Analysis of fast versions of the euclid algorithm

Eda Cesaratto, Julien Clément, Benoît Daireaux, Loïck Lhote, Véronique Maume-Deschamps, Brigitte Vallée

Analysis of fast versions of the Euclid Algorithm

Eda Cesaratto [*] Julien Clément [†] Benoît Daireaux[‡] Loïck Lhote [§]
Véronique Maume-Deschamps [¶] Brigitte Vallée [‖]

To appear in the ANALCO'07 Proceedings

Abstract

There exist fast variants of the gcd algorithm which are all based on principles due to Knuth and Schönhage. On inputs of size n, these algorithms use a Divide and Conquer approach, perform FFT multiplications and stop the recursion at a depth slightly smaller than $\lg n$. A rough estimate of the worst–case complexity of these fast versions provides the bound $O(n(\log n)^2 \log\log n)$. However, this estimate is based on some heuristics and is not actually proven. Here, we provide a precise probabilistic analysis of some of these fast variants, and we prove that their average bit–complexity on random inputs of size n is $\Theta(n(\log n)^2 \log\log n)$, with a precise remainder term. We view such a fast algorithm as a sequence of what we call interrupted algorithms, and we obtain three results about the (plain) Euclid Algorithm which may be of independent interest. We precisely describe the evolution of the distribution during the execution of the (plain) Euclid Algorithm; we obtain a sharp estimate for the probability that all the quotients produced by the (plain) Euclid Algorithm are small enough; we also exhibit a strong regularity phenomenon, which proves that these interrupted algorithms are locally "similar" to the total algorithm. This finally leads to the precise evaluation of the average bit–complexity of these fast algorithms. This work uses various tools, and is based on a precise study of generalised transfer operators related to the dynamical system underlying the Euclid Algorithm.

1 Introduction

Gcd computation is a widely used routine in computations on long integers. It is omnipresent in rational computations, public key cryptography or computer algebra. Many gcd algorithms have been designed since Euclid. Most of them compute a sequence of remainders by successive divisions, which leads to algorithms with a quadratic bit–complexity (in the worst-case and in the average-case). Using Lehmer's ideas [20] (which replace large divisions by large multiplications and small divisions), computations can be sped-up by a constant factor, but the asymptotic complexity remains quadratic. Major improvements in this area are due to Knuth [19], who designed the first subquadratic algorithm in 1970, and to Schönhage [24] who subsequently improved it the same year. They use both Lehmer's ideas and Divide and Conquer techniques to compute in a recursive way the quotient sequence (whose total size is $O(n)$). Moreover, if a fast multiplication with subquadratic complexity (FFT, Karatsuba...) is performed, then one obtains a subquadratic gcd algorithm (in the worst-case). Such a methodology has been recently used by Stehlé and Zimmermann [25] to design a Least-Significant-Bit version of the Knuth-Schönhage algorithm. According to experiments due to [5] or [22], these algorithms (with a FFT multiplication) become efficient only for integers of size larger than 10000 words, whereas, with the Karatsuba multiplication, they are efficient for smaller integers (around 100 words). A precise description of the Knuth-Schönhage algorithm can be found in [29, 22] for instance.

1.1 Previous results.

The average-case behaviour of the quadratic gcd algorithms is now well understood. First results are due to Heilbronn and Dixon in the seventies, who studied for the first time the mean number of iterations of the Euclid Algorithm, then Brent analysed the Binary algorithm [4], and Hensley [15] provided the first distributional analysis for the number of steps of the Euclid Algorithm. Since 90, the Caen Group [26, 28, 27] has performed an average-case analysis of various parameters of a large class of Euclidean algorithms. More recently, distributional results have also been obtained for the Euclid algorithm and some of its variants: first Baladi and Vallée prove that a whole class

[*] Facultad de Ingeniera, Universidad de Buenos Aires, Argentina and GREYC, Université de Caen, France
[†] GREYC, Université de Caen, F-14032 Caen, France
[‡] IrisResearch Center, Stavanger, Norway
[§] GREYC, Université de Caen, F-14032 Caen, France
[¶] IMB, Université de Bourgogne, F-21078 Dijon Cedex, France
[‖] GREYC, Université de Caen, F-14032 Caen, France

of so–called additive costs of moderate growth follows an asymptotic gaussian law [2] (for instance, the number of iterations, the number of occurrences of a given digit, etc). This year, Lhote and Vallée [21] show that a more general class of parameters also follows an asymptotic gaussian law. This class contains the length of a remainder at a fraction of the execution, and the bit-complexity. To the best of our knowledge, there are yet few results on "efficient" gcd algorithms. In [9], the authors perform an average-case analysis of Lehmer's algorithm, and exhibit the average speed-up obtained using these techniques. But, as far as we know, there does not exist any probabilistic analysis of subquadratic gcd algorithms. This is the goal of this paper to perform such a study.

1.2 Our results. There are two algorithms to be analyzed, the $\mathcal{HG}$ algorithm and the $\mathcal{G}$ algorithm. The $\mathcal{G}$ algorithm computes the gcd, and the $\mathcal{HG}$ algorithm (for Half-gcd Algorithm) only simulates the "first half" of the $\mathcal{G}$ algorithm. We first show that these algorithms can be viewed as a sequence of the so–called Interrupted Euclidean algorithms. An Interrupted Euclidean algorithm is a subsequence formed by successive iterations of the algorithm. On an input (A, B), the plain Euclid algorithm builds a sequence of remainders A_i, a sequence of quotients Q_i, and a sequence of matrix $\mathcal{M}_i$ [see Section 2.1]. On an input (A, B) of binary size n, the Interrupted Euclidean algorithm $\mathcal{E}_{[\delta,\delta+\gamma]}$ starts at the index k of the execution of the Euclid Algorithm, as soon as the remainder A_k has already lost $\delta\, n$ bits (with respect to the initial A which has n bits) and stops at index $k+i$ as soon as the remainder A_{k+i} has lost $\gamma\, n$ supplementary bits (with respect to the remainder A_k). The $\mathcal{HG}$ algorithm is just an algorithm which simulates the interrupted algorithm $\mathcal{E}_{[0,1/2]}$. A quite natural question is: How many iterations are necessary to lose these $\gamma\, n$ bits? Of course, it is natural to expect that this subsequence of the Euclidean algorithm is just "regular" and locally similar to the "total" Euclidean Algorithm; in this case, the number of iterations would be close to $\gamma\, P$ (where P is the number of iterations of the "total" Euclid algorithm). We prove in Theorem 1 that this is the case.

For a probabilistic study of these fast variants, a precise description of the evolution of the distribution during the execution of the plain Euclid Algorithm is of crucial interest. For real inputs, we know that the continued fraction algorithm does not terminate (except for rationals ...). Moreover, as the continued fraction algorithm is executed, the distribution of reals tends to the distribution relative to the Gauss density φ, defined as

$$\varphi(x) = \frac{1}{\log 2}\frac{1}{1+x}. \tag{1.1}$$

For rational inputs, one begins with some distribution on the set of the inputs $x := A_1/A_0$ of size n, and we consider the rationals $x_k := A_{k+1}/A_k$. We focus on the first index k where the binary size of x_k is less than $(1-\delta)n$ and we denote the corresponding rational x_k by $x_{\langle\delta\rangle}$. What is the distribution of the rational $x_{\langle\delta\rangle}$? The evolution of this distribution is clearly more intricate than in the real case, since at the end of the Algorithm (when $\delta = 1$), the distribution is the Dirac measure at $x = 0$. We obtain here a precise description of this distribution (see Theorem 2 and Figure 1) which involves another density

$$\psi(x) := \frac{12}{\pi^2}\sum_{m\geq 1}\frac{\log(m+x)}{(m+x)(m+x+1)}. \tag{1.2}$$

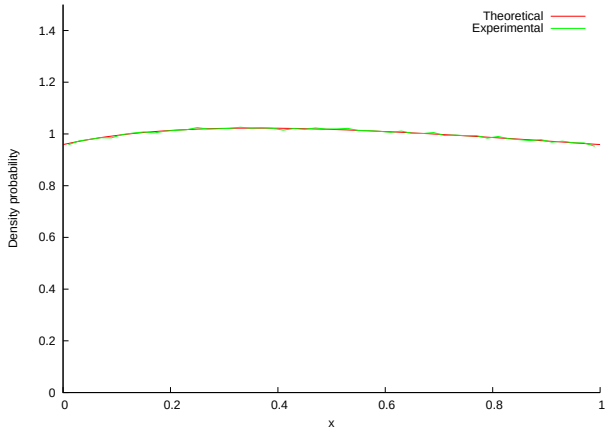

Figure 1: Density distribution of $x_{\langle\delta\rangle}$ in the case $\delta = 1/2$. This corresponds to the density distribution of the rational $x_k := A_{k+1}/A_k$ obtained as soon as $\ell(A_k)$ is smaller than $(1/2)\ell(A_0)$. The density distribution is plotted against its theoretical counterparts $\psi(x)$. Here we consider Ω_n for $n = 48$ (48 bits), the interval $[0, 1]$ is subdivided into equal subintervals of length $1/50$ to estimate the density, and 3 537 944 rational are drawn from Ω_n according to the initial density distribution φ by Monte Carlo simulation.

We also need precise results on the distribution of some truncatures of these remainders. This is done in Theorem 3. Then, the choice of parameters in the fast algorithms must take into account this evolution of distribution. This is why we are led to design some variants of the classical algorithms, denoted by $\underline{\mathcal{HG}}$ and $\underline{\mathcal{G}}$ for which the precise analysis can be performed.

The fast versions also deal with other functions, which are called the Adjust functions. Such functions perform

few steps of the (plain) Euclid Algorithm. However, the bit-complexity of the Adjust functions depends on the size of the quotients which are computed during these steps. Even for estimating the worst–case complexity of the fast variants, the Adjust functions are not precisely analyzed. The usual argument is "The size of a quotient is $O(1)$". Of course, this assertion is false, since this is only true on average. Since the Adjust functions are related to some precise steps, the size of the quotients computed at these precise steps may be large. We are then led to study the probability that the Euclid Algorithm only produces "small" quotients. This is done in Theorem 4. And of course, we also need this result for our truncated data. This is the goal of Theorem 5.

Finally, we obtain the exact average-case complexity of our versions of the two main algorithms of interest, the $\underline{\mathcal{HG}}$ algorithm, and the $\underline{\mathcal{G}}$ algorithm itself. We prove the following results [Theorems 6 and 7] about the average bit-complexity B, G of both algorithms, on the set of random inputs of size n

$$\mathbb{E}_n[B] = \Theta(n(\log n)^2 \log\log n),$$
$$\mathbb{E}_n[G] = \Theta(n(\log n)^2 \log\log n).$$

Furthermore, we obtain some precise information about the constants which are involved in the Θ–terms. Then, our proven average bit–complexity of the $\underline{\mathcal{HG}}, \underline{\mathcal{G}}$ algorithms is of the same order as the usual heuristic bound on the worst-case complexity of $\mathcal{HG}, \mathcal{G}$ algorithms.

1.3 Methods. All our main conclusions obtained here are "expected", and certainly do not surprise the reader. However, the irruption of the density ψ is not expected, and an actual proof of this phenomenon is not straightforward. This is due to the fact that there is a correlation between successive steps of the Euclid Algorithm. Then, the tools which are usual in analysis of algorithms [13], as generating functions, are not well-adapted to study this algorithm. All the analyses which will be described here are instances of the so–called dynamical analysis, where one proceeds in three main steps: First, the (discrete) algorithm is extended into a continuous process, which can be defined in terms of the dynamical system related to the Gauss map. Then, the transfer operator $\mathbf{H}_s$ defined as

$$\mathbf{H}_s[f](x) := \sum_{m\geq 1} \frac{1}{(m+x)^{2s}} f\left(\frac{1}{m+x}\right)$$

explains how the distribution evolves, but only in the continuous world. The executions of the gcd algorithm are now described by particular trajectories (i.e., trajectories of "rational" points), and a transfer "from continuous to discrete" must finally be performed.

The present paper mainly uses two previous works, and can be viewed as an extension of them: first, the average-case analysis of the Lehmer-Euclid algorithm performed in [9], second the distributional methods described in [2, 21]. First, we again use the general framework that Daireaux and Vallée have defined for the analysis of the Lehmer-Euclid Algorithm, which explains how the Lehmer-Euclid algorithm can be viewed as a sequence of Interrupted Euclidean algorithms $\mathcal{E}_{[\delta,\delta+\gamma]}$. However, in [9], we only used some "easy" properties of the transfer operator $\mathbf{H}_s$. We now need proving that properties which were already crucial in previous distributional analysis [2, 1, 21] –namely, the *US* Property for the quasi-inverse $(I-\mathbf{H}_s)^{-1}$ of the transfer operator– also hold in our context. The *US* property can be summarized as follows:
For any ξ, there exists a vertical strip $\mathcal{S}$ of the form $|\Re s - 1| \leq \sigma$ for which the following holds:

(*i*) The quasi-inverse has a unique pôle in $\mathcal{S}$,

(*ii*) On the left line $\Re s = 1-\sigma$, one has $(I-\mathbf{H}_s)^{-1}[1] = O(|\Im s|^{\xi})$.

Here, our various theorems lead to study parameters of various type, whose generating functions involve various operators $\mathbb{G}_{s,w}$ which depend on two variables s, w. However, for small w's, all these operators can be viewed as a perturbation of the quasi-inverse $(I-\mathbf{H}_s)^{-1}$ and we have to prove that the *US* Property extends to these perturbated quasi-inverses. In particular, the existence of a strip $\mathcal{S}$ where the *US* property holds uniformly with respect to w is crucial in the analysis: the main choices of the parameters in our versions $\underline{\mathcal{HG}}$ and $\underline{\mathcal{G}}$ of the fast algorithms depend on the width σ of this strip (see Theorem 3).

Plan and notations. Section 2 describes the main algorithms $\mathcal{HG}$ and $\mathcal{G}$. Section 3 is devoted to explain the main steps to be done for obtaining a proven analysis. We then state our main results of general interest. In Section 4, we describe the versions $\underline{\mathcal{HG}}$ and $\underline{\mathcal{G}}$ to be analyzed, and we prove the two main results about their average bit-complexity. The last Section is devoted to the proof of the main results of Section 3.
We denote the logarithm in base 2 by $\lg x$, and $\ell(x)$ denotes the binary size of integer x, namely $\ell(x) := \lfloor \lg x \rfloor + 1$.

2 Fast and Interrupted Euclidean algorithms

We present in this section the main algorithms studied in this paper. We first describe the general structure of

the Knuth-Schönhage algorithm. We explain how the $\mathcal{HG}$ algorithm can be seen as a sequence of interrupted Euclidean algorithms, where the sequence of divisions is stopped as soon as the integers have lost a fraction of their number of bits.

2.1 Euclid's algorithm. Let (A_1, A_0) be a pair of positive integers with $A_1 \leq A_0$. On input (A_1, A_0), the Euclid algorithm computes the remainder sequence (A_k) with a succession of divisions of the form

$$A_k = Q_{k+1}A_{k+1} + A_{k+2} \quad \text{with} \quad Q_{k+1} = \left\lfloor \frac{A_k}{A_{k+1}} \right\rfloor, \tag{2.3}$$

and stops when $A_{p+1} = 0$. The integer Q_k is the k–th quotient and the successive divisions are written as

$$\mathcal{A}_k = \mathcal{Q}_{k+1}\mathcal{A}_{k+1},$$

$$\text{with} \quad \mathcal{A}_k := \begin{pmatrix} A_{k+1} \\ A_k \end{pmatrix} \quad \text{and} \quad \mathcal{Q}_k := \begin{pmatrix} 0 & 1 \\ 1 & Q_k \end{pmatrix},$$

so that

$$\mathcal{A}_0 = \mathcal{M}_{(i)}\mathcal{A}_i \quad \text{with} \quad \mathcal{M}_{(i)} := \mathcal{Q}_1\mathcal{Q}_2\cdots\mathcal{Q}_i. \tag{2.4}$$

In the following, we consider a "slice" of the plain Euclidean Algorithm $\mathcal{E}$, between index i and index j, namely the interrupted algorithm $\mathcal{E}_{(i,j)}$ which begins with the pair $\mathcal{A}_i$ as its input and computes the sequence of divisions (2.3) with $i \leq k \leq j-1$. Its output is the pair $\mathcal{A}_j$ together with the matrix

$$\mathcal{M}_{(i,j)} = \prod_{k=i+1}^{j} \mathcal{Q}_k, \qquad \mathcal{M}_{(1,i)} = \mathcal{M}_{(i)}, \tag{2.5}$$

with matrix $\mathcal{M}_{(i)}$ defined in (2.4). We define the size of a matrix $\mathcal{M}$ as the maximum of the binary sizes of its coefficients. The size $\ell_{(i,j)}$ of the matrix $\mathcal{M}_{(i,j)}$ satisfies

$$\ell_{(i,j)} \leq 2(j-i) + \sum_{k=i+1}^{j} \ell(Q_k) \tag{2.6}$$

The (naive) bit-complexity $C_{(i,j)}$ of the algorithm $\mathcal{E}_{(i,j)}$,

$$C_{(i,j)} := \sum_{k=i+1}^{j} \ell(A_k)\cdot\ell(Q_k)$$

satisfies

$$C_{(i,j)} \leq \ell(A_{i+1}) \cdot \sum_{k=i+1}^{j} \ell(Q_k). \tag{2.7}$$

The Lehmer Algorithm [20, 18] replaces large divisions by large multiplications and small divisions. The fast algorithm applies recursively the principles of Lehmer, and using fast FFT multiplications of complexity $\Theta(\mu(n))$ (with $\mu(n) = n\log n\log\log n$) replaces the costly computation of the remainder sequence A_i (which requires $O(n^2)$ bit operations), by a sequence of matrix products: it divides the total Euclidean Algorithm into interrupted Euclidean algorithms, of the form $\mathcal{E}_{(i,j)}$ and computes matrices of the form $\mathcal{M}_{(i,j)}$, defined in (2.5). The recursion, based on Divide and Conquer techniques, is stopped when the integers are small enough, and, at this moment, the algorithm uses small divisions. One finally obtains a subquadratic gcd algorithm.

2.2 How to replace large divisions by small divisions? Lehmer remarked that, when two pairs (A, B) and (a, b) are sufficiently close (i.e., the rationals A/B and a/b are close enough), the Euclid algorithm on (A, B) or (a, b) produces (at least at the beginning) the same quotient sequence (Q_i). This is why the following definition is introduced:

Definition. *Consider a pair (A, B) with $A \leq B$ and an integer b of length $\ell(b) \leq \ell(B)$. We denote by $\pi_{[b]}(A, B)$ any pair (a, b) which satisfies*

$$\left|\frac{A}{B} - \frac{a}{b}\right| \leq \frac{1}{b}.$$

And the criterion (due to Lehmer and made precise by Jebelean) is:

Lemma 1. [Lehmer, Jebelean] *For a pair (A, B) with $A \leq B$ and $n := \ell(B)$, consider, for $m \leq n$, the small pair $(a, b) = \pi_{[b]}(A, B)$ of length $\ell(b) = m$, and the sequence of the remainders (a_i) of the Euclid Algorithm on the small input (a, b). Denote by k the first integer k for which a_k satisfies $\ell(a_k) \leq \lceil m/2 \rceil$. Then the sequence of the quotients q_i of the Euclid Algorithm on the small input (a, b) coincide with the sequence of the quotients Q_i of the Euclid Algorithm on the large input(A, B) for $i \leq k - 3$.*

Usually, this criterion is used with a particular pair $\pi_{[b]}(A, B)$ where the integer b is obtained by the m-truncation of B, i.e., the suppression of its $(n - m)$ least significant bits. Then a is easy to compute since it may be chosen itself as the m-truncation of A. In this case, the $\pi_{[b]}$ function corresponds to truncation of both A and B and is denoted by $T_m(A, B)$. However, the Jebelean criterion holds for any choice of $(a, b) = \pi_{[b]}(A, B)$, even if the integer a is less easy to compute in the general case: the integer a can be chosen as the integer part of the rational $(Ab)/B$, and its computation needs a product and a division.

2.3 Interrupted Algorithms. In Jebelean's property (Lemma 1), the Euclid Algorithm on the small pair (a, b) of binary size m is stopped as soon the remainder a_k has lost $\lceil m/2 \rceil$ bits. This is a particular case of the so–called Interrupted Euclidean Algorithm of parameter δ (with $0 < \delta < 1$), which stops as soon as the current remainder has lost $\lfloor \delta m \rfloor$ bits (with respect to the input which has m bits). This (general) interrupted Algorithm denoted by $\mathcal{E}_\delta$, and described in Figure 2, is defined as follows: On the input (A, B) of Ω_n, this algorithm begins at the beginning of the Euclid Algorithm, and stops as soon as the remainder A_i has lost $\delta\, n$ bits (with respect to the input B). Then, with the notations defined in Section 2.1, one has $\mathcal{E}_\delta = \mathcal{E}_{(1,P_\delta)}$, with

$$P_\delta := \min\big\{k;\ \ell(A_k) \le \lfloor (1-\delta)n \rfloor\big\}. \tag{2.8}$$

Figure 2 also describes the $\widehat{\mathcal{E}}_\delta$ Algorithm, which is just a slight modification of the $\mathcal{E}_\delta$ Algorithm, where the last two steps are suppressed (in view of applications of Lemma 1), and $\widehat{P}_\delta$ denotes the variable $P_\delta - 2$. Then, P_δ, and $\widehat{P}_\delta$ are just the number of iterations of the $\mathcal{E}_\delta, \widehat{\mathcal{E}}_\delta$ algorithms and $P_1 = P$ is just the number of iterations of the Euclid Algorithm.

In the following, it will be convenient to consider more general interrupted algorithms, of the form $\mathcal{E}_{[\delta,\delta+\gamma]}$. The Algorithm $\mathcal{E}_{[\delta,\delta+\gamma]}$ is defined as follows: On the input (A, B) of Ω_n, this algorithm begins at the P_δ-th iteration of the Euclid Algorithm, as soon as the remainder A_k has lost $\lfloor \delta\, n \rfloor$ bits (with respect to the input B) and stops when the remainder A_i has lost $\lfloor \gamma\, n \rfloor$ supplementary bits (with respect to the input B). Then, $\mathcal{E}_{[0,\delta]} = \mathcal{E}_\delta = \mathcal{E}_{(0,P_\delta)}$ and $\mathcal{E}_{[\delta,\gamma+\delta]} = \mathcal{E}_{(P_\delta,P_{\delta+\gamma})}$, where P_δ is defined in (2.8). Of course, we can also design the variants with a hat, where the last two steps are suppressed.

2.4 Description of the $\mathcal{HG}$ Algorithm. The general principles. This is the $\widehat{\mathcal{E}}_{1/2}$ algorithm which is used in Jebelean's Lemma. This lemma is a main tool to compute (in a recursive way) a function $\mathcal{HG}$ [for Half-gcd]. On an input (A, B) of binary size n, this function returns exactly the same result as $\widehat{\mathcal{E}}_{1/2}$, but runs faster. With the algorithm $\mathcal{HG}$, it is possible to design a fast algorithm denoted $\mathcal{G}$ which computes the gcd itself. Let us explain the main principles of the $\mathcal{HG}$ algorithm.

Suppose that the Euclid Algorithm, on an input (A, B) of length n, has already performed P_δ iterations. Now, the current pair, denoted by (A', B') has a binary size close to $(1-\delta)n$. We may use the Jebelean Property to continue. Then, we choose a length m for truncating, an integer b of length m, and consider the small pair $(a, b) = \pi_{[b]}(A', B')$. The $\mathcal{HG}$ algorithm on this pair

```
Algorithm 𝓔_δ(A, B)
  n := ℓ(B)
  i := 1
  A_1 := A, A_0 := B
  M_0 := I
  While ℓ(A_i) > (1 − δ) · n
    Q_i := ⌊A_{i−1}/A_i⌋
    A_{i+1} := A_{i−1} − Q_i A_i
    M_i := M_{i−1} · Q_i
    i := i + 1
  Return (A_{i−1}, A_i, M_{i−1})
```

```
Algorithm 𝓔̂_δ(A, B)
  n := ℓ(B)
  i := 1
  A_1 := A, A_0 := B
  M_0 := I
  While ℓ(A_i) > (1 − δ) · n
    Q_i := ⌊A_{i−1}/A_i⌋
    A_{i+1} := A_{i−1} − Q_i A_i
    M_i := M_{i−1} · Q_i
    i := i + 1
  Return (A_{i−3}, A_{i−2}, M_{i−3})
```

Figure 2: The $\mathcal{E}_\delta$ Algorithm, and the $\widehat{\mathcal{E}}_\delta$ algorithm, which is a slight modification of the $\mathcal{E}_\delta$ Algorithm.

(a, b) will produce a matrix $\mathcal{M}$ which is a matrix which would have been produced by the Euclid algorithm on the pair (A', B'). Then, the pair (C, D) computed as $\binom{C}{D} = \mathcal{M}^{-1}\binom{A'}{B'}$ is a remainder pair of the Euclid algorithm on the input (A, B). The size of the matrix $\mathcal{M}$ is approximately $m/2$, but smaller than $m/2$ (due to the two backward steps of Lemma 1), and thus of the form $(m/2) - r(a, b)$, where $r(a, b)$ is the number of bits which are "lost" for the matrix $\mathcal{M}$ during the backward steps. Then, with (2.6),

$$r(a, b) \le 2 \max\{\ell(q_i) + 1;\ 1 \le i \le p(a, b)\},$$

where q_i are the quotients that occur in $\mathcal{E}(a, b)$, and $p(a, b)$ the number of steps of $\mathcal{E}(a, b)$. If the truncature length m is chosen as a linear function of the input size n, of the form $m = 2\gamma n$, then the size of the pair (C, D) is approximately equal to $[1-\delta-\gamma]n$, but slightly larger. If we wish to obtain a remainder pair (C', D') of length $[1 - \delta - \gamma]n$, we have to perform, from the pair (C, D) a certain number of steps of the Euclid Algorithm, in order to cancel the loss due to the backward steps. This is the goal of the Adjust function, whose cost will be $\approx (1 - \delta)n \cdot r(a, b)$ [see (2.7)].

Finally, we have designed an algorithm which produces the same result as the interrupted algorithm $\mathcal{E}_{[\delta,\delta+\gamma]}$,

and can be described as follows:

(i) it truncates the input (A', B'), with a truncation length $m = 2\gamma n$, and obtains a small pair (a, b) of length m.

(ii) it performs the $\mathcal{HG}$ algorithm on the pair (a, b): this produces a pair (c, d) and a matrix $\mathcal{M}$.

(iii) it performs the product $\binom{C}{D} := \mathcal{M}^{-1} \binom{A'}{B'}$.

(iv) It uses the Adjust function, which performs some steps of the Euclid Algorithm from the pair (C, D) and stops as soon the current remainder pair (C', D') has a size equal to $\lfloor (1 - \delta - \gamma)n \rfloor$.

2.5 The usual designs for the $\mathcal{HG}$ and $\mathcal{G}$ algorithms. How to use this idea for computing (in a recursive way) the $\mathcal{HG}$ Algorithm? The usual choice for γ is $\gamma = 1/4$, more precisely $m = \lceil n/2 \rceil$. Then, the previous description provides a method to obtain $\mathcal{E}_{[0,1/4]}$ (with a first choice $\delta = 0$), then $\widehat{\mathcal{E}}_{[1/4,1/2]}$ (with a second choice $\delta = 1/4$). Since $\widehat{\mathcal{E}}_{[0,1/2]} = \mathcal{E}_{[0,1/4]} \cdot \widehat{\mathcal{E}}_{[1/4,1/2]}$, we are done. Remark that using the "hat" algorithm in the second step leads to modifying the Adjust function for this step. We use (for this second step) a so–called "hat Adjust" function, which may also perform some backward steps in the Euclid Algorithm on the large inputs.

The general structure of the algorithm $\mathcal{HG}$ is described in Figure 3. The recursion is stopped when the naive algorithm $\widehat{\mathcal{E}}_{1/2}$ becomes competitive. This defines a threshold for the binary size denoted by S (remark that $S = S(n)$ is a function of the input size n).

With this $\mathcal{HG}$ algorithm, we can obtain an algorithm named $\mathcal{G}$ which computes the gcd. The idea for designing such an algorithm is to decompose the total Euclid Algorithm into interrupted algorithms, as

$$\mathcal{E}_{[0,1]} = \mathcal{E}_{[0,1/2]} \cdot \mathcal{E}_{[1/2,3/4]} \cdot \ldots \cdot \mathcal{E}_{[1-(1/2)^k, 1-(1/2)^{k+1}]} \cdot \ldots$$

Then, the $\mathcal{HG}$ algorithm, when running on inputs of size $n/(2^k)$ produced by the $\mathcal{E}_{[0,1-(1/2)^k]}$ algorithm can easily simulate the $\mathcal{E}_{[1-(1/2)^k, 1-(1/2)^{k+1}]}$ algorithm.

This decomposition also stops when the naive algorithm `gcd` becomes competitive. This defines a threshold for the length denoted by T (remark that $T = T(n)$ is also a function of the input size n).

We now consider the $\mathcal{HG}$ Algorithm, where all the products use a FFT multiplication of order

$$\mu(n) = \Theta(n \log n \log\log n).$$

In this case, we choose the recursion depth H so that the main cost will be the "internal" cost, of order

```
Algorithm HG(A, B)
    n := ℓ(B)
    S := log² n
    Return HG(A, B, S)
```

```
Algorithm HG(A, B, S)
1   n := ℓ(B)
2   If n ≤ S then return Ê_{1/2}(A, B)
3   M := I
4   m := ⌊n/2⌋;
5   For i := 1 to 2 do
6     (a_i, b_i) := T_m(A, B)
7     (c_i, d_i, M_i) := HG(a_i, b_i, S)
8     (C_i, D_i)ᵀ := M_i^{-1} (A, B)ᵀ
9     Adjust_i(C_i, D_i, M_i)
10    (A, B) := (C_i, D_i)
11    M := M · M_i
12  Return (A, B, M)
```

```
Algorithm G(A, B)
1   n := ℓ(B)
2   T := √n log n
3   While ℓ(A) ≥ T do
4     (A, B, M_1) := HG(A, B)
5  Return gcd(A, B)
```

Figure 3: General structure of the classical algorithms $\mathcal{HG}$ and $\mathcal{G}$.

$\Theta(\mu(n)) \log n$, since the cost due to the leaves (where the naive $\widehat{\mathcal{E}}_{1/2}$ is performed) will be of asymptotic smaller order. Then, H satisfies the relation

$$2^H \cdot \left(\frac{n}{2^H}\right)^2 \approx_{\leq} \mu(n) \log n,$$

$$\text{so that} \quad \frac{n}{2^H} \approx_{\leq} S(n) = \log^2 n, \quad H \approx_{\geq} \lg n - 2 \lg\lg n.$$

This is the "classical" version of the Knuth–Schönhage algorithm. Clearly, the cost of this algorithm comes from three types of operations:

(i) the two recursive calls of line 7;

(ii) the products done at lines 8 and 11: with a clever implementation, it is possible to use in line 8 the pair (c, d) just computed in line 7. If all the matrices and integer pairs have –on average– the expected size, the total expected cost due to the products is $[12 + 8 + 8]\ \mu(n/4) = 28\ \mu(n/4)$;

(iii) the two functions `Adjust` performed at line 9, whose total average cost is $R(n)$.

We consider as the set of possible inputs of the $\mathcal{HG}$ algorithm the set

$$\Omega_n := \{(u, v); \quad 0 \leq u \leq v, \quad \ell(v) = n\}$$

endowed with some probability $\mathbb{P}_n$. We denote by $B(n)$ the average number of bit operations performed by the algorithm $\mathcal{HG}$ on Ω_n. Since each of the two recursive calls is made on data with size $n/2$, it can be "expected" that $B(n)$ asymptotically satisfies

$$B(n) \approx 2B(\frac{n}{2}) + 28\ \mu(\frac{n}{4}) + R(n) \qquad \text{for } n > S. \tag{2.9}$$

Moreover, the average cost $R(n)$ can be "expected" to be negligible with respect to the multiplication cost $\mu(n)$. If the FFT multiplication is used, with $\mu(n) = \Theta(n \log n \log\log n)$, the total average bit–cost is "expected" to be

$$B(n) \approx \Theta(\mu(n) \log n) = \Theta(n(\log n)^2 \log\log n).$$

With this (heuristic) analysis of the $\mathcal{HG}$ algorithm, it is easy to obtain the (heuristic) average bit–complexity of the $\mathcal{G}$ algorithm which makes a recursive use of the $\mathcal{HG}$ algorithm and stops as soon as the naive algorithm becomes competitive. It then stops at a recursion depth M, when

$$\left(\frac{n}{2^M}\right)^2 \approx_{\leq} \mu(n) \log n,$$

so that

$$\frac{n}{2^M} \approx_{\leq} T(n) = \sqrt{n} \log n, \quad M \approx_{\geq} \frac{1}{2} \lg n - \lg\lg n.$$

The average bit–cost $G(n)$ of the $\mathcal{G}$ algorithm on data of size n satisfies

$$G(n) \approx \sum_{i=0}^{M-1} B(\frac{n}{2^i}) \quad \text{ so that } \quad G(n) \approx \Theta(B(n)).$$

3 The main steps towards a proven analysis.

The analysis is based on the Divide and Conquer equation (2.9), which is not a "true" equality. It is not clear why a "true" equality should hold, since each of the two recursive calls is done on data which do not possess the same distribution as the input data. And, of course, the same problem will be asked at each depth of the recursion. We have to make precise *the evolution of the distribution during the Euclid Algorithm, but also the distribution of the truncated data.*

Moreover, we have also to make precise the average bit–complexity $R(n)$ of the Adjust function. Even for estimating the worst–case complexity of the fast variants, the Adjust functions are not precisely analyzed. The usual argument is "The size of a quotient is $O(1)$". Of course, this assertion is false, since this is only true on average. Since the Adjust functions are related to some precise steps, the size of the quotient computed at these precise steps may be large. We are then led to study the probability that the Euclid Algorithm only produces "small" quotients.

3.1 Evolution of densities. Consider a density f on the unit interval $= [0, 1]$. Any set Ω_n is seen as a set of rationals

$$\Omega_n := \{x \in \mathbb{Q} \cap]0, 1],\ x = u/v,\ \ell(v) = n\}$$

and is endowed with the restriction of f to Ω_n: for any $x_0 \in \Omega_n$,

$$\mathbb{P}_{n,f}(x_0) := \frac{1}{\sum_{x \in \Omega_n} f(x)} f(x_0).$$

The evolution of the density during the execution of the Euclid Algorithm is of crucial interest. We begin with all the sets Ω_n endowed with the probability $\mathbb{P}_{n,f}$.

For $x \in \Omega$, we recall that $P_\delta(x)$ is the smallest integer k where the binary size of x_k is less than $(1 - \delta)\ell(x)$. We are interested in describing the density of the rational $x_{\langle\delta\rangle}$ defined as

$$x_{\langle\delta\rangle} := x_k \qquad \text{when} \quad P_\delta(x) = k.$$

This rational is the input for all interrupted algorithms with a beginning parameter δ. It is then essential to study the random variable P_δ. Since the rational x loses $\ell(x)$ bits during $P(x)$ iterations, it can be expected that it loses $\delta\ell(x)$ bits during $\delta P(x)$ iterations, which would imply that $P_\delta(x)$ is sufficiently close to $\delta P(x)$. This is what we call the regularity of the algorithm. With techniques close to the renewal methods, we prove a quasi-powers expression for the moment generating function of P_δ, from which we deduce two results. The first one is interesting *per se*, and provides an extension of the result of Baladi-Vallée [2], which exhibits an asymptotic gaussian law for $P := P_1$.

Theorem 1. *Consider the set Ω_n endowed with a probability $\mathbb{P}_{n,f}$ relative to a strictly positive function f of class $\mathcal{C}^1$. Then, for any $\delta \in]0, 1]$, the random variable P_δ is asymptotically gaussian on Ω_n [with a speed of convergence of order $O(1/\sqrt{n})$]. Moreover,*

$$\mathbb{E}_{n,\psi}[P_\delta] \sim \frac{2 \log 2}{|\Lambda'(1)|} \cdot \delta n,$$

$$\mathbb{V}_{n,f}[P_\delta] \sim (2 \log 2)^2 \left|\frac{\Lambda''(1)}{\Lambda'(1)^3}\right| \cdot \delta n.$$

Theorem 2. *There exist n_0 and a constant $\sigma > 0$, such that, for any $n \geq n_0$, for any real δ with $0 < \delta < 1$, for any fonction $\varepsilon : [0,1] \to [0,1]$, such that $\varepsilon(x) \leq x$, for any interval $J \subset I$, for any strictly positive density f of class $\mathcal{C}^1$, the probability that the rational $x_{\langle\delta\rangle}$ computed by the Euclid Algorithm belongs to the interval J satisfies*

$$\mathbb{P}_{n,f}[x_{\langle\delta\rangle} \in J] = \left(\int_J \psi(t)dt\right) \cdot [1 + \beta'_n(\delta, J, \varepsilon)]\,.$$

Here, the density ψ is described in (1.2), the real σ is the width of the US *strip and*

$$\beta'_n(\delta, J, \varepsilon) = O\left(\frac{\varepsilon(|J|)}{|J|}\right) + O\Big(\frac{1}{\varepsilon(|J|)|J|}\Big)^{\frac{1}{2}} 2^{-\sigma(1-\delta)n} + O\big(2^{\frac{\sigma}{2}\delta n}\big).$$

The constants in the O-term only depend on the function f.

3.2 Probabilistic truncations. Finally, we are also interested by the distribution of the truncated rationals. We recall that the truncated rationals classically used are obtained with truncations of numerator A and denominator B of rational A/B. It is not clear how to reach the distribution of such truncated rationals. This is why we define a probabilistic truncation, which leads to more regular distributions, and also allows us to apply Jebelean's Property (Lemma 1).

For $x = A/B \in \Omega_n$, and $m \leq n$, we define $\pi_m(x)$ as follows:

(1) Choose a denominator b in the set of integers of binary size m, with a probability proportional to b. More precisely, we choose a denominator b according to the law

$$\mathbb{P}_m[b = b_0] = \frac{1}{\theta_m} \cdot b_0 \qquad \text{with} \quad \theta_m = \sum_{b;\ell(b)=m} b.$$

(2) Compute the integer a which is the integer part of $x \cdot b$. This computation involves the product $A \cdot b$ then the division of the integer $A \cdot b$ by B. This can be done in $O(\mu(n))$ with a O–constant larger than the constant of the multiplication. Of course, this does not give rise to a practical algorithm. However, we will see that using this probabilistic truncation does not change the order of the average complexity of the $\mathcal{HG}$ algorithm.

(3) Define $\pi_m(A/B)$ as the rational a/b, and remark that the set $\pi_m^{-1}(a/b)$ is the interval

$$J\big(\frac{a}{b}\big) := \big[\frac{a}{b},\ \frac{a}{b}+\frac{1}{b}\big[, \quad \text{with} \quad \Big|J\big(\frac{a}{b}\big)\Big| = \frac{1}{b} = \Theta(2^{-m}).$$

This is sufficient for applying Jebelean's criterion (Lemma 1).

We start with a strictly positive density f of class $\mathcal{C}^1$ on $[0,1]$. The function $g_m = g_m(f)$ defined on Ω_m as

$$g_m(f)(y_0) = \frac{1}{|J(y_0)|}\int_{J(y_0)} f(t)dt$$

satisfies $\mathbb{P}_{n,f}[x; \pi_m(x) = y] = \mathbb{P}_{m,g_m(f)}[y]$. Furthermore, the function $g_m(f)$ is a smoothed version of the initial function f, and

$$g_m(f)(x) = f(x) + O(|J| \cdot \|f\|_1)$$

$$\text{so that} \qquad \frac{\mathbb{P}_{m,g_m(f)}}{\mathbb{P}_{m,f}} = 1 + O(2^{-m}).$$

Since f is a density on $[0,1]$, the cumulative sum of $g_m(x)$ on Ω_m satisfies

$$\sum_{x\in\Omega_m} g_m(f)(x) = \theta_m.$$

This allows a comparison between two probabilities:

Lemma 2. *Consider a strictly positive density f of class $\mathcal{C}^1$ on I. For any n, for any $m \leq n$, for any $y \in \Omega_m$, one has*

$$\mathbb{P}_{n,f}[x; \pi_m(x) = y] = \mathbb{P}_{m,f}[y] \cdot [1 + O(2^{-m})],$$

where the constant in the O-term only depends on f.

3.3 Truncations and evolution of densities. With Theorem 2, the previous comparison of densities done in Lemma 2, and the optimal choice of $\varepsilon(x) = x^2$, we obtain the following result which will be a central tool in our analysis.

Theorem 3. *Consider the constant σ and the density ψ of Theorem 2. There exists n_0 such that, for any $n \geq n_0$, for any real δ with $0 < \delta < 1$, for any pair of strictly positive constants σ_1, σ_2 with $\sigma_1 < (2/3)\sigma$, for a truncation m which satisfies*

$$\sigma_2\,\delta n < m < \sigma_1\,(1-\delta)n,$$

the distribution of the m–truncation of the rational $x_{\langle\delta\rangle}$ computed by the Euclid Algorithm satisfies

$$\mathbb{P}_{n,\psi}[x;\ \pi_m(x_{\langle\delta\rangle}) = y_0] = \mathbb{P}_{m,\psi}[y_0] \cdot [1 + O(\beta_n)]\,,$$

$$\textit{with} \quad \beta_n = 2^{-\sigma_3(1-\delta)n} + 2^{-\sigma_4\delta n},$$

$$\textit{and} \quad \sigma_3 = \sigma - (3/2)\sigma_1, \quad \sigma_4 := \inf(\sigma/2, \sigma_2).$$

Remark that Lemma 2 is designed to deal with the case $\delta = 0$.

3.4 Analysis of the Adjust function: the rôle of small quotients. In the design of fast versions of the Euclid Algorithm, the rôle of the Adjust function is important. This is due to the fact that Jebelean's Property does not produce rationals of the exact size. We return to the notations of Section 2.4. Consider a (large) input (A, B) of size n. During the two backward steps that are necessary for the Euclid algorithm on the small input (a, b), the matrix $\mathcal{M}$ gets smaller, so that the rational (C, D) has a larger size. It is then necessary to perform some steps of the Euclid algorithm on (C, D), and the cost of these steps [see Section 2.4] is less than

$$2n \cdot \max\{\ell(q_i) + 1;\ 1 \leq i \leq p(a, b)\},$$

where q_i are the quotients that occur in $\mathcal{E}(a, b)$, and $p(a, b)$ the number of steps of $\mathcal{E}(a, b)$. If we wish this cost to be "small", it is then crucial to study the probability that the Euclid Algorithm produces small quotients, and the set of interest gathers the rationals x for which all the quotients $q_i(x)$ which occur in the Continued Fraction Expansion of x are "small". Of course, it is natural to relate the size of the quotients to the size of the rational itself, and we are led to introduce the subset $\mathcal{U}_n$ of Ω_n defined as

$$\mathcal{U}_n := \{x \in \Omega_n; \quad \ell(q_i(x)) \leq \log n \cdot (\log\log n)^{1/2}, \tag{3.10}$$
$$\text{for } 1 \leq i \leq p(x)\}.$$

Remark that, by construction, when $x = a/b$ belongs to $\mathcal{U}_m$, the Adjust function for the large input (A, B) of size n will be less than $n \log m (\log\log m)^{1/2}$: it will be negligible with respect to $\mu(n)$. More generally, for any sequence $n \mapsto M(n)$, we may define

$$\mathcal{O}_n^{[<M]} := \{x \in \Omega_n; \quad q_i < M(n), \quad \text{for } 1 \leq i \leq p(x)\}.$$

The following result provides a precise estimate of the probability that the Euclid Algorithm on inputs of size n, always produces quotients smaller than $M(n)$.

Theorem 4. [7] *There exist n_0 and a real $\sigma > 0$ such that, for any strictly positive density f of class $\mathcal{C}^1$, for any $n \geq n_0$, the probability of the set $\mathcal{O}_n^{[<M]}$ satisfies*

$$\mathbb{P}_{n,f}[\mathcal{O}_n^{[<M]}] = \exp\left(-\frac{n}{M(n)}\right) \cdot \left[1 + O\left(\frac{1}{M(n)}\right)\right] + O(2^{-\sigma n}).$$

In particular, when $\ell(M(n)) = \log n \cdot (\log\log n)^{1/2}$, the probability of the set $\mathcal{U}_n$ satisfies

$$\mathbb{P}_{n,f}[\mathcal{U}_n] = 1 - O(n^{1-(\log\log n)^{1/2}}).$$

We also need the same type of result for our variables $\pi_m(x_{\langle\delta\rangle})$.

Theorem 5. *Consider the general framework of Theorem 3. The probability that the rational $\pi_m(x_{\langle\delta\rangle})$ belongs to the set $\mathcal{U}_m$ of small quotients satisfies*

$$\mathbb{P}_{n,\psi}[x; \pi_m(x_{\langle\delta\rangle}) \in \mathcal{U}_m] = 1 - O(m^{1-(\log\log m)^{1/2}}).$$

Due to Lemma 2, this is also true for the case $\delta = 0$.

4 The algorithms to be analyzed.

There are two main differences between the usual $\mathcal{HG}$ and $\mathcal{G}$ Algorithm and our versions to be analyzed which are denoted as $\underline{\mathcal{HG}}$ and $\underline{\mathcal{G}}$. See Figure 4.

(*i*) the truncatures T_m defined in Section 2.2 are replaced by probabilistic truncatures π_m, as defined in Section 3.3.

(*ii*) For the $\underline{\mathcal{HG}}$ algorithm, the number of recursive calls L and the degree of truncatures (i.e., the ratio m/n) are not the same as in the $\mathcal{HG}$ Algorithm. The algorithm $\underline{\mathcal{HG}}$ is also built as a Divide and Conquer Algorithm; however, the relations between the truncation m and the parameter δ, crucial for applying Theorems 3 and 5, lead to a recursive algorithm $\underline{\mathcal{HG}}$ with L recursive calls, where L depends on the width of the *US* strip and can be greater than 2.

(*iii*) The study is done when the initial density equals ψ. This choice makes easier the study of various recursions. The constants which appear in Theorems 6 and 7 are relative to this particular case. Since any another strictly positive density f satisfies

$$\frac{\min f}{\max \psi} \leq \frac{\mathbb{E}_{n,f}[C]}{\mathbb{E}_{n,\psi}[C]} \leq \frac{\max f}{\min \psi},$$

Theorems 6 and 7 hold with any strictly positive density, with other constants, which depend on f.

As before, the recursive calls in the $\underline{\mathcal{HG}}$ Algorithm are stopped when the naive $\widehat{\mathcal{E}}_{1/2}$ Algorithm becomes competitive. The calls of the $\underline{\mathcal{G}}$ Algorithm to the $\underline{\mathcal{HG}}$ algorithm are stopped when the naive gcd algorithm becomes competitive.
We now describe more precisely the point (*ii*).

4.1 The first recursive call. Inside the first recursive call of $\underline{\mathcal{G}}$ to $\underline{\mathcal{HG}}$, the parameter δ belongs to $[0, 1/2]$, and the possible values of $(1 - \delta)$ vary between 1 and

```
Algorithm HG(A, B)
1   n := ℓ(B)
2   S := log² n
3   Return HG(A, B, S)
```

```
Algorithm HG(A, B, S)
1   n := ℓ(B)
2   If n ≤ S then return Ê_{1/2}(A, B)
3   M := I
4   m := ⌊n/L⌋;
5   For i := 1 to L do
6     (a, b) := π_m(A, B)
7     (c_i, d_i, M_i) := HG(a, b, S)
8     (C_i; D_i) := M_i^{-1} (A; B)
9     Adjust_i (C_i, D_i, M_i)
10    (A, B) := (C_i, D_i)
11    M := M · M_i
12  Return (A, B, M)
```

```
Algorithm G(A, B)
1   n := ℓ(B)
2   T := √n log n
3   While  n ≥ T do
4     (A, B, M) := HG(A, B)
5  Return gcd(A, B)
```

Figure 4: General structure of the algorithms $\mathcal{HG}$ and $\mathcal{G}$ to be analyzed. The integer $L \geq 2$ defined as $L := \max(2, \lceil 4/\sigma \rceil)$ depends, via the strip width σ, on the *US* property of the Euclidean Algorithm.

1/2. The truncation m is chosen of the form $m = n/L$ with an integer $L \geq 2$. Then there are L recursive calls of $\underline{\mathcal{HG}}$ to himself, for values of δ_i of the form

$$\delta_i = \frac{i}{2L}, \quad \text{with} \quad 0 \leq i \leq L-1. \tag{4.11}$$

For $i \geq 1$, the following constraints

$$\frac{2}{L}\delta n \leq m = \frac{1}{L}n \leq \frac{2}{L}(1-\delta)n,$$

lead to choose L greater than $\lceil 4/\sigma \rceil$, where σ is the strip width, so that Theorems 3 and 5 can be applied with $\sigma_4 = \sigma_3 = 1/L$. Denote by B_L the bit–complexity of the $\underline{\mathcal{HG}}$ Algorithm with a number of calls equal to L. Then, Theorem 3 and Lemma 2 (for $i = 0$) entail the relation

$$\mathbb{E}_{n,\psi}[B_L] = \left(\sum_{i=0}^{L-1} \mathbb{E}_{(1-\delta_i)n,\psi}[B_L] \right) \cdot \left[1 + O\big(2^{-n/L}\big) \right] + \sum_{i=1}^{L} \mathbb{E}_{n,\psi}[A_i] + \sum_{i=1}^{L} \mathbb{E}_{n,\psi}[M_i],$$

where A_i is the bit complexity of the i-th Adjust function, and M_i is the bit complexity of the various multiplications done during the i-th loop. We stop the recursion at a recursion depth H for which

$$L^H \cdot \left(\frac{n}{L^H}\right)^2 \approx_{\leq} \Theta(L^2)\mu(n)\log n,$$

$$\text{so that} \quad H = \Theta\left(\frac{\log n}{\log L}\right), \quad \frac{n}{L^H} = \log^2 n.$$

Finally, with Theorem 3, the total cost of the first call of $\underline{\mathcal{G}}$ to $\underline{\mathcal{HG}}$ is equal to

$$\mathbb{E}_{n,\psi}[B_L] = (1 + O(\frac{1}{\log\log n}))\,[1 + \varepsilon(n)]\,C_n.$$

Here, the first error term comes from the cost of the leaves due to the previous choice of H, and C_n is the mean cost of all the functions Adjust and the products, the sum taken over all the nodes of the tree of the recursive calls and $\varepsilon(n)$ is the error term which comes from the comparison of the distributions made with Theorem 3. Each application of Theorem 3, at each node of a branch, gives rise to a multiplicative term of the form $[1 + \varepsilon_i]$. The total error term $\varepsilon(n)$ (along a branch) is then of the form

$$\varepsilon(n) := \sum_{i=1}^{H} \varepsilon_i = O(\log n) O\left(2^{-n/(L^H)}\right) = O\left(n^{-\log n}\right).$$

Multiplications. The multiplications are done between integers of size $n/(2L)$ and integers of size $(1-\delta_i)n$ defined in (4.11). The cost $\mathbb{E}_{n,\psi}[M_i]$ of the products performed during the i-th loop is thus $(2L-i)\mu(n/(2L))$. The total cost of the product is then

$$\sum_{i=1}^{L} \mathbb{E}_{n,\psi}[M_i] = \Theta(L^2)\mu(\frac{n}{2L}). \tag{4.12}$$

Adjust functions. The mean cost of the i-th function Adjust decomposes as

$$\mathbb{E}_{n,\psi}[A_i] = \mathbb{E}_{n,\psi}[A_i \cdot 1_{\mathcal{U}^{(i)}}] + \mathbb{E}_{n,\psi}[A_i \cdot 1_{\mathcal{V}^{(i)}}]. \tag{4.13}$$

Here, $\mathcal{U}^{(i)}, \mathcal{V}^{(i)}$ are the ordinary and exceptional subsets relative to small quotients, namely (with $m = n/L$)

$$\mathcal{U}^{(i)} := [x;\ \pi_m(x_{\langle \delta_i \rangle}) \in \mathcal{U}_m],$$

and the set $\mathcal{U}_n$ is defined in (3.10). Together with Theorem 5 which proves that

$$\mathbb{P}_{n,\psi}[x \notin \mathcal{U}^{(i)}] = O(m^{1-(\log\log m)^{1/2}}),$$

the decomposition (4.13) entails the estimate

$$\begin{aligned}\mathbb{E}_{n,\psi}[A_i] &= \Theta(L)\ m\ \log m\ (\log\log m)^{1/2} + \\ &\qquad \Theta(L)\, O(m^2)\, O\big(m^{1-(\log\log m)^{1/2}}\big) \\ &= \Theta(L)\,\mu(\frac{n}{L})\ \left(\frac{1}{\log\log(n/L)}\right)^{1/2}.\end{aligned}$$

The total mean cost of the Adjust functions is then

$$\Theta(L^2)\mu(\frac{n}{L})\cdot\left(\frac{1}{\log\log(n/L)}\right)^{1/2}. \tag{4.14}$$

Total mean cost C_n. The mean cost C_n is the sum of the mean costs of all the functions Adjust and the products, at any node of the tree of the recursive calls. Relations (4.12) and (4.14) entail the equality

$$\begin{aligned}C_n &= \Theta(L)\sum_{h=1}^{H} L^h\mu(\frac{n}{2L^h})\ + \\ &\qquad \Theta(L)\sum_{h=1}^{H} L^h\mu(\frac{n}{L^h})\cdot\left(\frac{1}{\log\log\frac{n}{L^h}}\right)^{1/2} \\ &= \Theta\left(\frac{L}{\log L}\right)\mu(n)\log n\cdot\left[1+O\left(\frac{1}{\log\log\log n}\right)^{1/2}\right].\end{aligned}$$

Finally, we have proven the following

Theorem 6. *Consider the $\underline{\mathcal{HG}}$ algorithm defined in Figure 3, relative to a parameter L which satisfies $L \geq \max(\lceil 4/\sigma\rceil, 2)$, where σ is the width of the* US *strip. The average bit–complexity B_L of this $\underline{\mathcal{HG}}$ algorithm on inputs of size n satisfies*

$$\mathbb{E}_{n,\psi}[B_L] = \Theta\big(\frac{L}{\log L}\big)\mu(n)\log n\big[1+O\big(\frac{1}{\log\log\log n}\big)^{1/2}\big],$$

where the constants in the O and Θ–terms are uniform with respect to L.

4.2 The k-th recursive call. The k-th recursive call of $\underline{\mathcal{G}}$ to $\underline{\mathcal{HG}}$ is made on integers with size $n_k = n(1/2)^{k-1}$. It deals with values δ which belong to the interval $[1-(1/2)^{k-1}, 1-(1/2)^k]$. We consider a truncation m_k of the form $m_k = n_k/L_k$ with an integer $L_k \geq 2$, and the following constraints due to Theorem 3 are now:

$$\frac{1}{2^{k-1}L_k}\delta n \leq m_k = \frac{1}{L_k}n_k \leq \frac{2^{k-1}}{L_k}(1-\delta)n_k = \frac{1}{L_k}(1-\delta)n.$$

We can choose L_k equal to L, and then Theorems 3 and 5 can be applied. Then, in the same vein as previously, the bit-complexity $B_{k,L}$ of the k-th recursive call is

$$\begin{aligned}\mathbb{E}_{n,\psi}[B_{k,L}] &= \big[1+O(2^{-n_k})\big]\,\Theta\big(\frac{L}{\log L}\big)\mu(n_k)\log n_k\ \times \\ &\qquad \left[1+O\left(\frac{1}{\log\log\log n_k}\right)^{1/2}\right],\end{aligned} \tag{4.15}$$

where the first factor is due to Theorem 3, and the other factors due to Theorem 6.

4.3 End of the recursion. We stop calling the algorithms $\underline{\mathcal{HG}}$ inside the $\underline{\mathcal{G}}$ algorithm when the naive gcd algorithm becomes competitive, namely, when the bound $T(n) = n_M$ (see Figure 4) satisfies

$$n_M^2 = \Theta\left(\frac{L}{\log L}\right)\mu(n_M)\log n_M,$$

$$\text{so that}\quad n_M = \sqrt{n}\log n,\quad M = \Theta(\log n).$$

Then the total cost G of the $\underline{\mathcal{G}}$ Algorithm satisfies

$$\begin{aligned}\mathbb{E}_{n,\psi}[G] &= \sum_{k=1}^{M}\mathbb{E}_{n,\psi}[B_{k,L}] \\ &= \Theta\left(\frac{L}{\log L}\right) n\log^2 n\ (\log\log n)\cdot[1+\varepsilon'(n)],\end{aligned}$$

where the error term $\varepsilon'(n)$ satisfies

$$1+\varepsilon'(n) = \left[1+O(n^{-\sqrt{n}})\right]\cdot\left[1+O\left(\frac{1}{\log\log\log n}\right)^{1/2}\right],$$

$$\text{so that}\quad \varepsilon'(n) = O\left(\frac{1}{\log\log\log n}\right)^{1/2}.$$

Finally, we have proven the following:

Theorem 7. *Consider the $\underline{\mathcal{G}}$ algorithm defined in Figure 3, relative to a parameter L which satisfies $L = \max(\lceil 4/\sigma\rceil, 2)$, where σ is the width of the* US *strip. The average bit–complexity G of this $\underline{\mathcal{G}}$ algorithm on inputs of size n satisfies*

$$\begin{aligned}\mathbb{E}_{n,\psi}[G] = \Theta\left(\frac{L}{\log L}\right) n\ \log^2 n\ \log\log n\ \times \\ \left[1+O\left(\frac{1}{\log\log\log n}\right)^{1/2}\right],\end{aligned}$$

where the constants in the O and Θ–terms are uniform with respect to L.

5 Dynamical Analysis: Proofs of main Theorems 1, 2, 4.

Here, we explain how to prove these Theorems. We first present the general framework of the so–called dynamical analysis, which uses various tools, that come from analysis of algorithms (generating functions, here of Dirichlet types, described in 5.3) or dynamical systems theory (mainly transfer operators $\mathbf{H}_s$, described in 5.4). We introduce the main costs C of interest (in 5.2), and their related Dirichlet series, for which we provide an alternative expression with the transfer operator (in 5.5). For obtaining the asymptotic estimates of Theorems 1, 2, 3, 5, we extract coefficients from these Dirichlet series, in a "uniform way". Then, Property *US* (already described in 1.3) is crucial here for applying with success the Perron Formula, as in previous results of Baladi and Vallée [2], where this Property is proven to hold for the quasi-inverse $(I - \mathbf{H}_s)^{-1}$. However, the true quasi-inverse is replaced here by some of its perturbated versions for which we have to prove that Property *US* also holds. A sketch of this proof is done in 5.7.

5.1 The Euclidean Dynamical system. When computing the gcd of the integer-pair (a_0, a_1), Euclid's algorithm performs a sequence of divisions. A division $a = bq + r$ replaces the pair (u, v) with the new pair (r, u). If we consider now rationals instead of integer pairs, there exists a map S which replaces the (old) rational u/v by the (new) rational r/u, defined as

$$S(x) = \frac{1}{x} - \left\lfloor \frac{1}{x} \right\rfloor, \qquad S(0) = 0.$$

When extended to the real interval $I = [0, 1]$, the pair (I, S) defines the so–called dynamical system relative to Euclid algorithm. We denote by $\mathcal{H}$ the set of the inverse branches of S,

$$\mathcal{H} = \{h_{[q]} : x \to \frac{1}{q + x}; \quad q \geq 1\},$$

and by $\mathcal{H}^p$ the set of inverse branches of depth p (i.e., the set of inverse branches of S^p), namely $\mathcal{H}^p = \{h = h_1 \circ \cdots \circ h_p; h_i \in \mathcal{H}, \forall i\}$. The set $\mathcal{H}^\star := \cup_p \mathcal{H}^p$ is the set of all the possible inverse branches of any depth. Then, the sequence (2.3) builds a continued fraction

$$\frac{u}{v} = h(0) \quad \text{with} \quad h = h_1 \circ h_2 \circ \ldots \circ h_p \in \mathcal{H}^p. \tag{5.16}$$

One then associates to each execution of the algorithm a unique LFT $h \in \mathcal{H}^\star$ whose depth is exactly the number p of divisions performed. Remark that the i-th LFT h_i used by the algorithm is exactly the LFT relative to matrix $\mathcal{Q}_i$ of Section 2.1, so that the LFT $h_1 \circ h_2 \circ \ldots \circ h_i$ is relative to matrix $\mathcal{M}_{(i)}$ of Section 2.1. Then, the *CF*–expansion (5.16) of a_1/a_0, when splitted at depth i, creates two LFT's $b_i := h_1 \circ h_2 \circ \ldots \circ h_{i-1}$ and $e_i := h_i \circ \ldots \circ h_p$ defining each a rational number: the "beginning" rational $b_i(0)$, and the "ending" rational $e_i(0)$. The "ending" rational $e_i(0)$ can be expressed with the remainder sequence (a_i)

$$e_i(0) := h_{i+1} \circ h_{i+2} \circ \cdots \circ h_p(0) = \frac{a_{i+1}}{a_i},$$

while the "beginning" rational $b_i(0)$ can be expressed with the two co-sequences $(u_i), (v_i)$ related to coefficients of matrix $\mathcal{M}_{(i)}$

$$b_i(0) := h_1 \circ h_2 \circ \cdots \circ h_{i-1}(0) = \frac{|u_i|}{|v_i|}.$$

The main parameters of interest of the Euclid Algorithm involve the denominators sequences a_i, v_i, which are called the continuants. The continuants are closely related to derivatives of LFT's: for any LFT h, the derivative $h'(x)$ can be expressed with the denominator function D defined by $D[g](x) = cx + d$

$$\text{for } g(x) = \frac{ax + b}{cx + d} \quad \text{with} \quad \gcd(a, b, c, d) = 1,$$

as

$$h'(x) = \frac{\det h}{D[h](x)^2}. \tag{5.17}$$

And, finally, since any LFT $h \in \mathcal{H}^\star$ has a determinant of absolute value equal to 1, one has:

$$v_i = |b_i'(0)|^{-1/2}, \qquad a_i = |e_i'(0)|^{-1/2}. \tag{5.18}$$

5.2 Costs of interest. We now describe the main costs that intervene in this paper. For Theorem 1, we consider the cost $C_1 := P_\delta$ for $\delta \in [0, 1]$, defined by the relation (2.8). This means that $P_\delta(x) = k$ iff

$$\ell(x_k) \leq \lfloor (1 - \delta)\ell(x) \rfloor < \ell(x_{k-1}).$$

For Theorem 2, we consider the cost C_2 (which depends on the interval J),

$$C_2(x) = \mathbf{1}_{\{x_{\langle \delta \rangle} \in J\}}.$$

For Theorem 4, we consider the cost C_3 (which depends on an integer M)

$$C_3(x) = \mathbf{1}_{\Omega^{[<M]}}(x),$$

where $\Omega^{[<M]}$ is the subset of Ω formed with the rationals whose all the quotients of the Continued Fraction expansion are less than M.

5.3 Dirichlet series. For analysing a cost C, we deal with the generating Dirichlet series of this cost C. We denote by $\Omega, \widetilde{\Omega}$ the set of possible inputs,

$$\Omega := \{(u,v); 0 \le u \le v\},$$
$$\widetilde{\Omega} := \{(u,v); 0 \le u \le v, \gcd(u,v) = 1\}.$$

We denote by $\widetilde{\Omega}_n, \Omega_n$ the subsets of inputs with $\ell(v) = n$, and we will explain later why it will be sufficient to deal with inputs of Ω (which is, from the algorithmic point of view, the set of trivial inputs...).
To any cost C, defined on Ω or $\widetilde{\Omega}$, we associate Dirichlet series

$$F_C(s) = \sum_{(u,v)\in\Omega} \frac{1}{v^{2s}}\, C(u,v), \quad \widetilde{F}_C(s) = \sum_{(u,v)\in\widetilde{\Omega}} \frac{1}{v^{2s}}\, C(u,v).$$

Then,

$$F_C(s) = \sum_{v\ge 1} \frac{c_v}{v^{2s}}, \quad \widetilde{F}_C(s) = \sum_{v\ge 1} \frac{\widetilde{c}_v}{v^{2s}},$$

where c_v, $\widetilde{c}_v$ denote the cumulative costs of C on

$$\omega_v := \{(u,v) \in \Omega\}, \qquad \widetilde{\omega}_v := \{(u,v) \in \widetilde{\Omega}\},$$

namely,

$$c_v = \sum_{(u,v)\in\omega_v} C(u,v), \qquad \widetilde{c}_v = \sum_{(u,v)\in\widetilde{\omega}_v} C(u,v).$$

For the trivial cost ($C \equiv 1$), the corresponding cumulative costs a_v or $\widetilde{a}_v$ are just the cardinalities of subsets $\omega_v, \widetilde{\omega}_v$. The mean values of the cost C on $\Omega_n, \widetilde{\Omega}_n$ are then given by the ratio of partial sums,

$$\mathbb{E}_n[C] = \frac{\sum_{\ell(v)=n} c_v}{\sum_{\ell(v)=n} a_v}, \qquad \widetilde{\mathbb{E}}_n[C] = \frac{\sum_{\ell(v)=n} \widetilde{c}_v}{\sum_{\ell(v)=n} \widetilde{a}_v}. \tag{5.19}$$

We are mainly interested by some particular costs C. We will provide alternative expressions for $F_C(s)$, which deal with the transfer operator $\mathbf{H}_s$ relative to the Euclidean dynamical system. Then, the singularities of the Dirichlet series will be related to the dominant spectral objects of the transfer operator $\mathbf{H}_s$, and become apparent. This will lead to the asymptotic study of the coefficients of these Dirichlet series. We first recall some basic facts about transfer operators.

5.4 Transfer operators. The main tool of dynamical analysis is the transfer operator [23], denoted by $\mathbf{H}_s$. It generalizes the density transformer $\mathbf{H}$ that describes the evolution of the density: if $f = f_0$ denotes the initial density on I, and f_1 the density on $\mathcal{I}$ after one iteration of S, then f_1 can be written as $f_1 = \mathbf{H}[f_0]$, where $\mathbf{H}$ is defined by

$$\mathbf{H}[f](x) = \sum_{h\in\mathcal{H}} |h'(x)|\, f\circ h(x). \tag{5.20}$$

It is useful to introduce a more general operator that depends on a complex parameter s,

$$\mathbf{H}_s[f](x) = \sum_{h\in\mathcal{H}} |h'(x)|^s\, f\circ h(x) = \sum_{m\ge 1} \frac{1}{(m+x)^{2s}} f\left(\frac{1}{m+x}\right),$$

and multiplicative properties of derivatives entail that

$$\mathbf{H}_s^p[f](x) = \sum_{h\in\mathcal{H}^p} |h'(x)|^s\, f\circ h(x),$$
$$(I - \mathbf{H}_s)^{-1}[f](x) = \sum_{h\in\mathcal{H}^\star} |h'(x)|^s\, f\circ h(x).$$

Now, relation (5.17) between the denominator and the derivative of a LFT, and the fact that any element of $\mathcal{H}^\star$ has a determinant equal to ± 1, entail an alternative expression for the transfer operator,

$$\mathbf{H}_s^p[f](x) = \sum_{h\in\mathcal{H}^p} \frac{1}{D[h](x)^{2s}}\, f\circ h(x)$$

$$(I - \mathbf{H}_s)^{-1}[f](x) = \sum_{h\in\mathcal{H}^n} \frac{1}{D[h](x)^{2s}}\, f\circ h(x),$$

which shows, with (5.18) that the transfer operator can be viewed as a generating operator for denominator sequences a_i, v_i.

5.5 The Dirichlet series $F_C(s)$. We look for an alternative expression of the Dirichlet series $F_C(s)$, as a function of operator $\mathbf{H}_t$.

Cost C_3 for Theorem 5. The Dirichlet series relative to the third cost C_3 is

$$F_3(s) = (I - \mathbf{H}_{[<M],s})^{-1}[f](0),$$

and involves the the constrained operator $\mathbf{H}_{[<M],s}$ defined by

$$\mathbf{H}_{[<M],s}[f] := \sum_{m<M} \frac{1}{(m+x)^{2s}} \cdot f\left(\frac{1}{m+x}\right).$$

This constrained operator can be viewed (when M is large) as a perturbation of the plain transfer operator $\mathbf{H}_s$.

Cost C_1 for Theorem 1. A main tool for studying the second cost P_δ, via its moment generating function

$\mathbb{E}[\exp(tP_\delta)]$, is the Dirichlet series $F_1(s,w,t)$ which depends on three parameters s, w, t and is equal to

$$(I - \mathbf{H}_{s+w})^{-1} \circ (\mathbf{H}_{s+w} - \mathbf{H}_s) \circ (I - e^t \mathbf{H}_s)^{-1}[f](0).$$

Cost C_2 for Theorem 2. A main tool for studying the distribution of $x_{\langle\delta\rangle}$ (via the estimate of $\mathbb{P}_n[x_{\langle\delta\rangle} \in J]$ is the Dirichlet series which depends on three parameters s, w, together with the interval J,

$$F_2(s,w,J) = (I - \mathbf{H}_{s+w})^{-1}[f_{s,w,J}](0) \qquad \text{with}$$

$$f_{s,w,J}(x) = 1_J(x) \cdot (\mathbf{H}_{s+w} - \mathbf{H}_s) \circ (I - \mathbf{H}_s)^{-1}[f](x).$$

5.6 Spectral properties of the transfer operator $\mathbf{H}_s$ We now recall the main properties of the transfer operator $\mathbf{H}_s$ and its quasi-inverse $(I - \mathbf{H}_s)^{-1}$. These properties depend on the Banach space where the operator acts. Here, the Banach space is $\mathcal{C}^1(\mathcal{I})$, and we recall now the main properties of the operator $\mathbf{H}_s$, when acting on this functional space.

For $\Re(s) > 1/2$, the operator $\mathbf{H}_s$ acts on $\mathcal{C}^1(\mathcal{I})$ and the map $s \to \mathbf{H}_s$ is analytic. For $s = 1$, the operator is quasi–compact: there exists a spectral gap between the unique dominant eigenvalue (that equals 1, since the operator is a density transformer) and the remainder of the spectrum. By perturbation theory, these facts — existence of a dominant eigenvalue $\lambda(s)$ and of a spectral gap— remain true in a complex neighborhood $\mathcal{V}$ of $s = 1$. There, the operator splits into two parts: the part relative to the dominant eigensubspace, denoted $\mathbf{P}_s$, and the part relative to the remainder of the spectrum, denoted $\mathbf{N}_s$, whose spectral radius is strictly less than $\eta|\lambda(s)|$(with $\eta < 1$). This leads to the following spectral decomposition

$$\mathbf{H}_s[f](x) = \lambda(s)\mathbf{P}_s[f](x) + \mathbf{N}_s[f](x),$$

which extends to the powers $\mathbf{H}_s^n$ of the operator

$$\mathbf{H}_s^n[f](x) = \lambda^n(s)\mathbf{P}_s[f](x) + \mathbf{N}_s^n[f](x), \tag{5.21}$$

and finally to the quasi-inverse $(\mathbf{I} - \mathbf{H}_s)^{-1}$

$$(I-\mathbf{H}_s)^{-1}[f](x) = \frac{\lambda(s)}{1-\lambda(s)}\mathbf{P}_s[f](x) + (\mathbf{I}-\mathbf{N}_s)^{-1}[f](x). \tag{5.22}$$

The first term on the right admits a pole (of order 1) at $s = 1$, while the second term is analytic on the half–plane $\{\Re(s) > 1\}$. The dominant eigenvalue $\lambda(s)$ is analytic in a neighborhood of $s = 1$, and the pressure function $\Lambda(s) := \log \lambda(s)$ plays an important rôle.

5.7 *US* Property for the Dirichlet series $F_C(s)$. With this alternative expression of $F_C(s)$ at hand, we now perform the second step: we find the dominant singularities of this Dirichlet series and their nature, and then transfer this information for obtaining asymptotic expressions of their coefficients. As a main tool, we rely on convenient "extractors" which express coefficients of series as a function of the series itself. There exist an easy "extractor" for Dirichlet series: Tauberian Theorems. However, they do not provide remainder terms, and they are not adapted for our study, since we wish to obtain uniform estimates with respect to auxiliary parameters δ, w, t, J. We then adopt the Perron Formula, which may provide remainder terms, as soon as we have a precise knowledge of $F_C(s)$ on vertical strips.

The Perron Formula of order two (see [12]) is valid for a Dirichlet series $F(s) = \sum_{n\geq 1} a_n n^{-s}$ and a vertical line $\Re s = L > 0$ inside the convergence domain of F,

$$\Psi(T) := \sum_{n\leq T} a_n(T-n) = \frac{1}{2i\pi}\int_{L-i\infty}^{L+i\infty} F(s)\frac{T^{s+1}}{s(s+1)}\,ds\,.$$

It is next natural to modify the integration contour $\Re s = L$ into a contour which contains a unique pole of $F(s)$, and it is thus useful to know that the Property *US* [Uniform Estimates on Strips] holds [see Section 1.3].

From works of Dolgopyat [11] and Baladi-Vallée [2], we know that $(I - \mathbf{H}_s)^{-1}$ satisfies the *US* Property, with a strip of width σ. The real σ mentioned in Theorem 2, 3 and 5 is precisely the one associated to this *US* Property.

For Theorem 2, the Dirichlet series $(1/w)F_2(s,w,J)$ defined in Section 5.5 can be viewed as a perturbation of

$$F_2(s,J) := (I - \mathbf{H}_s)^{-1}[\mathbf{1}_J \cdot \mathbf{H}'_s \circ (I - \mathbf{H}_s)^{-1}[f]](0).$$

This Dirichlet series $F_2(s,J)$ involves the operator $\mathbf{H}'_s := (d/ds)\mathbf{H}_s$, has a pôle of order 2 at $s = 1$, and satisfies for s close to 1,

$$F_2(s,J) \sim \frac{1}{(s-1)^2}\left(\frac{1}{\lambda'(1)}\right)^2 \varphi(0)\left(\int_J \mathbf{H}'[\varphi](t)dt\right),$$

where φ is the Gauss density defined in (1.1). This explains why $\psi = \mathbf{H}'[\varphi]$ introduced in (1.2) plays a central rôle in our analyses.

For Theorem 4, Cesaratto and Vallée studied in [7] the constrained operator $\mathbf{H}_{[<M],s}$ defined by

$$\mathbf{H}_{[<M],s}[f] := \sum_{m<M} \frac{1}{(m+x)^{2s}} \cdot f\left(\frac{1}{m+x}\right)$$

and proved that it satisfies the *US* Property on the same strip as the plain quasi-inverse.

6 Conclusion

This paper provides the first average-case analysis of a subquadratic gcd algorithm. We therefore extend the domain of applicability of dynamical analysis techniques, and show that such methods are also efficient for studying more complex Euclidean algorithms. The type of analysis performed here requires a precise study of the interrupted algorithms, and a precise description of the evolution of the distribution during the execution of the algorithm. This heavily uses the powerful tools of distributional analysis provided by [2, 21].

It would be also interesting to adapt the methodology developed here to other subquadratic gcd algorithms. We have in mind the algorithm recently designed by Stehlé and Zimmermann [25], based on a division using the least significant bits of the integers. The analysis of the plain gcd algorithm using this division is done in [10]. This is clearly a first step in that direction; however, a complete analysis of the SZ Algorithm would use Property *US*, and this Property is not known to hold in the context of the dynamical system related to this gcd using the least significant bits. Anyway, comparing the average–case behaviour of the SZ algorithm to other $\mathcal{HG}$ algorithms would be interesting since it would point out the influence of the division used, and explain experimental results observed in [25, 22].

References

[1] Baladi, V. and Vallée, B. Exponential Decay of Correlations for surface semi-flows without finite Markov partitions, Proceedings of the American Mathematical Society, 133 (3) pp 865-874, 2004.

[2] Baladi, V. and Vallée, B. Euclidean Algorithms are Gaussian, Journal of Number Theory, Volume 110, Issue 2 (2005) pp 331–386.

[3] Bedford, T., Keane, M., and Series, C., Eds. *Ergodic Theory, Symbolic Dynamics and Hyperbolic Spaces*, Oxford University Press, 1991.

[4] Brent, R.P. Analysis of the Binary Euclidean algorithm, Algorithms and Complexity, New directions and recent results, ed. by J.F. Traub, Academic Press 1976, pp 321–355.

[5] Cesari, G. Parallel Implementation of Schönhage's Integer GCD Algorithm, Proceedings of ANTS-III, LNCS 1423, pp64-76.

[6] Cesaratto, E. Remarks and extensions on the paper "Euclidean algorithms are gaussian", by V. Baladi et B. Vallée, in preparation.

[7] Cesaratto, E. and Vallée, B. Personal communication, to be submitted.

[8] Dixon, J. D. The number of steps in the Euclidean algorithm, *Journal of Number Theory 2* (1970), 414–422.

[9] Daireaux, B., and Vallée, B. Dynamical analysis of the parameterized Lehmer-Euclid Algorithm, Combinatorics, Probability, Computing, pp 499–536 (2004).

[10] Daireaux, B., Maume-Deschamps, V., Vallée, B. The Lyapounov Tortoise and the Dyadic hare, Discrete Mathematics and Theoretical Computer Science 2005, Proceedings of AofA'05, pp 71-94 (2005).

[11] Dolgopyat, D. On decay of correlations in Anosov flows, *Ann. of Math.* 147 (1998), pp 357–390.

[12] Ellison, W. and Ellison, F. *Prime Numbers,* Hermann, Paris, 1985.

[13] Flajolet, P. and Sedgewick, R. *Analytic Combinatorics*, Book in preparation (1999), see also INRIA Research Reports 1888, 2026, 2376, 2956.

[14] Heilbronn, H. On the average length of a class of continued fractions, Number Theory and Analysis, ed. by P. Turan, New-York, Plenum, 1969, pp 87-96.

[15] Hensley, D. The number of steps in the Euclidean algorithm, *Journal of Number Theory* 49, 2 (1994), 142–182.

[16] Jebelean, T. Practical Integer Division with Karatsuba Complexity Proceedings of ISSAC'97.

[17] Jebelean, T. A Double-Digit Lehmer–Euclid Algorithm for finding the GCD of Long Integers. Journal of Symbolic Computation (1995) 19, pp 145–157.

[18] Knuth, D.E. *The art of Computer programming*, Volume 2, 3rd edition, Addison Wesley, Reading, Massachussets, 1998.

[19] Knuth, D.E. The analysis of algorithms, Actes du Congrès des Mathématiciens, Volume 3, pp 269-274, Gauthier-Villars 1971.

[20] Lehmer, D. H. Euclid's algorithm for large numbers. Am. Math. Mon. (1938) 45 pp 227–233.

[21] Lhote, L. and Vallée, B. Sharp estimates for the main parameters of the Euclid Algorithm, Proceedings of LATIN'06, LNCS 3887, pp 689-702.

[22] Möller, N. On Schönhage's algorithm and subquadratic integer gcd computation, submitted.

[23] Ruelle, D. *Thermodynamic formalism,* Addison Wesley (1978).

[24] Schönhage, A. Schnelle Berechnung von Kettenbruchentwicklungen, Acta Informatica pp 139–144 (1971)

[25] Stehlé, D. and Zimmermann, P. A Binary Recursive Gcd Algorithm, Proceedings of ANTS'04, LNCS 3076 (2004), pp 411-425.

[26] Vallée, B. Dynamical Analysis of a Class of Euclidean Algorithms, Theoretical Computer Science, vol 297/1-3 (2003) pp 447–486.

[27] Vallée, B. Euclidean Dynamics, [55 pages], Discrete and Continuous Dynamical Systems, 15 (1) May 2006, pp 281-352.

[28] Vallée, B. Digits and Continuants in Euclidean Algorithms. Ergodic Versus Tauberian Theorems, Journal de Théorie des Nombres de Bordeaux 12 (2000) pp 531-570.

[29] Yap, C.K. *Fundamental Problems in Algorithmic Algebra*, Princeton University Press (1996).