k could be constructed through the matrix A_{k-1} by adding two columns: the first will be the vector's coefficient of the polynomial P and which is added in the $k+1^{th}$ column starting at the $k+1^{th}$ row and fulfilling all the above case by 0. The second will be the vector's coefficients of the $-Q$ polynomial; this one is added at the end of the matrix starting from the $k+K+1^{th}$ row and fulfilling the rest of all empty cases by 0.

2.3.2. Ending the loop

Once we assemble the matrix A_k, we will test if his rank whether or not is maximal; i.e.

$$rank(A_k) = 2(k+1) + K$$

[5]here $n = r$. When $0 < K = s - r$,so $n = s$

If this is the state, therefore the system does not have a solution; i.e. the Rational polynomial with the proposed degrees is not the reduced one. Thus, we continue to construct the matrix A_{k+1} and ask if its rank is not maximal.

We continue this loop and test if A_k for $k = 0...m$ has a maximal rank until getting one of the two options:

1. the first one is when $k = m$ and so the given fraction is irreducible and the output will be the same entries' vectors.
2. The second one is when we do not have the maximal rank. Therefore the proposed fraction (P', Q') is the reduced one and we can compute its coefficients by solving the system considering that b_0 is unitary.

Remark 1. *When we deal whether, with multiplication with finite field or large time integers, the computation of the rank is associated with the Echelon form of the matrix and no confusion with both multiplications. We will use the SVD technique when we deal with floating-point multiplication.*

2.3.3. Computing of coefficients

Once we get the matrix A_k with no maximal rank, coefficients of the reduced fractions will be the kernel of this matrix. Using the Gaussian elimination, we can get the kernel element which is of order 1. This step will be used for one time: this adds to the complexity of the algorithm.

When the multiplication is in finite fields or large integerse, we use the Gaussian elimination to get the kernel of the matrix. Technically, it consists of solving a linear system

$$AX = b$$

when matrix A is related to A_k.

In the case of floating-point, we use the SVD to compute the rank of the matrix because of its dynamics with over-floating point issues.

For more details, let's consider that Q has the minimum order of non-zero coefficient. In this case, b_0 will be unitary. As the first column in the matrix A_k is multiplying b_0, it will be considered as the right-hand side of the system we need to construct and solve to find coefficients. Matrix A is established from the rest of A_k after removing the first column. Let's take the medium block of $2k + K + 1$ rows if the $n + k + 1$ rows. The reason why we took this block is that the block contains the minimum number of zero elements: we prevent to have a sparse matrix to not wrong the computation of coefficients. A is a square matrix with shape $2k + K + 1$ and X is a vector of $2k + K + 1$ (k elements of Q' without b_0 and $k + K + 1$ elements of P')

$$X = (b_1, \ldots, b_k, a_0, a_1, \ldots, a_{k+K})^T \tag{11}$$

and b are the $2k+K+1$ elements of the first column of A_k choosing from the same block of the matrix A and multiplying by -1

$$b = -(p_0, p_1, \ldots, p_k, \ldots)^T \tag{12}$$

The following theorem will ensure that the system has a unique solution. In the case when $q_0 = 0$, we will find the reduced rational polynomial for its inverse; i.e. the coefficients will be exchanged.

Theorem 1. *If*

$$rankA_k \neq 2(k+1) + K \tag{13}$$

then the system

$$AX = b \tag{14}$$

defined above, has a unique solution.

Proof. Let's verify this theorem for $k = 0$ and suppose that the rank of A_0, for a non zero polynomial's coefficient, is not $K+2$. So it is equal to $K+1$ because of the structure of the matrix: the last $K+1$ columns form is a lower triangular with non-zero diagonals coefficients. That is meaning that the matrix A appearing in the system (14) is inverted and so the solution is unique with unitary b_0. The proof is done for $k = 0$.

We continue to test if (13) is verified, and let's suppose that, at the $(k-1)^{th}$ iteration, the test is not verified; i.e. that

$$rank(A_{k-1}) = 2k + K$$

and at the kth iteration, it is verified. As $2k+K = rank(A_{k-1})$ and $rank(A_k) < 2(k+1)+K$, so

$$2k + K = rank(A_{k-1}) < rank(A_k) < 2(k+1) + K$$

using lemma 1, therefore

$$rank(A_k) = 2k + 1 + K$$

and the equation (14) has a unique solution. □

To conclude the proof of the theorem, we need to verify the following lemma:

Lemma 1. *For the matrix A_k constructed via the P and Q coefficients,*

$$rank(A_{k-1}) < rank(A_k) \tag{15}$$

Proof. As A_k is constructed via A_{k-1}, so we have the possibility that

$$rank(A_{k-1}) \leq rank(A_k)$$

Holding that we have an equal situation. That is meaning the two columns added, which they are the polynomial's coefficients of P and Q are a linear combination of columns of

A_{k-1}. As the first two elements of the two added columns are zeros and the first row of the rest of the matrix contains non-zeros coefficients, therefore no linear combination exists with non-zero coefficients.

In the other side, the end of these two columns contain at least one non-zero elements, and the last row of the rest of the matrix is all zero, so it is impossible to find so linear combination to obtain one of the two elements. Therefore, a contradiction of the thesis which supposes that both ranks are equal.

□

We presented above the algorithm to reduce rational polynomial, which is suitable for whose coefficients are in finite field or are large integers. In the next section, the algorithm is presented in the case of real numbers. In this case, SVD is employed to overcome over-floating errors.

2.3.4. SVD methods

In this part, we will show how SVD methods will be useful to find the reduced rational polynomials. Once we construct the matrix A_k, we apply the SVD method to find this following form of the matrix:

$$A_k = U_k \Sigma_k V_k^T \tag{16}$$

Where Σ_k is a matrix containing singular values, U_k and V_k are orthonormal matrix such that:

$$A_k v_i = \sigma_{ii} u_i$$

Once the SVD is applied, the rank is the number of the values in the diagonal of Σ_k that the value is non-zero. For floating-point operation, a fixed tolerance *tol* is given and the rank is the number of singular value that its magnitude is greater than the tolerance *tol*.

$$rank(A_k) = card(S > \max(S) * (n + k) * tol)$$

Once we get that the rank of the matrix A_k is not full, so the set of singular values contains a zero one. This mean that for vector v_0 in V and u_0 in U, we have

$$A_k v_0 = 0 \tag{17}$$

This equation verify the relation (8) and we have the unknown vector $v \in \mathbb{R}^{2k+K+2}$ which contains coefficients of rational polynomials.

Remark 2. *We can see that A_{m-1} is the Sylvester matrix [30] for the two polynomial P and Q. Let recall that each matrix of $\mathcal{M}_{r,s}$ is a linear application from $\mathbb{R}^r$ to $\mathbb{R}^s$. In our case A_{m-1} is an endomorphism of $\mathbb{R}^{r+s}$ and if the determinant is zero thus the couple (P,Q) has a GCD different than 1. This refers to have a zero singular value. The order of the GCD polynomials is equal to the number of zeros singular values of A_{m-1}.*

2.4. Algorithms

In this part, we will present two algorithms to compute the reduced Rationals polynomials. The first one, which use Gaussian elimination to determine rank and kernel base, is addressed to the multiplication in finite fields and large integers. The second one uses SVD methods and dedicated to floating-point operation. It computes the rank by computing k singular values of the matrix A_k and to compute also k singular vectors.

2.4.1. Algorithm 1

The above-explained strategy is presented in the following algorithm. We provide a fraction which is defined by the couple $(P : (p_0, .., p_r), Q : (q_0, ..q_s))$, and the algorithm will produce the reduced fraction (P', Q'). Note that the matrix A_k is build via a function that we name it *Construct* which takes vectors $(p_0, .., p_r), (q_0, .., q_s)$, orders of two polynomials r and s. The integer k represents the iteration of the algorithm.

Algorithm 1 Find $(P', Q') = \text{Reduced}(P, Q)$

Require: $P : (p_0, \ldots, p_r), Q : (q_0, \ldots, q_s)$
  $n \leftarrow \max(r, s)$
  $K \leftarrow |r - s|$
  $k \leftarrow 0$
  $A_k \leftarrow \text{Construct}(P, Q, k)$
  while $rank(A_k) = 2(k + 1)$ **do**
    $k \leftarrow k + 1$
    $A_k \leftarrow \text{Construct}(P, Q, k, K)$
  end while
  if $k = n$ **then**
    $P' \leftarrow P, Q' \leftarrow Q$
  else $\{k < n\}$
    if $q_0 \neq 0$ **then**
      solve the system (9) with $b_0 = 1$
    else $\{p_0 \neq 0\}$
      solve the system (9) with $a_0 = 1$
    end if
  end if

2.4.2. Algorithm 2

SVD methods could be translated in the above algorithm, we can directly determine the coefficients of the reduced rational polynomial via the orthonormal matrix V: the last vector related to the singular value is the vectors containing at the first the $k + 1$ coefficients of Q' and the rest are the coefficients of P'.

Algorithm 2 Find $(P', Q') = \text{Reduced}(P, Q)$

Require: $P : (p_0, \ldots, p_r), Q : (q_0, \ldots, q_s)$
  $n \leftarrow \max(r, s)$
  $K \leftarrow |r - s|$
  $k \leftarrow 0$
  $A_k \leftarrow \text{Construct}(P, Q, k)$
  while $rank(A_k) = 2(k + 1)$ **do**
    $k \leftarrow k + 1$
    $A_k \leftarrow \text{Construct}(P, Q, k, K)$
  end while
  if $k = n$ **then**
    $P' \leftarrow P, Q' \leftarrow Q$
  else $\{k < n\}$
    compute singular vector of zero s.value
  end if

2.4.3. Complexity

The exact complexity of the algorithm proposed here is decomposed on two parts: the complexity of rank's computation of the A_k multiplied by $deg(Q) - deg(GCD)$. For a matrix A of dimension $n \times m$, the complexity of rank computation is of $0(rnm + r^w)$ where r is the rank of A and $w > 0$. The complexity of Gaussian elimination is added at the end, whose complexity for a matrix of the same dimension could be considered at the same complexity of computing the rank. Therefore, the more the degree of GCD is big, the less the algorithm is complex. The complexity is given in the next theorem

Theorem 2 (Complexity of the Algorithm 1)**.** *For a given polynomials P and Q with degree n and m respectively and having a GCD G of degree d, the complexity of the algorithm is of*

$$\sum_{k=1}^{m-d} O(2k\ r_k(n + k) + r_k^{w_k}) \tag{18}$$

where r_k is the rank of A_k and $w_k > 0$

Proof. According to the algorithm, the rank of the matrix rank's computation is done all A_k matrix when $k = 1..m - d$. The last sum is the complexity of the Gaussian elimination to establish the kernel space. □

The complexity could be bounded by the complexity of SVD methods for a $n \times m$ matrix. We will consider cases when the matrix is square; i.e. the two polynomials of the fraction have equal degree ($n = m$).

Theorem 3. *For two polynomials P and Q of order n and m respectively ($n = m$), the complexity of the algorithm 1 proposed above is*

$$O(2k(n + k)) + O((2k + 1)^2)$$

where $k = m - d$ is the iteration we end the loop and d is the order of GCD polynomial.

Proof. As known, for a $n \times m$ matrix ($m < n$), the complexity of computing k singular value is O(kn) [31]. In our case, the matrix A_k is of size $r + k + 1, 2k + K + 1$ (here $K = 0$). Then we stop the algorithm when we find that the last singular value is zero. So we need to compute $2k + 1$ singular value. Thus its complexity is of $O(2k(r + k))$. After, we need to solve the system of $2k + 1$ unknown where the complexity is $O((2k + 1)^2)$. □

We can reduce the complexity of this algorithm by escaping the solving of the system and computing just the singular vector relative to the zero singular value. Thus, a simplified version of the algorithm is presented in the algorithm 2. Thanks to [31], a fast algorithm to compute k singular value and singular vectors of a matrix with size $n \times m$ is done with $O(kn)$ time. So

Theorem 4. *The complexity of algorithm 2 proposed above could be reduced to*

$$O(2k(n + k))$$

3. Experimental tests and comparison

In this part, we will compare the efficiency of both algorithms for computation time and precision. As far as computation time is concerned, the comparison is made on the multiplication of finite field, large integers and fractions field. But, there is no comparison between them regarding the precision of outputs. The comparison of accuracy and robustness is made in the arithmetic of floating-point operations

3.1. Time comparison

In this part, we conduct experimental tests to see the efficiency of Algorithm 1 to elaborate the reduced fraction. A comparison with Brown's algorithm will be presented. Brown's algorithm, which is used for the comparison here, was proposed by Collins [27] and improved by Brown [24]. Note that Brown's algorithm calculates the gcd of two polynomes and not the reduction. And vice versa, the algorithm proposed here calculates the direct reduction and not their GCD. Once, the comparison is made on time needed to calculate the reduced polynomials than on the gcd calculation. In these cases, the Euclidean division is necessary for one of them to obtain the gcd if we proceeded by our algorithm or to obtain the reduced polynomials if we proceeded by Brown's algorithm. We have chosen to compare on time needed to calculate the reduced polynomials given our application.

Three comparison sections; in the case where the multiplication is done in the finite field (here we choose F_2), a second section that of large integers multiplications and a third with multiplication in fraction field.

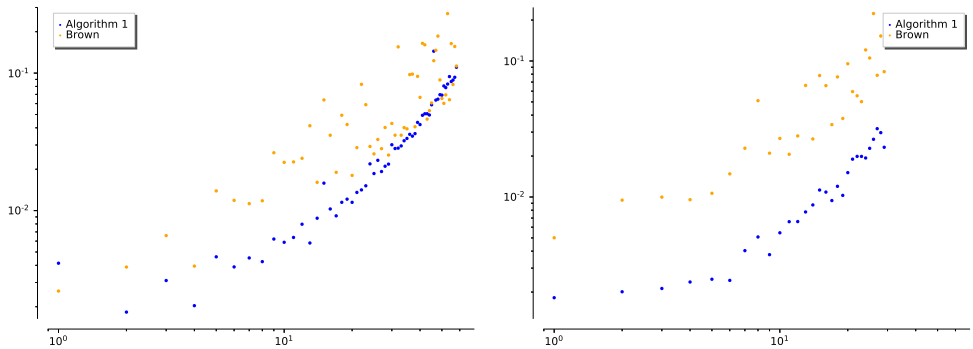


Figure 1: Time computation to accomplish reducing of fractional polynomial in finite field F_2 with randomly coefficients and when $deg(P) = deg(Q)$. Left figure is when there is no idea about GCD and the right is when GCD increase with the degree of P

3.1.1. Finite field multiplication

A series of polynomials with different degrees and random coefficients in F_2 are elaborated. We have no idea about the degree of their GCD, and we will try to reduce these polynomials using both algorithms: Brown's one and the one proposed here.

The left figure in figure 1 shows the computation time elapsed, in logarithmic scale for the x-axis(n) and y-axis (time computation), by both algorithms and that in the case where P and Q have the same degree. The right figure shows the same comparison but in the case when the degree of the GCD increases with n. We can see how is different the time of computation in the right figure regarding the time with Brown's algorithm. These results are coherent with the theory that the algorithm is more efficient in the case where GCD is greater than 1.

In figure 2, we present two table-like figures which represent the elapsed time for each algorithm and a series of experimental companions for different degrees of polynomials P and Q. The rows represent the degree n of P, and the column is the degree m of Q. We restrict ourselves to the lower part of the matrix ($deg(P) > deg(Q)$). The colour brown corresponds to Brown's algorithm, and the colour blue corresponds to the algorithm proposed here. The colour grey corresponds to time ratio of Algorithm 1 and Brown (Algorithm 1/ Brown). The first three tables correspond to the case where there is no idea about the GCD, and the second is when the GCD's degree increase with $deg(P)$. Table of ratio, relative to the case when there is no idea about GCD, show that the time computation needed in the Algorithm 1 is less than the time needed by Brow and this time increase when n increase lose to the time to Brown's algorithm when the degree of P and Q are close.

3.1.2. Large integers multiplications

We proceed to the same recipe of comparison. We mention that, as for the comparison realised with the finite field, we have taken a set of rational polynomials such that multiplication of large integers, and with different degree of P and Q such that $deg(P) = 1...64$

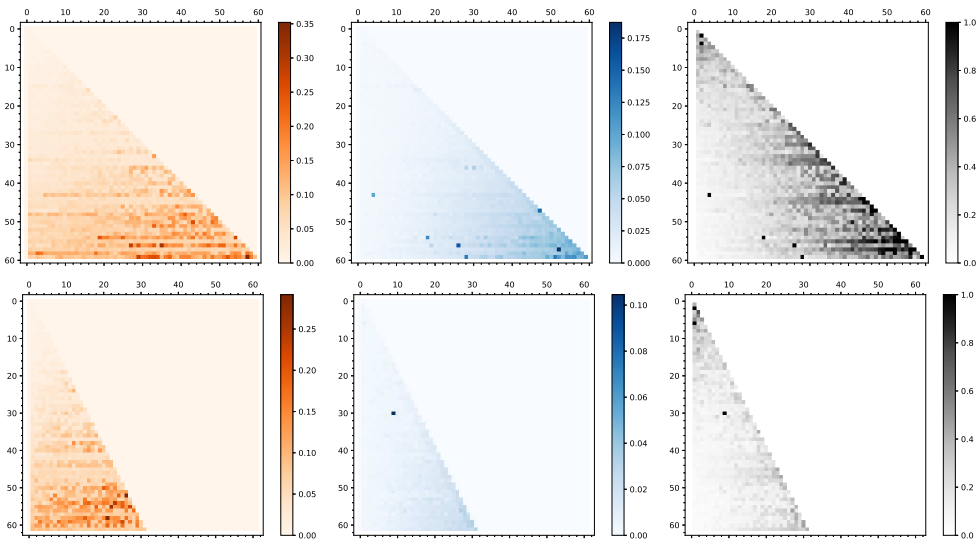


Figure 2: Time table computation for reducing series of fractions polynomials. upper row: no idea about GCD. lower row: deg(GCD) increase with the degree of P. The brown color represent the time with Brown algorithm. Blue color is for Algorhtm 1 and the Grey is for the ration(Algorithm 1/ Brown).

and $deg(Q) = 1...64$. We compare the time computation needed to finish applying both algorithms. We plot the figures on the logarithmic scale for x and y-axis. The object is to show the slope of the asymptotic evolution to be able to read the order of complexity of each algorithm.

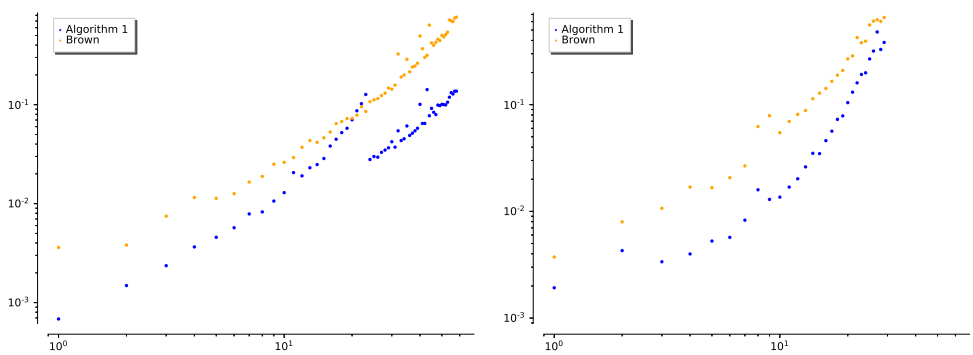


Figure 3: Time computation to accomplish reducing of fractional polynomial with large integers coefficients chosen randomly and for $deg(P) = deg(Q)$. Left figure is when there is no idea about GCD and the right is when GCD increase with the degree of P

In the figure 3, we plot the results relative to these comparison. In the left figure, we show the comparison between algorithms when the degree of the GCD is 3. The right figure corresponds to the case when the degree of GCD increases with the degree of P. Recall that the degree of P and Q are equal. A special remark of the plotting in the case when the degree of GCD is fixed at 3. This is also remarked in the time table of Algorithm 1 (Blue table in the first row of the figure 4) In the same tables of figures, the first row is relative to the time

table computation when the set of rational polynomials have all a GCD with a degree of 3. The second one is when the degree of GCD increase. In these time tables, rows correspond wen degree of P is fixed and Q change. Columns correspond to time computation when the degree of Q is fixed and P change.

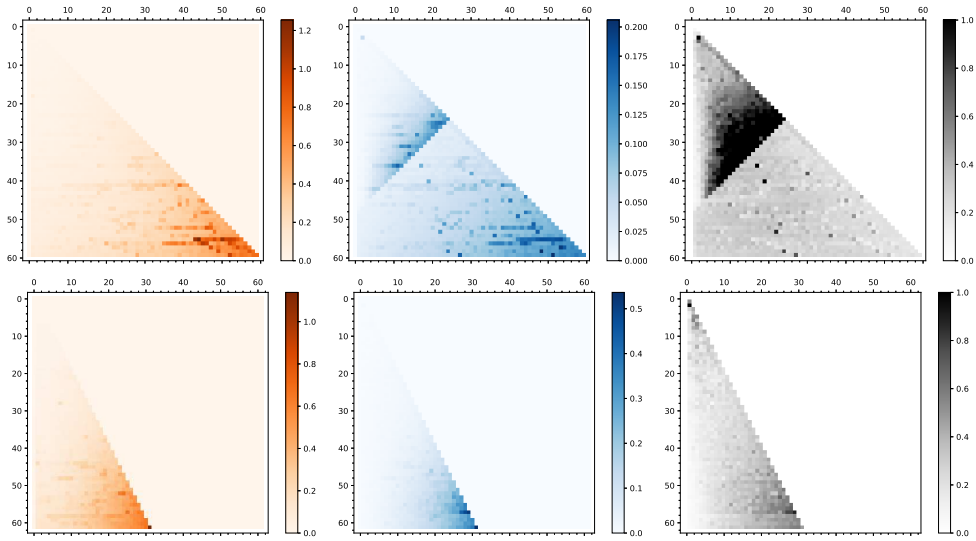


Figure 4: Time computation for multiplications with large integers coefficients. The brown color represent the time with Brown algorithm. Blue is for Algorhtm 1 and the Grey is for the ration(Algorithm 1/ Brown).

3.1.3. Fractional domain

In the last comparison, we took coefficient int the Fractions domain—hereafter the time's tables and figures which plot the results of time computation. The results are not the same as the multiplication for large integers. In figure 5 we plot in a logarithmic scale of x and y-axis the time computation needed by both algorithm to accomplish reducing rational polynomial having the same degree of numerator and denominator. In the left one, no idea about the degree of GCD. In the right one, the degree of GCD increases with n (same as before).

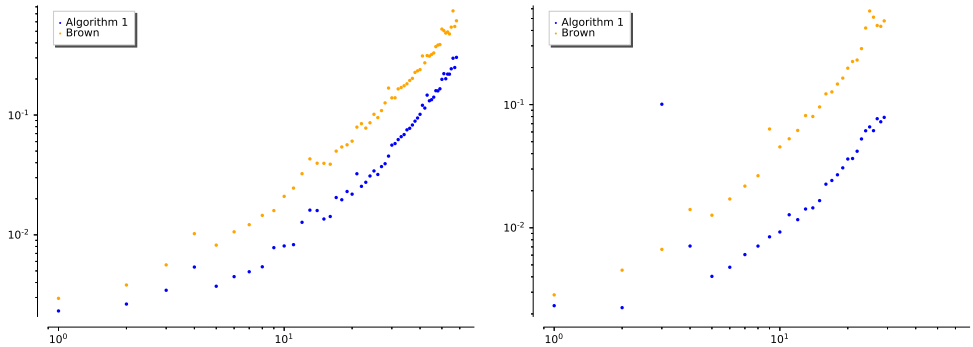


Figure 5: Comparison of time computation in case when multiplication is in Fractional domain.

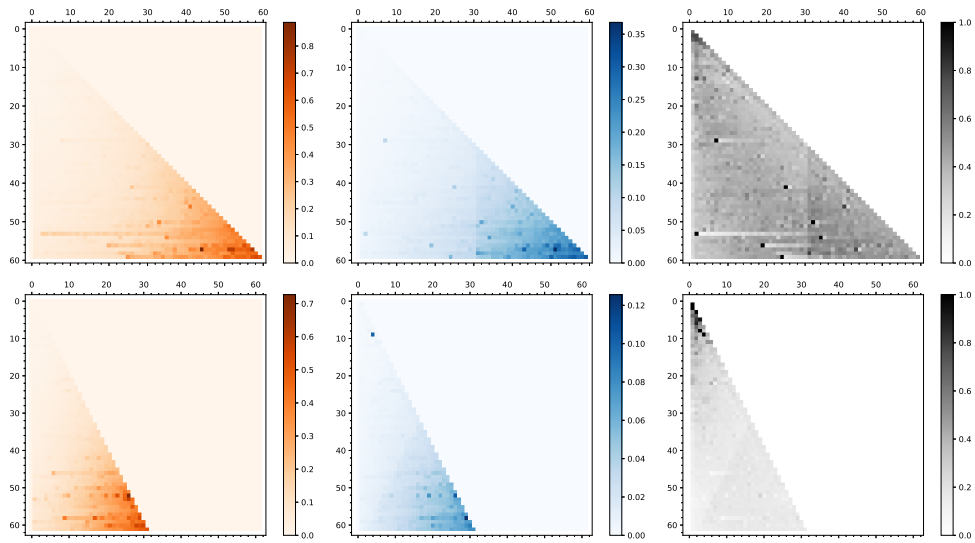


Figure 6: Time table computation and ration for fractional domain multiplication. Upper row: no idea about degree of GCD. Lower row: the degree of GCD increase with n.

To conclude this part, the performance of algorithm 1 is at the same order of the Brown's algorithm: In this case, Gaussian elimination or SVD methods are employed m times on every A_k matrix $k = 0, ...s$. This application renders the algorithm more complex than when iteration does not go to the end. This simplification is the case when the fractional polynomial is reducible: when the order of GCD polynomial is greater than 1, the Algorithm 1 start to be more effective because of the application of Gaussian elimination or SVD on a matrix with a size smaller than Sylvester's one.

3.2. Accuracy and robustness

In this section, we compare both algorithms, Brown's algorithm and Algorithm 2, on the issue of reducing fractional polynomials with floating-point multiplication. The accuracy and robustness of them are not under discussion when multiplication is in the finite field or large integers because effectively, both conduct to the same result: computing the rank of a matrix, Gaussian elimination and computing the determinant to construct the PRS are algebraic operations. Nevertheless, when arithmetic with floating points, rounded floating-point numbers and rounded error need to be considered.

Let's take two polynomials P and Q given as below:

$$P = x^3 + 3x^2 + 3x + 1 \qquad Q = x^2 + \epsilon x - (1 + \eta) \tag{19}$$

The polynomial Q is given with no rounded error ϵ at the coefficient multiplying the monomial x. The improved PRS algorihtm which is proposed by [24] is applied to compute the GCD of P and Q. In this case the arithmetic is the floating point arithmetic and we see the step

of the algorithm. In the following table, we will illustrate the step of the algorithm which in each row will be the remainder sequence constructed via P and Q

$$\begin{array}{cccc} 1 & 3 & 3 & 1 \\ \hline & 1 & \epsilon & -1-\eta \\ \hline & & 4+\eta-3\epsilon+\epsilon^2 & 4-\epsilon+3\eta-\eta\epsilon \\ \hline & & & -\epsilon^3-3\eta\ \epsilon^2-3\eta^2\epsilon-\eta^3 \end{array} \tag{20}$$

We can see that the error is o(ϵ^3). For a polynomial P of order n, and Q of order m such that the euclidean division is given as below:

$$P = SQ + R$$

we can prove that

$$P = \tilde{S}(Q + \epsilon x^a) + R + o(\epsilon^n)$$

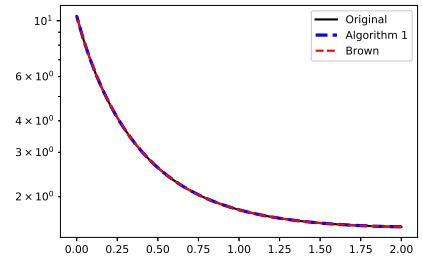
where $\tilde{S} = S + o(\epsilon^{n-a}, x^{n-a})$. This means that for any perturbation in the coefficients of Q, it is diffused in the algorithm of the PRS. If nothing is done to treat this error, like a filter or otherwise, the PRS algorithm will not be able to compute a gCd and then to reduce rational polynomial which will conduct us to an over floating of operations in arithmetic floating points. A simple way to treat these type of error is putting a filter on the remainder and test if all coefficients are less then a limit. Once we accomplish all of these steps, we will proceed to test the robustness of Brown's algorithm in the next paragraph.

In the other hand, the Algorithm proposed here and which uses rank computation as a criterion can intrinsically deal with these types of issues. It is sufficient, to use SVD decomposition and to compute the rank with relative to a tolerance given at the beginning.

To compare them, we will, from a given fractional polynomial, reduce it and compare its reduced one with the original and on a subset of its function's domain. In the following figures, we present three tests of reducing fractional polynomial. Coefficients are given randomly. Brown's and Algorithm 1 are applied. Original coefficients (before applying any of them) and ones after applying are plotted. Next to the, I plot the figure of their functions in the range $[0, 2]$.

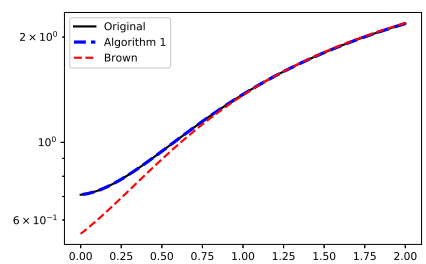
In the first test, the reducing rational polynomial by Brown's algorithm, has the same plot of the original one and the one with the Algorithm 1. it seems like that no difference with the plots.

	Original	*Brown*	*Algorithm*1
	0.01840888	1.78089695	0.87031592
	0.0044131	0.42692875	0.20863806
P	0.02115195	2.04626494	1
	0.0044494	0.43044014	0.21035406
	0.00101453	0.09814695	00.04796395
	0.00176295	0.17055032	0.08334714
	0.01033686	1.	0.48869527
Q	0.0101883	0.98562829	0.48167189
	0.00558991	0.54077477	0.26427407



In the next two tests, we can see that Brown's algorithm, produce a false reduced fractional which functions are not exact as the original. Nevertheless, the Algorithm, produce a rational polynomial which has the same figure of plots.

	Original	*Brown*	*Algorithm*1
	0.00605593	0.54768768	0.37014882
	0.01222366	0.46164872	0.74713045
P	0.01317623	0.79035358	0.80535297
	0.01429358	0.19887569	0.87364775
	0.00323836		0.19793418
	0.00854742	1.	0.52243257
	0.01636081	0.01081719	1.
Q	0.00342533	0.45378848	0.20936188
	0.0073892		0.45164023

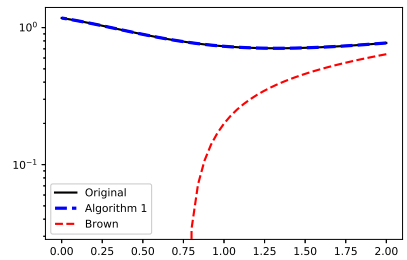


To understand the result, we show in the next table, the steps of computing the PRS.

0.00605593	0.01222366	0.01317623	0.01429358	0.00323836
0.00854742	0.01636081	0.00342533	0.0073892	
$7.01225839e-11$	$1.59470461e-10$			
$4.40560344e-15$				

We can see, the last element in the PRS is floating-point constant with order 10^{-15}. As the limit we have been fixed in the Brown's algorithm is the same order of this floating error, the algorithm considers this value as zero elements and then according to the strategy, the before the last element, which a polynomial of degree 1, is considered to be the GCD. That is why the produced fraction has bot polynomial of degree less than 1. This problem happens for the last test also.

	Original	*Brown*	*Algorithm*1
	0.00917196	0.98600671	1
	0.00234571	−1.17127546	0.25574847
P	0.00442135	0.11083152	0.48205077
	0.00143866	−0.340403	0.15685432
	0.00146626		0.15986394
	0.00780282	−0.4774229	0.85072636
	0.00545098	−0.60306091	0.59430946
Q	0.00822645	−1	0.8969133
	0.00430744		0.46963141



4. Conclusion

This work was motivated by our object to solve parameterized partial differential equations occurred in Heat and Mass Transfer in porous materials using PGD technique. That lead us to solve parameterized non-linear equations. We use Newton's methods to solve this type of equation, and rational polynomial is in need to be reduced. Getting a reduced one becomes an important issue. As we deal with a scientific programming language like Python, simplifying the rational polynomials using effective algebraic algorithms is not sufficient because of over-floating errors. So, this algorithm is proposed here to fulfil this lack.

The algorithm uses the kernel of a linear map to get the coefficients of the reduced polynomial. This linear map is iteratively built, and at each iteration, we test if the kernel contains a non-trivial element. The rank computation is used to get test the non-null of the kernel space, and the Gaussian elimination is applied to get its element. These steps are applied when the multiplication is for finite fields and large integers.

We establish a theorem to ensure the validity of the algorithm and its efficiency. The proof is presented then, and experimental tests are carried to show its efficiency regarding effective algorithm as Brown's algorithm.

Another variant of the algorithm relative to the arithmetic of floating-point is presented and which uses SVD methods. SVD methods are heavily used as tools for model order reduction. We compare this algorithm with Brown's algorithm to measure its efficiency regarding the performance about time complexity and carrying reduced rational polynomials with high accuracy. Examples which are conducted show that the time complexity is equivalent to the time complexity when the degree of polynomials increase and that for finite field and large integers multiplication. Besides, the algorithm based on SVD methods does not have problems with high values of coefficients as Brown's has. Further studies are a point of interests in case of the multi-variate fractional polynomial.

Declarations

Fundings

This work was supported by the Region and the European Union and the Ministry of Ecological and Solidarity Transition in France

Conflicts of interest

Not applicable

Availability of data and material

All data are available and they can produce directly through the code.

Code availability

Codes are availabile and could be shared at any time when readers ask. They can send a mail on the address displayed in the top of the article. Codes are programmed under Python language

Author's contribution

- Rafik Belarbi: Contribution as supervisor and as co-author: reading and revision
- Ahmad DEEB: Corresponding author.

Ethics approval

Not applicable

Consent to participate

Not applicable

Consent for publication

Not applicable

Acknowledgements

The Region and the European Union support the project < CPER-FEDER Bâtiment durable Axis 2 MADUR Project: High-performance building materials with low environmental impact, sustainable and resilient > within the framework of the ≪ Operational Program FEDER/FSE 2015-2020≫ and Energy saving certificate program of the Ministry of Ecological and Solidarity Transition "SmartReno support " 2019-2021.

References

[1] H. Padé, "Sur la représentation approchée d'une fonction par des fractions rationnelles," *ASENS*, vol. 9, no. 3, pp. 3–93, 1892. Thèse.

[2] G. A. Baker Jr. and P. R. Graves-Morris, *Padé Approximants*. Encyclopedia of Mathematics and its Applications, Cambridge University Press, 1996.

[3] A. A. Cuyt, "Padé-approximants in operator theory for the solution of nonlinear differential and integral equations," *Computers and Mathematics with Applications*, vol. 8, no. 6, pp. 445 – 466, 1982.

[4] J. P. Boyd, "Padé approximant algorithm for solving nonlinear ordinary differential equation boundary value problems on an unbounded domain," *Computers in Physics*, vol. 11, no. 299, 1997.

[5] H. Vazquez-Leal, B. Benhammouda, U. Filobello-Nino, and al., "Direct application of padé approximant for solving nonlinear differential equations," *SpringerPlus*, vol. 3, no. 563, 2014.

[6] C. M. Bender and T. T. Wu, "Anharmonic oscillator. II. A study of perturbation theory in large order," *Phys. Rev. D*, vol. 7, pp. 1620–1636, Mar 1973.

[7] B. Cochelin, N. Damil, and M. Potier-Ferry, "Asymptotic–numerical methods and pade approximants for non-linear elastic structures," *International journal for numerical methods in engineering*, vol. 37, no. 7, pp. 1187–1213, 1994.

[8] M. Afshar Kermani and M. Farzam, "Approximate solution for heat equation via the pade approximation," *Journal of Interpolation and Approximation in Scientific Computing*, vol. 2017, pp. 49–55, 11 2017.

[9] H. A. Peker, O. Karaoğlu, and G. Oturanç, "The differential transformation method and pade approximant for a form of blasius equation," *Mathematical and Computational Applications*, vol. 16, no. 2, pp. 507–513, 2011.

[10] C. Sopuerta, *Gravitational Wave Astrophysics: Proceedings of the Third Session of the Sant Cugat Forum on Astrophysics*. Astrophysics and Space Science Proceedings, Springer International Publishing, 2014.

[11] W. Hao, Y. Xiao-Peng, and Z. Ya-Nan, "Cosmological applications of padé approximant," *Journal of Cosmology and Astroparticle Physics*, vol. 2014, pp. 045–045, jan 2014.

[12] S. Peris, "Large-N_c qcd and pade approximant theory," *Phys. Rev. D*, vol. 74, p. 054013, Sep 2006.

[13] C. M. Bender and S. A. Orszag, *Advanced Mathematical Methods for Scientists and Engineers I: Asymptotic Methods and Perturbation Theory.* Advanced Mathematical Methods for Scientists and Engineers, Springer, 1999.

[14] A. Cohen, "A padé approximant to the inverse langevin function," *Rheol Acta*, vol. 30, p. 270–273, 1991.

[15] R. Jedynak, "Approximation of the inverse langevin function revisited," *Rheol Acta*, vol. 54, p. 29–39, 2015.

[16] S. Landau, "Factoring polynomial over algebraic number field," *SIAM Journal of Computation*, vol. 14, no. 1, pp. 184–195, 1985.

[17] E. R. Berlekamp, "Factoring polynomials over finite fields," *The Bell System Technical Journal*, vol. 46, no. 8, pp. 1853–1859, 1967.

[18] D. G. Cantor and H. Zassenhaus, "A new algorithm for factoring polynomials over finite fields," *Mathematics of Computation*, vol. 36, no. 154, pp. 587–592, 1981.

[19] D. Knuth, *Seminumerical algorithms.* The Art of Computer Programming, Addison-Wesley, 1969.

[20] A. Bostan, F. Chyzak, M. Giusti, R. Lebreton, G. Lecerf, B. Salvy, and E. Schost, *Algorithmes Efficaces en Calcul Formel.* Palaiseau: Frederic Chyzak, sep 2017.

[21] M. J. Encarnacion, "Computing gcds of polynomials over algebraic number fields," *Journal of Symbolic Computation*, vol. 20, no. 3, pp. 299–313, 1995.

[22] E. Kaltofen, "Greatest common divisors of polynomials given by straight-line programs," *Journal of the Association for Computing Machinery*, vol. 35, no. 1, pp. 231–264, 1988.

[23] L. Langemyr and S. MacCallum, "The computation of the greatest common divisors over algebraic number field," *Journal of Symbolic computation*, vol. 38, pp. 429–448, 1989.

[24] W. S. Brown, "The subresultant prs algorithm," *ACM Transactions on Mathematical software*, vol. 4, pp. 237–249, September 1978.

[25] M. v. Hoeij and M. Monagan, "Algorithms for polynomial gcd computation over algebraic function fields," vol. 38, p. 297–304, 2004 2004.

[26] W. S. Brown, "On euclid's algorithm and the computation of polynomial greatest common divisors," *Journal of the Association for Computing Machinery*, vol. 18, pp. 478–504, October 1971.

[27] G. Collins, "Subresultants and reduced polynomial remainder sequences," *Journal of the Association for Computing Machinery*, vol. 14, pp. 128–142, January 1971.

[28] D. Triantafyllou and M. Mitrouli, "Two resultant based methods computing the greatest common divisor of two polynomials," in *Numerical Analysis and Its Applications* (Z. Li, L. Vulkov, and J. Wasniewski, eds.), (Berlin, Heidelberg), Springer Berlin Heidelberg, 2005.

[29] N. Karcanias, M. Mitrouli, and S. Fatouros, "A resultant based computation of the greatest common divisor of two polynomials," 01 2003.

[30] A. G. Akritas, "Sylvester's forgotten form of the resultant," *Fibonacci Quart*, pp. 325–332, 1993.

[31] Chapman and Hall/CRC, eds., *Handbook of Linear Algebra*, ch. Computation if Singular value Decomposition, pp. 45.1–45.14. discrete mathematics and its applications ed., 2016.

Figures

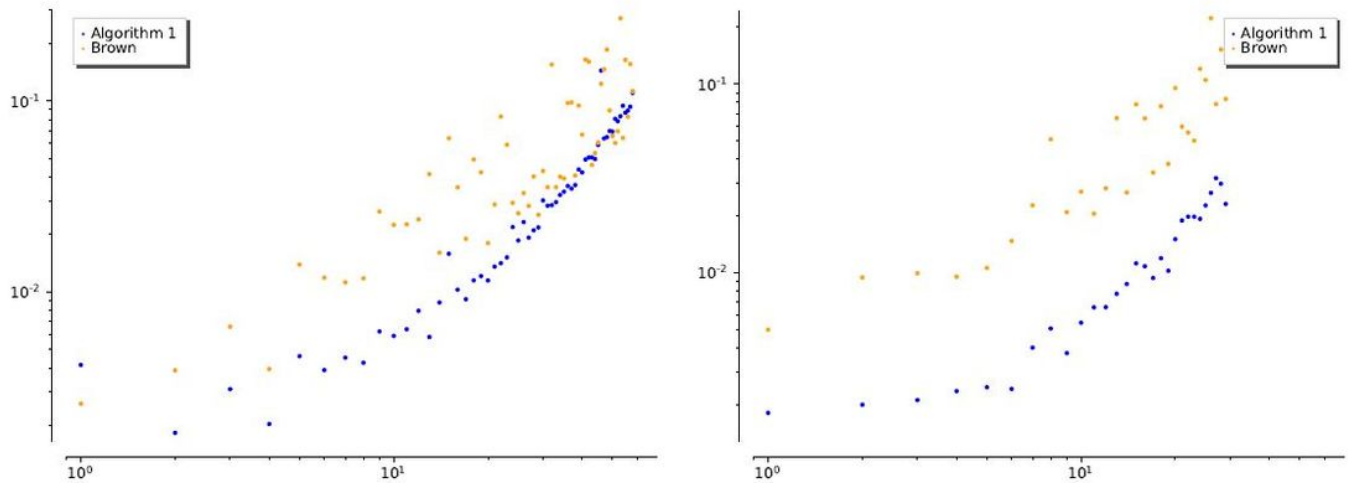


Figure 1

Time computation to accomplish reducing of fractional polynomial in finite field F2 with randomly coefficients and when deg(P) = deg(Q). Left figure is when there is no idea about GCD and the right is when GCD increase with the degree of P

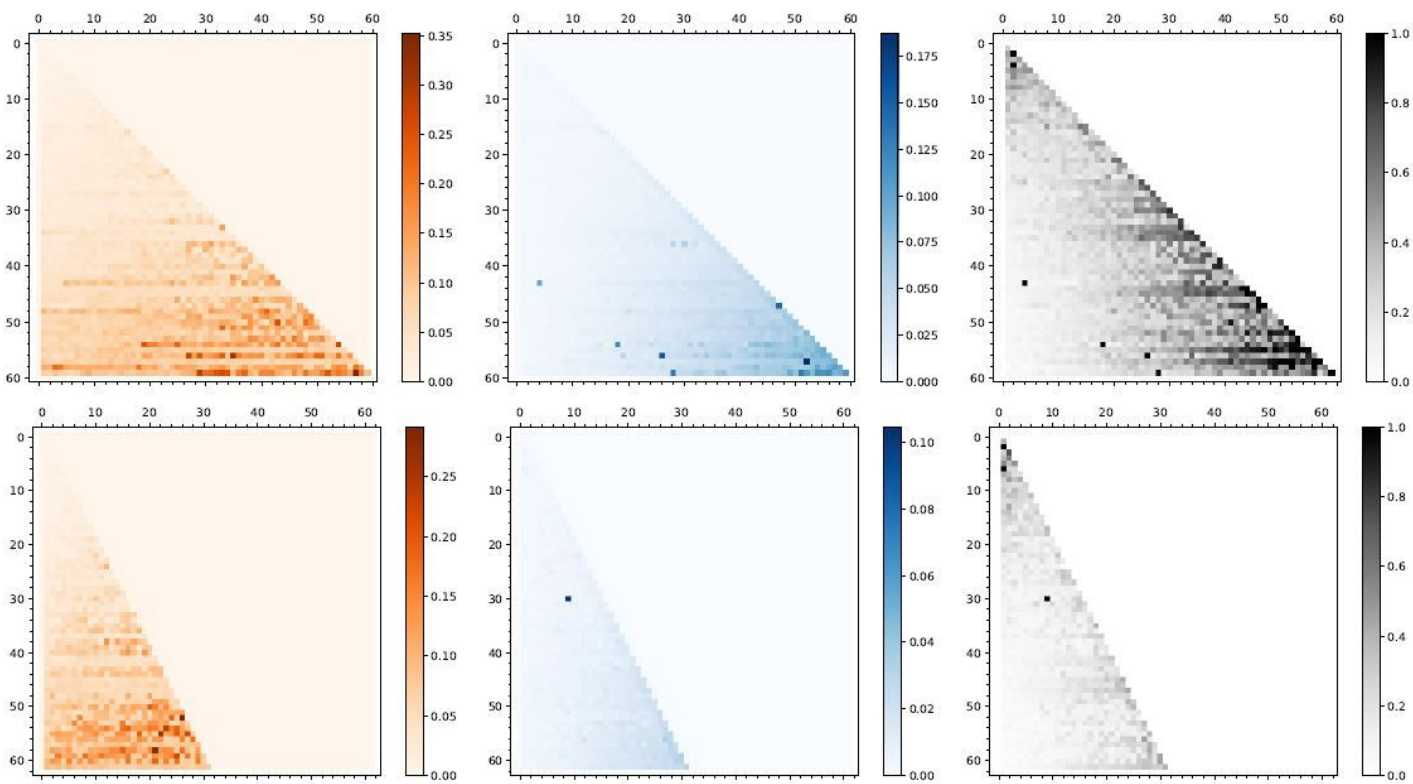


Figure 2

Time table computation for reducing series of fractions polynomials. upper row: no idea about GCD. lower row: deg(GCD) increase with the degree of P. The brown color represent the time with Brown algorithm. Blue color is for Algorhtm 1 and the Grey is for the ration(Algorithm 1/ Brown).

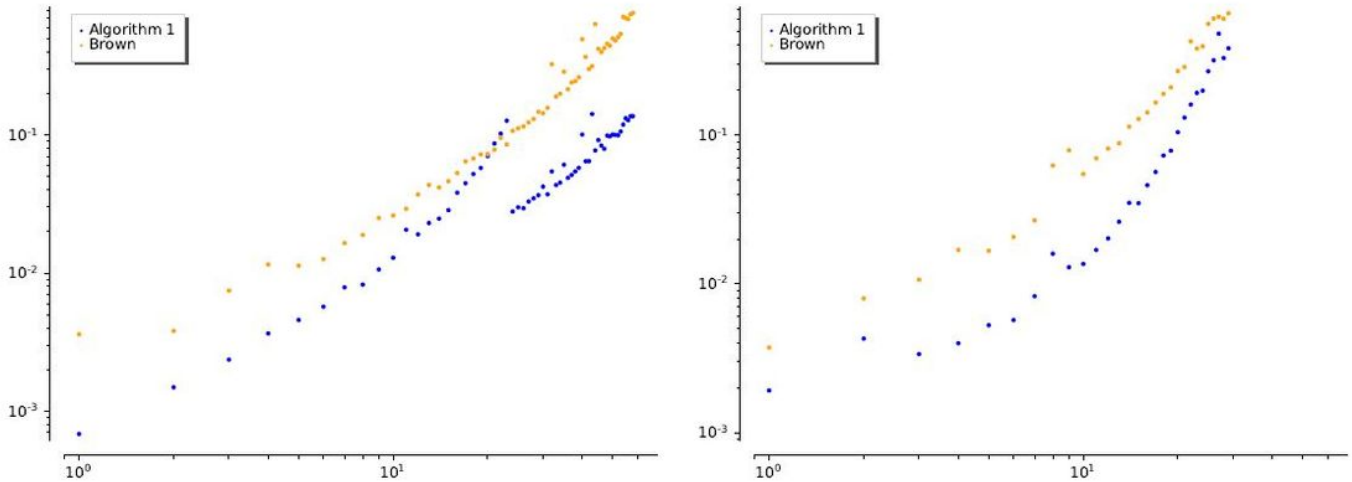


Figure 3

Time computation to accomplish reducing of fractional polynomial with large integers coefficients chosen randomly and for deg(P) = deg(Q). Left figure is when there is no idea about GCD and the right is when GCD increase with the degree of P

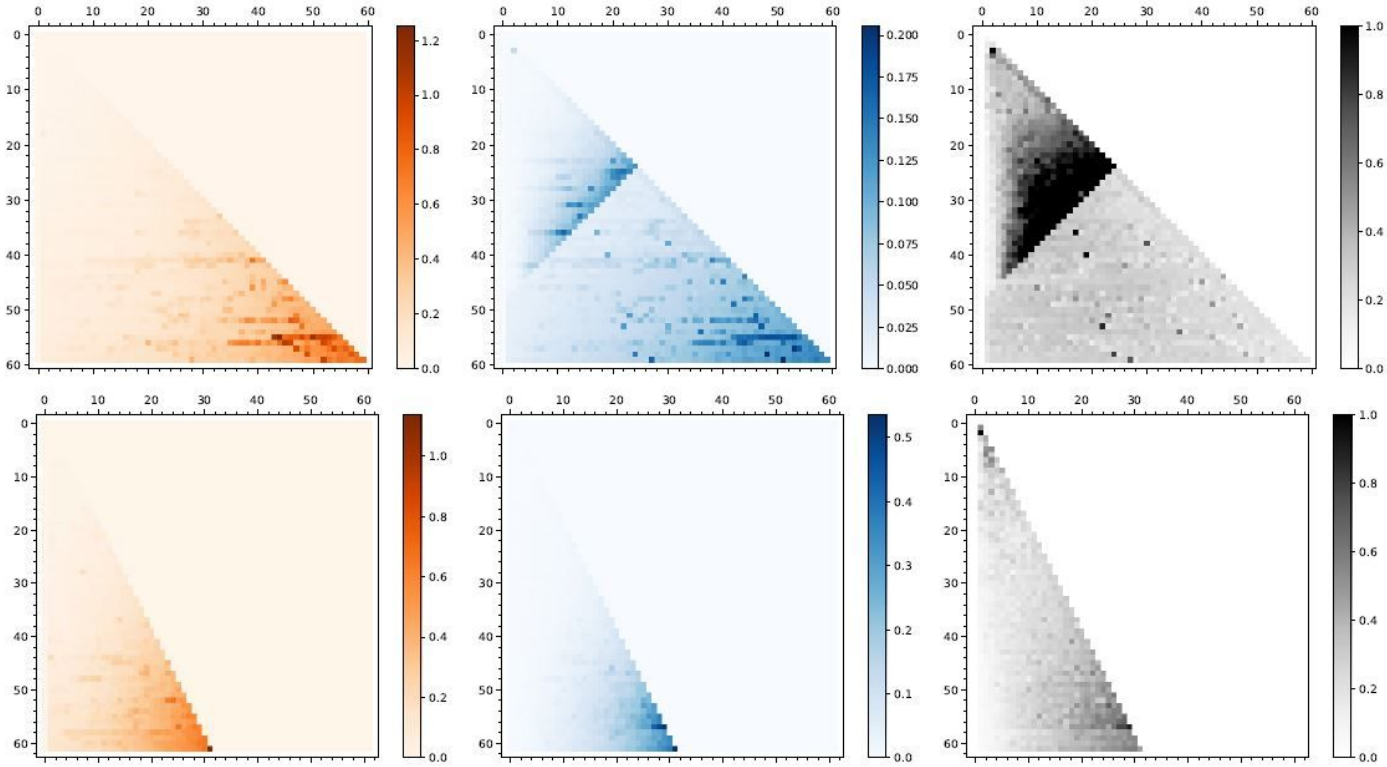


Figure 4

Time computation for multiplications with large integers coefficients. The brown color represent the time with Brown algorithm. Blue is for Algorhtm 1 and the Grey is for the ration(Algorithm 1/ Brown).

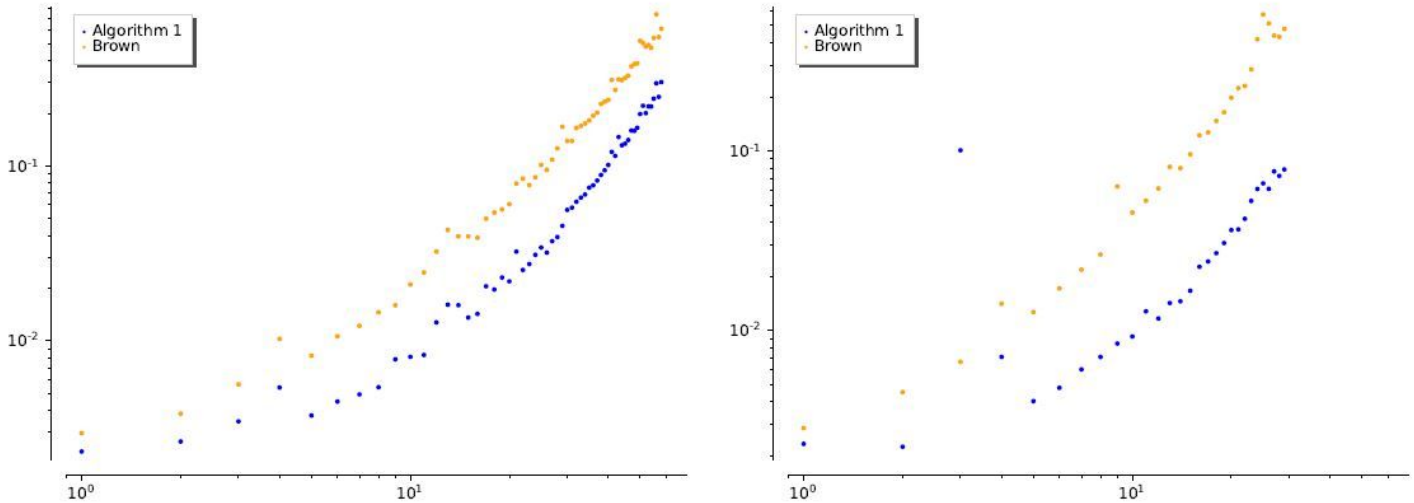


Figure 5

Comparison of time computation in case when multiplication is in Fractional domain.

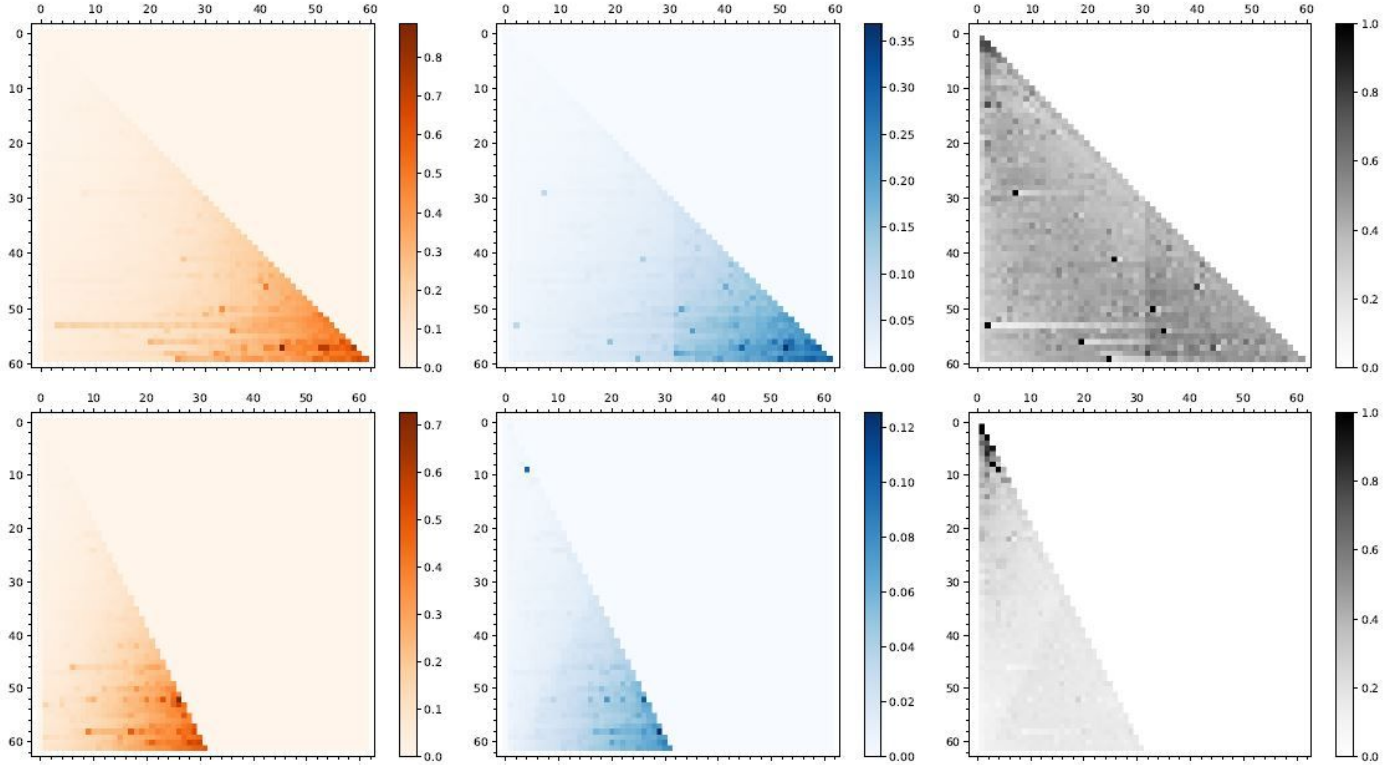


Figure 6

Time table computation and ration for fractional domain multiplication. Upper row: no idea about degree of GCD. Lower row: the degree of GCD increase with n.