GENEALOGY OF LATTICE REDUCTION: ALGORITHMIC DESCRIPTION AND DYNAMICAL ANALYSES

BRIGITTE VALLÉE

Abstract. The present study describes the main algorithms devoted to solving the lattice reduction problem. This is a central algorithmic problem, due to its intrinsic theoretical interest, together to its multiple possible applications, located at many various areas in the interface between mathematics and computer science : computational number theory, integer programming but also complexity theory and cryptology. We first describe the algorithms themselves, inside their genealogy, and explain how the main ideas of small dimensions are used in higher dimensions. We are mainly interested in their probabilistic analysis, and wish to describe in a probabilistic way the main properties of their execution or the geometry of their outputs. Finally, the methodology that conducts these analyses is itself a main subject of interest, as it involves an original mixing between probabilistic modelling of the inputs, analytic combinatorics, and also tools that come from dynamical systems. This method, called dynamical analysis, is completely fruitful in small dimensions, and well explains the transition between the two smaller dimensions. For higher dimensions, such a direct approach is no longer possible, but it can be adapted via the introduction of simplified models.

1. Introduction

1.1. **Genealogy of lattice reduction.** A lattice is probably the simplest structure of the discrete linear algebra. A lattice $\mathcal{L} \subset \mathbb{R}^n$ of dimension d is a discrete additive subgroup of $\mathbb{R}^n$. Such a lattice is generated by integral linear combinations of vectors from a family $\boldsymbol{b} := (b_1, b_2, \dots b_d)$ of $d \leq n$ linearly independent vectors of $\mathbb{R}^n$, which is called a basis of the lattice $\mathcal{L}$. A lattice is possibly generated by infinitely many bases that are related to each other by integer matrices of determinant ± 1. However, when the ambient space is endowed with its Euclidean structure, these bases may strongly differ from their Euclidean properties (see Figure 1). A lattice reduction algorithm aims at finding, from a given basis of $\mathcal{L}$, a so-called *reduced* basis $\widehat{\boldsymbol{b}}$ of $\mathcal{L}$ with good Euclidean properties, namely with *short enough* and *almost orthogonal* vectors.

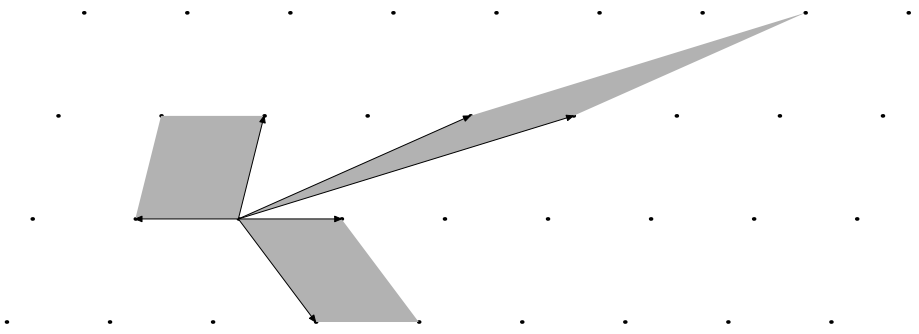


Figure 1. A two-dimensional lattice and three of its bases represented by the parallelogram they span. The basis on the left is "reduced" whereas the two other ones are skew.

In two dimensions, the Gauss algorithm solves the reduction problem in an optimal sense: it returns a minimal basis, after a number of iterations which is at most linear with respect to the input size. It can be viewed as a plain extension of the Centered Euclid algorithm, and it can be generalized itself in other small dimensions $d \leq 4$ (see for instance [68] or [61]). However, algorithms of this optimal quality no longer exist in higher dimensions where there are various points of view on the lattice reduction. We focus here on the LLL algorithm, designed in 1982 by Lenstra, Lenstra and Lovász in [43], that can be viewed as an approximation algorithm which finds a good basis (not optimal generally speaking) after a polynomial number of iterations (not linear generally speaking). This algorithm coincides with the Gauss algorithm for $d = 2$. On a lattice of large dimensions $d \geq 3$, it performs a succession of Gauss reduction steps on two-dimensional sublattices of the input lattice. This is why it is important to precisely describe and study the case $d = 2$ and thus, the case $d = 1$ which is itself very helpful to understand the case $d = 2$.

A lattice is clearly a natural objet to model linear structures which arise in algorithmics. In the context of lattice reduction, the structure is viewed via its embedding in the vectorial space $\mathbb{R}^n$ endowed with its Euclidean structiure, and the lattice reduction problem is a central problem in the interplay between *algebra* and *metric Euclidean topology*. This explains the multiple possible applications of lattice reduction, for instance in integer programming, computational number theory, or more recently in cryptology, where lattices are now one of the main tools for designing cryptosystems.

1.2. **Probabilistic analysis of lattice reduction algorithms.** This survey also describes complementary approaches which can be adopted in a probabilistic analysis of the main lattice reduction algorithms.

Probabilistic analysis aims to precisely quantify the *generic* behavior of an algorithm – for instance the main characteristics of its execution, or its outputs–. When the set of inputs is endowed with a given distribution, these characteristics become random variables and may be studied in this way, via their expectation, their variance, or their distribution, notably when the size of the inputs become large. Beyond its intrinsic theoretical interest, such an analysis allows a fine understanding of the algorithm and a justification of many experimental facts observed. It often conditions its algorithmic improvements in its application areas. This is more important in the present framework, due to the multiple possible applications of the lattice reduction problem. In this case, there are many experimental observations, regarding the execution of the algorithms and the geometry of their outputs, that pose important questions and lead to natural conjectures. The results obtained in this perspective may then be expected to contribute to a *general algorithmic strategy* for lattice reduction.

We begin with smaller dimensions, and we first explain how the dynamical methodology is proven fruitful for small dimensions d, corresponding to the Euclid algorithm ($d = 1$) and to the Gauss algorithm ($d = 2$). Such an approach mixes probabilistic and combinatorial tools, and makes a great use of the underlying dynamical system. It leads to a precise description of the probabilistic behaviour of the two algorithms, and explains their resemblances, and their differences. We introduce the notion of *valuation* which quantifies the *difficulty* of the inputs and also provides a precise transition between the two algorithms, when the basis of the two–dimensional lattice becomes "flat". Of course, as small dimensions are strongly used in the algorithmic design in high dimensions, such a study for small dimensions constitute an important step in the analysis of lattice reduction in any (high) dimension.

However, for higher dimensions, the probabilistic behavior of lattice reduction algorithms is very complex and far from being well understood. There are two main difficulties in such a probabilistic analysis: the first one comes from the structure of the LLL algorithm itself, which yet gives rise to a dynamical system, but very difficult to deal with; the second one is due to the needed modelling of the inputs, and, as the algorithm is used in various domains, one has to design dedicated modelling for each application. Then, we design simplified models, both for the execution (for which we introduce simplified dynamical systems) and the probabilistic modelling of the inputs (where we extend the notion of valuation).

1.3. **Plan of the paper.** Section 2 is devoted to probabilistic analysis; it introduces and describes the general features of dynamical analysis. Then, the following two sections focus on smaller dimensions, first the one-dimensional-case with the Euclid Algorithm (Section 3) then the two dimensional-case with the Gauss algorithm (Section 4). Each section begins with a description of the algorithm, and then provides its precise probabilistic analysis, based on dynamical methods. Section 4 also introduces the notion of valuation which well explains the transition between these two smaller dimensions. Finally, Section 5 is devoted to higher dimensions: after a precise description of the LLL algorithm, it surveys the previous existing probabilistic analyses. It then introduces the simplified models, both for the distribution of the inputs or the execution of the algorithm, that can be used in the first steps of a (future) more *realistic* analysis of this central algorithmic process.

1.4. **Other relevant surveys.** There are other surveys that may be interesting for the reader. Paper [77] focuses on Sections 2 and 3. The methodology of dynamical analysis is thorougly described and applied to the whole class of Euclidean Algorithms. For precisions about the Gauss Algorithm and its analysis, one can read papers [79] or thesis [80]. There is now a large litterature about the LLL Algorithm. The book [55] is totally devoted to the algorithm, together with its multiple applications. A shorter paper [67] may also be of interest.
The present survey corresponds to a course given in the Summer School entitled "Natural extension of arithmetic algorithms and S-adic systems" (20-24 July 2015, Tambara Institute of Mathematical Sciences, The University of Tokyo).

2. Probabilistic analysis of algorithms and dynamical systems.

This section first introduces the main concepts of the probabilistic analysis of algorithms, first introduced by Knuth in his books [36] and further developed in a genuine scientific domain, called analytic combinatorics, by Flajolet and Sedgewick in the book [26]. We then explain the particularities of the algorithms we wish to analyse, due to two main factors: they deal with numbers, and there exists a dynamical system that underlies the algorithm. The arithmetical operations create carries which perturbate the analysis, and do not permit a plain application of analytic combinatorics. On the other side, the underlying dynamical system is a powerful tool to understand these perturbations, mainly via its transfer operator. Then, dynamical analysis is based by the conjoint use of tools that come from both domains –analytic combinatorics and dynamical systems theory – and the transfer operator plays there a crucial role in the crossroads, the role of a generating operator. As in classical analytic combinatorics, there are then two steps: a combinatorial step which expresses the main probabilistic objects in terms of the transfer operator, an analytic step which transfers the analytic properties of the operator into asymptotic properties.

2.1. **Probabilistic analysis of algorithms.** We consider an algorithm with its set of inputs Ω, and a parameter (also called a cost) C defined on Ω which describes
– the execution of the algorithm (number of iterations, bit–complexity)
– or the geometry of the output
We gather the inputs with respect to their size, and consider the family of subsets

$$\Omega_n := \{\omega \in \Omega, \quad \text{size}(\omega) = n\}.$$

We then consider a distribution on Ω_n(for instance the uniform distribution), and the restriction C_n of the cost C to Ω_n becomes a random variable. We study it in a probabilistic way: we wish to estimate the mean value of C_n, its variance, its distribution... in an asymptotic way when the size n becomes large.

2.2. **Analytic combinatorics and generating functions.** Analytic combinatorics [26] is a modern basis for the quantitative study of combinatorial structures (such as words, trees, mappings, and graphs), with applications to probabilistic study of algorithms that are based on these structures. The idea is to transform a sequence (a_n) related to the probabilistic analysis of the family (C_n) related to a cost C

into a unique function $A(z)$ which "gathers" the whole sequence. There exist generating functions of various types

$$A(z) := \sum_{n\geq 0} a_n z^n, \qquad \widehat{A}(z) := \sum_{n\geq 0} a_n \frac{z^n}{n!}, \qquad \widetilde{A}(s) := \sum_{n\geq 1} \frac{a_n}{n^s}$$

The first one is an ordinary generating function, the second one is an exponential generating function, whereas the third one is a Dirichlet generating function. One then uses two types of methods: symbolic and analytic. The symbolic side views generating functions as formal objects; it uses combinatorial tools and derives characterizations of generating functions. The analytic side treats those functions as functions in the complex plane, uses many various analytic tools and leads to precise characterization of limit distributions.
These objects are very useful when the distribution of data does not change "too much" during the execution of the algorithm. In this case, the algorithm gives rise to a precise symbolic description of the associated generating functions. This is the case for instance for the Euclid Algorithm on polynomials (see for instance [11]).

Here, we mainly deal with integer numbers, and this is no longer the case. The existence of carries in the main arithmetical operations changes the distribution of data; as the probabilistic study of the dynamical system underlying the algorithm explains how the distribution of data evolves during the execution of the algorithm, it may be of central interest in this case. This leads to the paradigm of

Dynamical Analysis := Analysis of Algorithms + Dynamical Systems

2.3. **Dynamical Analysis.** Analytic combinatorics can be viewed as an interaction between the discrete world (symbolic view on generating functions) and the continuous word (analytic view on generating function). But it proves often useful to also operate a more direct interaction on the algorithm itself and perform the following three main steps.

(a) The discrete algorithm is extended into a continuous process.
(b) This continuous process is studied – more easily, using all the analytic tools.
(c) We wish to return to the discrete algorithm,

Remark that the discrete data are of zero measure amongst the continuous data. This is why Step (c) is often difficult. We will see in the sequel two methods for Step (c): For the Euclid algorithm (Section 3), this step is performed in an indirect way via generating functions, whereas the analysis of the Gauss algorithm (Section 4) directly considers an embedding of discrete data inside the continuous data.

We then describe the three steps of a dynamical analysis:

Input. A discrete algorithm.

Step 1. Extend the discrete algorithm into a continuous process, i.e. a dynamical system. (X, W) X compact, $W : X \to X$: it is for instance possible to replace integers by rational numbers, and then extend the process to a generic real. We are then led to a continuous dynamical system, where the discrete algorithm gives rise to particular trajectories.

Step 2. Study this (continuous) dynamical system, via its generic trajectories with a main tool associated with the dynamical system, namely the transfer operator.

Step 3. Return to the algorithm: we need comparing the discrete trajectories with the generic trajectories. We use the transfer operator as a generating operator, which generates itself ..... the generating functions or more generally the probabilistic tools used in the analysis of the discrete process.

Output. Probabilistic analysis of the algorithm.

2.4. **Dynamical systems.**

Definition 1. *A dynamical system is a pair formed by a set X and a mapping $W : X \to X$ for which there exists*

(a) a (finite or denumerable) set $\mathcal{Q}$, whose elements are called digits,

(b) a topological partition $\{X_q\}_{q\in\mathcal{Q}}$ of the set X into subsets X_q

such that the restriction W_q of W to X_q is of class $\mathcal{C}^2$ and injective.
The dynamical system is complete when all maps $W_q : X_q \to X$ are surjective.

Here, we deal with the so–called complete dynamical systems, and a special rôle is played by the set $\mathcal{H}$ of branches of the inverse function W^{-1} of W: we denote by $h_{\langle q\rangle}$ the inverse of the map W_q, so that X_q is exactly the image $h_{\langle q\rangle}(X)$. The set $\mathcal{H}^k$ is the set of the inverse branches of the iterate W^k; its elements are of the form $h_{\langle q_1\rangle} \circ h_{\langle q_2\rangle} \circ \cdots \circ h_{\langle q_k\rangle}$ and are called the inverse branches of depth k. The set $\mathcal{H}^\star := \cup_{k\geq 0}\mathcal{H}^k$ is the semi-group generated by $\mathcal{H}$.

Given an initial point x in X, the sequence $\mathcal{W}(x) := (x, Wx, W^2x, \ldots)$ of iterates of x under the action of W forms the *trajectory* of the initial point x. We say that the system has a *hole* Y if any point of X eventually falls in Y: for any x, there exists $p \in \mathbb{N}$ such that $W^p(x) \in Y$.
We give in Figure 2 four examples of dynamical systems related to Euclidean Algorithms.

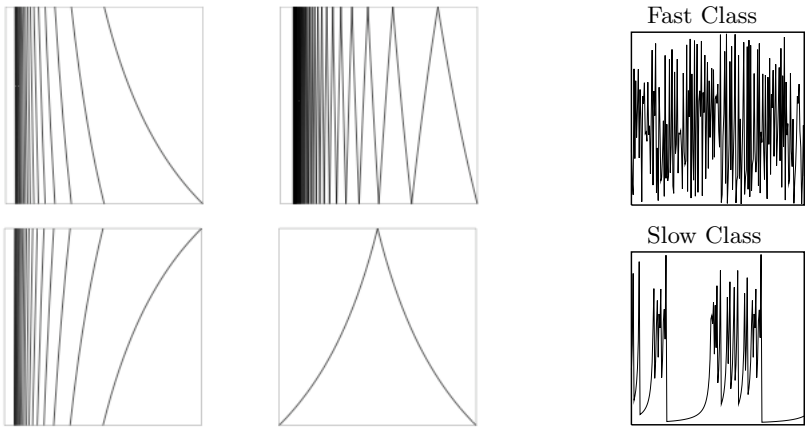


FIGURE 2. Above, dynamical systems associated with Standard and Centered algorithms; on the bottom, dynamical systems associated with By-Excess and Subtractive algorithms where there are indifferent points : $x = 1$ or 0, for which $U(x) = x, |U'(x)| = 1$.

2.5. **Transfer operators.** The main study in dynamical systems concerns itself with the interplay between properties of the transformation W and properties of trajectories under iteration of the transformation. The behavior of typical trajectories of dynamical systems is more easily explained by examining the flow of densities. The time evolution governed by the map W modifies the density on X, and the successive densities $f_0, f_1, f_2, \ldots, f_n, \ldots$ describe the global evolution of the system at discrete times $t = 0$, $t = 1$, $t = 2$, ...
The *density transformer* $\mathbf{H}$ expresses the new density f_1 in terms of the old density f_0, as $f_1 = \mathbf{H}[f_0]$. It involves the jacobians $J(h)$ of the inverse branches h of the set $\mathcal{H}$,

$$\mathbf{H}[f](x) := \sum_{h\in\mathcal{H}} J(h)(x) \cdot f \circ h(x)$$

With a parameter s, it gives rise to the (plain) *transfer operator* $\mathbf{H}_s$

$$\mathbf{H}_s[f](x) := \sum_{h\in\mathcal{H}} J(h)(x)^s \cdot f\circ h(x)$$

and, with a cost c defined on $\mathcal{H}$, it gives rise to tthe *weighted transfer operator* $\mathbf{H}_{s,w,(c)}$

$$\mathbf{H}_{s,w,(c)}[f](x) := \sum_{h\in\mathcal{H}} w^{c(h)}\, J(h)(x)^s \cdot f\circ h(x)\,.$$

Due to the multiplicative properties of the Jacobian, its k–iterate deals with the set $\mathcal{H}^k$, and the additive extension of c to $\mathcal{H}^k$,

$$\mathbf{H}^k_{s,w,(c)}[f](x) := \sum_{h\in\mathcal{H}^k} w^{c(h)} J(h)(x)^s \cdot f\circ h(x)\,,$$

and its quasi-inverse deals with the set $\mathcal{H}^\star$

$$(I-\mathbf{H}_{s,w,(c)})^{-1}[f](x) := \sum_{h\in\mathcal{H}^\star} w^{c(h)} J(h)(x)^s \cdot f\circ h(x)\,.$$

2.6. **Dynamical Analysis.** When an algorithm is associated with a dynamical system, it uses at each step the set $\mathcal{H}$, and, when it always terminates, the set of all its possible executions is described by the set $\mathcal{H}^\star$. This is why the quasi-inverse $(I-\mathbf{H}_s)^{-1}$ plays a central role in the present study and will be omnipresent in all the expressions which arise in our probabilistic studies. We use the following general framework:

Geometric properties of the dynamical system
⇓
Spectral properties for the transfer operator $\mathbf{H}_s$
in a convenient functional space.
⇓
Analytical properties of the quasi-inverse $(I-\mathbf{H}_s)^{-1}$
⇓
Probabilistic analysis of the algorithm

Figure 2 gives an instance of the influence of the geometry on the probabilistic behaviour of the trajectories. The presence of indifferent points in the two dynamical systems of the bottom line creates intermittent trajectories which remain a long time near these points. These dynamical systems give rise to *slow* algorithms. On the contrary, the two dynamical systems of the above line are expansive and the associated algorithms are *fast*.

2.7. **Historical and bibliographic notes.** A readable treatment of dynamical systems of intervals can be found in [40] and [15], and the book [10] provides a general overview on dynamical systems. The dynamical systems related with the Euclidean divisions were early studied. For instance, the system related with the classical division was first studied by Gauss himself. The dynamical system related to the Centered Euclidean division was studied by Rieger [56, 57]. Later on, the paper [77] introduces a whole class of Euclidean systems, in relation with Euclidean algorithms. Scweiger [63] also introduces a general framework for dynamical systems in relation with the Euclidean type.

The density transformer, also known as the Perron-Frobenius operator, was introduced early in the study of continued fractions (see for instance Lévy [42], Khinchin [34], Kuzmin [37], Wirsing [82] and Babenko [7]). The density transformer is a special case of a transfer operator, and the general notion of transfer operators was introduced by Ruelle, in connection with his thermodynamic formalism (see for instance [58, 59]). In the Euclidean context, it was recently deeply studied by Mayer, in a sequence of papers [49, 48, 50, 51]. The book of V. Baladi [8] provides a general and complete overview on transfer operators.

3. The Euclid Algorithm

The present section is devoted to the Euclid algorithm. This is the ancestor in the genealogy: first, it is the oldest of the three algorithms of interest; second, this is the mother of the Gauss algorithm, which is the actual first algorithm of the lattice reduction. The first sections, Section 3.1 and 3.2 describe the algorithm and its parameters of interest, and Section 3.3 presents the underlying dynamical system and its transfer operator. Then, Section 3.4 introduces the generating functions of interest and relate them with the (quasi-inverse of the) transfer operator. Then, the analytic study of this transfer operator performed in Section 3.5 leads to analytic properties of the generating functions, that are transferred into asymptotic properties of the coefficients. This leads in Section 3.6 to the average-case analysis of the algorithm. Section 3.7 describes the main ideas that are used in the distributional analysis of the Euclid algorithm.

3.1. Description of the centered Euclid Algorithm.

The classical Euclid algorithm which is "the grandfather of all the algorithms", as Knuth says, was discovered as early as 300BC. It was analysed first in the worst case in 1733 by de Lagny. It uses the classical Euclidean division. On an integer pair (u, v) with $0 < u < v$, this division computes a quotient m and a remainder r, and writes

$$v = mu + r, \qquad \text{with} \quad 0 \le r < u\,.$$

The (classical) gcd algorithm performs successive (classical) Euclidean divisions followed with exchanges.

Here, in the sequel, we consider another Euclid algorithm[1]. It is based on another division, the *centered Euclidean division*, called here the `C-Euclid` division. On an integer pair (u, v) with $0 < u \le v/2$, this division computes a quotient m, a sign $\epsilon = \pm 1$ and a remainder r, and writes

$$v = mu + \epsilon r, \qquad \text{with} \quad 0 \le r \le u/2, \quad \epsilon = \pm 1\,.$$

The pair (m, ϵ) is defined by the relations

$$m := \left\lceil \frac{v}{u} \right\rfloor \qquad \epsilon := \operatorname{sign}\left(m - \frac{v}{u}\right)\,.$$

On the input (u, v) with $0 < u \le v/2$, the `C-Euclid` algorithm has been considered by Rieger [56]; it starts with $(u_0 := v; u_1 := u)$, and performs the following steps, each step being formed with a centered division followed by an exchange:

$$\left\{\begin{array}{lcccccl} u_0 & = & m_1 u_1 & + & \epsilon_1\, u_2 & 0 < u_2 \le u_1/2, & \epsilon_1 = \pm 1 \\ u_1 & = & m_2 u_2 & + & \epsilon_2\, u_3 & 0 < u_3 \le u_2/2, & \epsilon_1 = \pm 1 \\ \cdots & = & \cdots & + & & & \\ u_{p-2} & = & m_{p-1} u_{p-1} & + & \epsilon_{p-1}\, u_p & 0 < u_p \le u_{p-1}/2, & \epsilon_1 = \pm 1 \\ u_{p-1} & = & m_p u_p & + & 0 & u_{p+1} = 0 & \end{array}\right\}.$$

The last non-zero remainder u_p is the gcd of u and v, the pairs $q_i := (m_i, \epsilon_i)$ are the digits, the remainders u_i are the *continuants* and p is the depth. Such an algorithm also computes the centered continued fraction expansion (CCFE) of the rational u/v

$$\frac{u}{v} = \cfrac{1}{m_1 + \cfrac{\epsilon_1}{m_2 + \cfrac{\epsilon_2}{\ddots + \cfrac{\epsilon_{p-1}}{m_p}}}}\,. \tag{1}$$

As well as the gcd, this continued fraction expansion is another actual output of the algorithm, that can be used as an input for all the computations that directly use this expansion for computing with the rational u/v.

[1] In the lattice reduction framework, this is the "natural" division, as we see later on

This algorithm clearly always terminates. Since the sequence of remainders u_i satisfy $u_{i+1} \leq (1/2)u_i$, the number $p := P(u, v)$ of iterations is $O(\log \max(u, v))$. We state a more precise worst-case bound in the next subsection.

3.2. **Main costs of interest for the probabilistic analysis.** The length of an input $(u, v) \in \mathbb{N}^2$ is defined as $\max(u, v)$, whereas its size is defined via the binary size ℓ of integers as $\max(\ell(u), \ell(v))$. The set Ω of the (valid) inputs, together with the set Ω_M of the inputs of length at most[2] N are defined as

$$\widetilde{\Omega} := \{(u, v) \in \mathbb{N}^2 \mid 0 < u \leq v/2\}, \qquad \widetilde{\Omega}_N := \{(u, v) \in \Omega \mid v \leq N\}\,.$$

In fact, we deal with the untilded version of these input sets, restricted with coprime[3] pairs (u, v) with $\gcd(u, v) = 1$, namely

$$\Omega := \{(u, v) \in \mathbb{N}^2 \mid 0 < u \leq v/2,\ \gcd(u, v) = 1\}, \qquad \Omega_N := \{(u, v) \in \Omega \mid v \leq N\}\,. \tag{2}$$

The main costs of interest are related to the execution of the algorithm or to its ouput. We are first interested in the number of iterations $P(u, v)$, here equal to p. But, it is also interesting to study the size of the output, closely related to the sum of the size of the used digits $q_i = (m_i, \epsilon_i)$. Generally speaking, we are interested by additive costs C, which depend on the digits $q_i = (m_i, \epsilon_i)$ via a cost $c : \mathcal{Q} \to \mathbb{R}^+$ and are defined as

$$C(u, v) := \sum_{i=1}^{p} c(q_i) \qquad \text{when} \quad \frac{u}{v} = h(0), \quad h := h_1 \circ h_2 \circ \ldots \circ h_p\,.$$

There are natural instances of such costs, for instance: – for $c = 1$, C is the number of iterations, – for $c = \mathbf{1}_q$, C is the number of digits equal to q, – for $c = \ell$ (the binary size), C is the size of the output.
There are also some other costs which are not additive, as the bit-complexity closely related[4] to

$$B(u, v) := \sum_{i=1}^{p} \ell(q_i) \log u_i\,.$$

In the sequel, we shall study these costs in a probabilistic way, as described in Section 2. We now return for a moment to the worst-case analysis. At each step, the smallest possible quotient (in the lexicographic order) is $q = (2, +1)$, then the worst-case is associated to the smallest possible sequence[5] of remainders defined by the recurrence $A_0 = A_1 = 1,\ A_{k+1} = 2A_k + A_{k-1}$, itself related to the quadratic number $1 + \sqrt{2}$ root of the polynomial $x^2 - 2x - 1 = 0$. This proves that the maximal possible value of P on Ω_N is $\log_{1+\sqrt{2}} N$.

3.3. **The underlying dynamical system.** The trace of the execution of the Euclid Algorithm on (u_1, u_0) is described by the sequence of integer pairs,

$$(u_1, u_0) \to (u_2, u_1) \to (u_3, u_2) \to \ldots \to (u_{p-1}, u_p) \to (u_{p+1}, u_p) = (0, u_p)\,.$$

When the integer pair (u_i, u_{i-1}) is replaced by the rational $x_i := u_i/u_{i-1}$, the division $u_{i-1} = m_i u_i + \epsilon_i\, u_{i+1}$ is then written as

$$x_{i+1} = \epsilon \left(\frac{1}{x_i}\right) \left(\frac{1}{x_i} - \left\lfloor \frac{1}{x_i} \right\rceil\right) \qquad \text{with} \quad \epsilon(x) := \operatorname{sign}(x - \lfloor x \rceil)\,.$$

[2]This is not exactly the framework which is described in Section 2.1 and we should deal with the size ω_N of the inputs of length N. However, it does not seem possible to conduct a precise probabilistic analysis on these sets ω_n.

[3]It seems strange to deal with "trivial" inputs on which the algorithm returns 1, but we will explain how these inputs appear in a natural way and are in a sense generic

[4]The exact bit-complexity deals with the binary size $\ell(u_i)$, but it is closely related to B.

[5]For the classical Euclid algorithm, the minimal sequence is $A_0 = A_1 = 1,\ A_{k+1} = A_k + A_{k-1}$ that is related to the golden ratio $\phi := (1 + \sqrt{5})/2$.

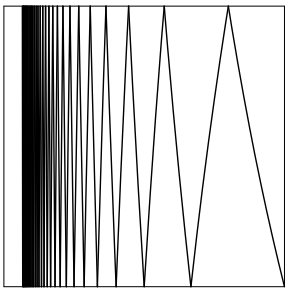


FIGURE 3. The dynamical system underlying the `C-Euclid` algorithm

If we introduce the map $U : [0, 1/2] \to [0, 1/2]$ defined as

$$U(x) = \epsilon \left(\frac{1}{x}\right) \left(\frac{1}{x} - \left\lfloor \frac{1}{x} \right\rceil\right) \quad \text{for} \quad x \neq 0, \quad U(0) = 0, \tag{3}$$

then the division is written as $x_{i+1} = U(x_i)$. An execution of the Euclidean Algorithm leads to the "trajectory" $(x, U(x), U^2(x), \ldots, 0)$. This is a particular trajectory since all the points of the trajectory are rationals and the trajectory ends at the point 0. This is then a particular trajectory of the dynamical system that we will now define.

We denote by $\mathcal{I}$ the interval $[0, 1/2]$ and introduce the dynamical system related to the pair $(\mathcal{I}, U)$ which involves the shift U defined in (3). This is a complete dynamical system represented in Figure 3. It has a denumerable system of branches $(U_{[m,\epsilon]})$

$$(m, \epsilon) \geq (2, 1), \qquad U_{[m,\epsilon]} : \left]\frac{1}{m}, \frac{2}{2m+\epsilon}\right[\longrightarrow]0, 1[, \qquad U_{[(m,\epsilon)]}(x) := \epsilon \left(\frac{1}{x} - m\right),$$

and the set $\mathcal{H}$ of the inverse branches of U is

$$\mathcal{H} := \left\{ h_{[m,\epsilon]} : \left]0, \frac{1}{2}\right[\longrightarrow \left]\frac{1}{m}, \frac{2}{2m+\epsilon}\right[, \quad h_{[m,\epsilon]}(x) := \frac{1}{m + \epsilon x}, \quad (m, \epsilon) \geq (2, 1) \right\}$$

Then, the set $\mathcal{H}$ builds one step of the CF's, whereas the set $\mathcal{H}^n$ of the inverse branches of U^n builds CF's of depth n. Finally, the set $\mathcal{H}^\star := \bigcup \mathcal{H}^n$ builds all the (finite) CF's, and the continued fraction expansion in (1) gives rise to the decomposition

$$\frac{u}{v} = h_{[m_1,\epsilon_1]} \circ h_{[m_2,\epsilon_2]} \circ \ldots \circ h_{[m_p,\epsilon_p]}(0)$$

There is a characterization of $\mathcal{H}^+$ due to Hurwitz which involves the golden ratio $\phi = (1 + \sqrt{5})/2$:

$$\mathcal{H}^+ := \left\{ h(z) = \frac{az + b}{cz + d}; \quad (a, b, c, d) \in \mathbb{Z}^4, b, d \geq 1, ac \geq 0, \right. \\ \left. |ad - bc| = 1, \quad |a| \leq \frac{|c|}{2}, \quad b \leq \frac{d}{2}, \quad -\frac{1}{\phi^2} \leq \frac{c}{d} \leq \frac{1}{\phi} \right\}. \tag{4}$$

The system $(\mathcal{I}, U)$ is defined on the real interval $\mathcal{I}$, and the Jacobian $J(h)(x)$ is just equal to $|h'(x)|$. Due to the precise expression of the set $\mathcal{H}$, the transfer operator $\mathbf{H}_s$ is expressed as, for any $x \in [0, 1/2]$,

$$\mathbf{H}_s[f](x) = \sum_{(m,\epsilon) \geq (2,1)} \left(\frac{1}{m + \epsilon x}\right)^{2s} \cdot f\left(\frac{1}{m + \epsilon x}\right). \tag{5}$$

The C-Euclid algorithm is then extended into a continuous dynamical system, which builds centered continued fraction expansions. However, the discrete process is quite *singular* since it *terminates*, whereas the continuous process *never terminates* (except for the discrete data). It seems difficult to recover the behaviour of the discrete process via the qstudy of the continuous process. This is why we perform an indirect transfer [*continuous* $\leftrightarrow$ *discrete*] via generating functions.

3.4. **Relation between Dirichlet generating functions and transfer operators.** The Dirichlet series relative to a cost X is defined as

$$S_X(s) := \sum_{(u,v)\in\Omega} \frac{1}{v^{2s}} X(u,v), \qquad S_X(s,w) := \sum_{(u,v)\in\Omega} \frac{1}{v^{2s}} e^{wX(u,v)}\,. \tag{6}$$

We consider costs X defined in Section 3.2, namely additive costs C or $\log U_i$, where U_i is the i–th continuant. We will then "mix" these two studies and consider the bit-complexity B.

We now explain why the series $S_X(s)$ is a convenient tool for studying the expectations $\mathbb{E}_N[X]$. This is due to two facts. *First*, we recover the expectations as coefficients[6] of generating functions and the equality

$$\mathbb{E}_N[X] = \frac{\sum_{n\le N} b_n}{\sum_{n\le N} a_n} \tag{7}$$

holds: here, the denominator deals with the coefficients (b_n) of the series $S_X(s)$, whereas the denominator deals with the coefficients (a_n) of the series $S_1(s)$ (related to the cost $X \equiv 1$).
Second, the series in (6) admit alternative expressions which involve various versions of the quasi-inverse $(I-\mathbf{H}_s)^{-1}$ of the transfer operator, as we now see. As this quasi-inverse is omnipresent, we denote it by $\mathbf{G}_s$ and let $\mathbf{G}_s := (I-\mathbf{H}_s)^{-1}$. We first deal with the bivariate series in (6), and we return to the first series in (6) with the derivative with respect to w, at $w=0$.
The Euclid Algorithm builds a bijection[7] between the set Ω of its inputs and the set $\mathcal{H}^\star$ of LFT's,

$$(u,v) \mapsto h \quad \text{with} \quad \frac{u}{v} = h(0), \quad \text{so that} \quad \frac{1}{v} = |h'(0)|^{1/2}\,.$$

The right equality is due to the fact that branches $h \in \mathcal{H}^\star$ are LFT's of determinant 1. It is central in our analysis, as it implies an alternative expression for the bivariate Dirichlet series, first in the case of an additive cost C,

$$S_C(s,w) := \sum_{(u,v)\in\Omega} \frac{1}{v^{2s}} e^{wC(u,v)} = \sum_{h\in\mathcal{H}^\star} w^{c(h)} |h'(0)|^s = (I - \mathbf{H}_{s,w,(c)})^{-1}[1](0)\,. \tag{8}$$

Taking the derivative wrt to w at $w=0$ gives an expression of

$$S_C(s) = \sum_{(u,v)\in\Omega} \frac{1}{v^{2s}} C(u,v) = \mathbf{G}_s \circ \mathbf{H}_s^{[c]} \circ \mathbf{G}_s[1](0) \tag{9}$$

---

[6] The formula involves sums of coefficients: this is why we deal with the set of inputs of length *at most* N.

[7] This is not exactly the case: the last step of the C-Euclid algorithm is particular and it only uses $\epsilon = 1$. Then, the LFT used by the algorithm belongs to $\mathcal{H}^\star \times \mathcal{F}$ where $\mathcal{F}$ is the set of LFT's used in the last step. In fact, any rational admits two C-CFE's, the first expansion, built by the Euclid Algorithm, and called *proper*, ends with an element of $\mathcal{F}$, while the second one (improper) cannot be produced by the Algorithm. We consider here these two C-CFE's which generate together the whole set $\mathcal{H}^\star$, and we do not study exactly costs X defined in Section 3.2 but their "smoothed version" $\widetilde{X}$ which takes into account the two possible CFE's: it is the average between the cost on the proper extension and the cost on the improper one. For all the costs which are studied here, the asymptotic behaviour of X and $\widetilde{X}$ is the same, and we shall denote by S_X the Dirichlet series relative to this smoothed version $\widetilde{X}$.

as a function of the quasi-inverse $\mathbf{G}_s := (I - \mathbf{H}_s)^{-1}$ of the plain operator $\mathbf{H}_s$ together with the weighted operator, defined as

$$\mathbf{H}_s^{[c]}[f](x) = \sum_{h \in \mathcal{H}} c(h) \cdot |h'(x)|^s \cdot f \circ h(x)\,.$$

In the same vein, the continuants $u_i := U_i(u, v)$ are also written as derivatives of LFT's. With an input (u, v) of Ω on which the algorithm performs p iterations, one associates, as previously, the LFT h of depth p such that $u/v = h(0)$ which decomposes into two LFT's g and r of depth i and $p - i$ such that $h = g \circ r$; we thus "recover" u_i with the relation $|r'(0)| = |u_i|^{-2}$, and we obtain

$$S_{\log U_I}(s, 2w) = \sum_{p \geq i} \sum_{\substack{(u,v)\in\Omega \\ P(u,v)=p}} \frac{u_i^{2w}}{v^{2s}} = \left(\sum_{p \geq i} \mathbf{H}_{s-w}^{p-i} \circ \mathbf{H}_s^i[1](0) \right) = (I - \mathbf{H}_{s-w})^{-1} \circ \mathbf{H}_s^i[1](0)\,,$$

and the generating function of the cost $\log U_i$ is obtained with the derivative with respect to w, namely

$$S_{\log U_i}(s) = -\frac{1}{2} \mathbf{G}_s \circ \mathbf{H}'_s \circ \mathbf{G}_s \circ \mathbf{H}_s^i[1](0) \tag{10}$$

It is then possible to "mix" the two approaches used for an additive cost and continuants, and obtain the generating function of the bit-complexity

$$S_B(s) = -\frac{1}{2} \mathbf{G}_s \circ \mathbf{H}'_s \circ \mathbf{G}_s \circ \mathbf{H}_s^{[c]} \circ \mathbf{G}_s[1](0)\,. \tag{11}$$

Proposition 1. *The three Dirichlet generating functions $S_X(s)$ admit expressions* (9)(10)(11) *that involve the quasi-inverse $\mathbf{G}_s$.*

Then, as announced in Section 2, the main properties of the `C-Euclid` algorithm are closely related to spectral properties of the transfer operator $\mathbf{H}_s$, when it acts on a convenient functional space. As $\mathbf{H}_1$ is the density transformer, it admits an eigenvalue equal to 1, and the quasi-inverse $\mathbf{G}_s$ is singular at $s = 1$. Furthermore, this is a "dominant" pole, and it will be possible to apply a Tauberian theorem due to Delange to transfer the analytic properties of the Dirichlet series $S_R(s)$ into asymptotic properties of its coefficients, as we now see.

3.5. **Functional analysis and Tauberian theorem.** We first state the following Tauberian theorem which will perform the transfer between analytic properties of the series and asymptotic properties of its coefficients:

Theorem. [Delange] [21] *Let $F(s)$ be a Dirichlet series with non negative coefficients such that $F(s)$ converges for $\Re(s) > \sigma > 0$. Assume that*

(i) $F(s)$ is analytic on $\Re(s) = \sigma, s \neq \sigma$, and
(ii) for some $\gamma \geq 0$, one has $F(s) = A(s)(s - \sigma)^{-\gamma - 1} + C(s)$, where A, C are analytic at σ, with $A(\sigma) \neq 0$.

Then, as $K \to \infty$, one has: $$\sum_{n \leq K} a_n = \frac{A(\sigma)}{\sigma \Gamma(\gamma + 1)}\, K^\sigma \log^\gamma K \quad [1 + \epsilon(K)\,], \quad \epsilon(K) \to 0.$$

We then describe the analytic properties of the transfer operator $\mathbf{H}_s$ on a convenient functional space which is now introduced. Consider the open disk $\mathcal{V}$ of diameter $[-1/2, 1]$ and the functional space $A_\infty(\mathcal{V})$ of all functions f that are holomorphic in the domain $\mathcal{V}$ and continuous on the closure $\overline{\mathcal{V}}$. Endowed with the sup-norm, this is a Banach space. We observe that, for any $h = h_{[m,\epsilon]} \in \mathcal{H}$ the function $x \mapsto |h'(x)|$ defined on the interval $\mathcal{I}$ extends into an analytic function $\check{h}$ defined in $\mathcal{V}$ as $\check{h}(z) = 1/(m + \epsilon z)$, and we yet denote by $\mathbf{H}_s$ the operator extended in this way. For $\Re(s) > (1/2)$, the transfer operator $\mathbf{H}_s$ acts on $A_\infty(\mathcal{V})$ and is compact. Furthermore, when weighted by a cost of moderate growth [i.e., $c(h_{\langle q \rangle}) = O(\log q)$], for w close enough to 0, and $\Re s > 1/2$, the operator $\mathbf{H}_{s,w,(c)}$ also acts on $A_\infty(\mathcal{V})$, and is also compact.

The spectral properties of the transfer operator $\mathbf{H}_s$ play a central rôle in the analysis of the algorithm. For real s, the transfer operator $\mathbf{H}_s$ has a unique dominant eigenvalue $\lambda(s)$, which is real and separated from the remainder of the spectrum by a spectral gap. THe dominant eigenfunction is denoted by ψ_s. For $s = 1$, the dominant eigenvalue of the density transformer $\mathbf{H}$ satisfies $\lambda(1) = 1$, and the dominant eigenfunction $\psi := \psi_1$ (which is then invariant under the action of $\mathbf{H}$) admits a closed form that involves the golden ratio $\phi = (1 + \sqrt{5})/2$,

$$\psi(x) = \frac{1}{\log \phi} \left(\frac{1}{\phi + x} + \frac{1}{\phi^2 - x} \right).$$

This is the analog (for the `C-Euclid` algorithm) of the celebrated Gauss density associated to the standard Euclid algorithm and equal to $(1/\log 2)1/(1 + x)$.

Moreover, due to the spectral gap, the quasi-inverse $(I - \mathbf{H}_s)^{-1}$ has a pôle at $s = 1$, and satisfies

$$(I - \mathbf{H}_s)^{-1}[f](x) \sim_{s\to 1} \frac{1}{s-1} \frac{1}{h(\mathcal{E})} \psi(x) \int_{\mathcal{I}} f(x)dx, \tag{12}$$

where the constant $h(\mathcal{E})$ is the entropy of the `C-Euclid` dynamical system, and satisfies

$$h(\mathcal{E}) = |\lambda'(1)| = \frac{\pi^2}{6 \log \phi} \approx 3.41831. \tag{13}$$

3.6. **Average-case analysis of the `C-Euclid` Algorithm.** One now mixes all the tools: one deals with the Dirichlet series $S_X(s)$ relative to cost X defined in Eq (6), together with $X \equiv 1$. Using their expressions in terms of $\mathbf{G}_s$ given in Proposition 1 together with the spectral relation (12) shows that Assertion (ii) of Tauberian Theorem is satisfied for $\sigma = 1$, with various possible values of γ. We also need to prove that $\mathbf{G}_s$ is analytic on the punctured vertical line $\Re s = 1, s \neq 1$. This *aperiodicity* property also holds[8] but is not proven here. Then, it is possible to apply the Tauberian Theorem to each of the three Dirichlet series; with Relation (7), this leads to the asymptotic study of the expectations $\mathbb{E}_N[X]$, and entails the following result.

Theorem 1. [Vallée, Akhavi and Vallée] (1995-2000) *Consider the* `C-Euclid` *algorithm on the set Ω_N formed with valid co-prime input pairs of length at most N. Then, the mean number of iterations P, the mean value of an additive cost C of moderate growth, the mean value of the bit-complexity B satisfy, when $N \to \infty$,*

$$\mathbb{E}_N[P] \sim \frac{2}{h(\mathcal{E})} \log N, \quad \mathbb{E}_N[C_{(c)}] \sim \frac{2}{h(\mathcal{E})} \mathbb{E}[c] \log N, \quad \mathbb{E}_N[B] \sim \frac{1}{h(\mathcal{E})} \mathbb{E}[\ell] \log^2 N.$$

Here, $h(\mathcal{E})$ denotes the entropy of the `C-Euclid` *dynamical system, described in (13) and $\mathbb{E}[c]$ denotes the mean value of the step-cost c with respect to the invariant density ψ. This is a constant of Khinchin's type, of the form*

$$\mathbb{E}[c] := \sum_{h \in \mathcal{H}} \int_{h(\mathcal{I})} c(h)\psi(x)dx.$$

In particular, when c is the binary length ℓ, there is a nice formula for $\mathbb{E}[\ell]$, namely

$$\mathbb{E}[\ell] = \frac{1}{\log \phi} \log \prod_{k \geq 1} \frac{2^k \phi^2 + \phi}{2^k \phi^2 - 1} \approx 2.02197.$$

The same asymptotic results hold on the set $\widetilde{\Omega}_N$ which gathers all the pairs (u, v).

[8] It plays an important role in distributional analyses.

3.7. **Distributional analysis of the the `C-Euclid` Algorithm.** There exist also distributional results [9] which show that all these costs $P, C_{(c)}$, together with a regularized version of B, admit asymptotic Gaussian laws for $N \to \infty$.

Such Gaussian laws can be proven when the moment generating function $\mathbb{E}_N(\exp[wC])$ behaves as a *uniform quasi-power* when w is close to 0 and $N \to \infty$. Then, we begin with the bivariate Dirichlet series $S_C(s, w)$ defined in (8), study its analytic properties, related to spectral properties of the transfer operator $\mathbf{H}_{s,w,(c)}$ and transfer these analytic properties into asymptotic properties of coefficients. Then, we obtain a good knowledge about the asymptotics of $\mathbb{E}_N(\exp[wC])$ when $N \to \infty$ and w close to 0. The difficulty lies on the needed uniformity with respect to w, and Tauberian theorems are now useless, as they do not provide remainder terms. We use instead the Perron Formula which provide such remainder terms as soon as the quasi-inverse $(I - \mathbf{H}_{s,w,(c)})^{-1}$ well behaves in vertical strips, with a polynomial growth when $|\Im s|$ becomes large. This can be viewed as a *strong* version of the *aperiodicity* property which was already needed for proving Assertion (i) of Tauberian Theorem to hold. Dolgopyat [23] explains how geometric properties of the underlying dynamical system may entail such a behaviour for the plain quasi-inverse $\mathbf{G}_s = (I - \mathbf{H}_s)^{-1}$, and he introduces the UNI condition which states that the system must be *quite different* from a system with *affine branches*. He proves that UNI Conditions entail the *strong aperiodicity* property for $\mathbf{G}_s$. As the `C-Euclid` system satisfies the UNI Condition, this leads to a general result that we only state here for the number of iterations P:

Theorem 2. (Baladi and Vallée) (2003) *Consider the* `C-Euclid` *algorithm on the set Ω_N formed with valid co-prime input pairs of length at most N. Then, the number of iterations P asymptotically a Gaussian law, and there exist two constants $\mu > 0$ and $\rho > 0$ such that, for any N, and any $y \in \mathbb{R}$*

$$\mathbb{P}_N \left[(u, v); \frac{C(u, v) - \mu \log N}{\rho \sqrt{\log N}} \leq y \right] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{y} e^{-x^2/2}\, dx + O\left(\frac{1}{\sqrt{\log N}} \right) .$$

Moreover, the two constants μ and ρ are expressed with the pressure function $\Lambda(s) := \log \lambda(s)$, namely

$$\mu = \frac{2}{|\Lambda'(1)|} \quad \rho^2 = 2 \frac{|\Lambda''(1)|}{|\Lambda'(1)^3|} > 0 .$$

The constant ρ (called the Hensley constant) does not seem to have a closed form, but it is polynomial-yime compuyable [45].

3.8. **Historical notes.**

Euclidean algorithmics. A general description of Euclidean Algorithms is provided in Knuth's and Shallit's vivid accounts [36, 62]. There are many various Euclidean algorithms, and the survey of the author [77] provides a general description, together with many bibliographic references and designs a general framework for their dynamical analysis. There are many algorithms which were studied: a whole class is presented in [76]; but there are many others, as the binary algorithm [73], the α-Euclidean algorithms [12] or a natural algorithm [17] which may be modelled as a race between a Lyapounov tortoise and a dyadic dare....

The standard Euclidean Algorithm was analysed first in the average-case around 1969 independently by Heilbronn [27] and Dixon [22]. The centered algorithm was studied by Rieger [56]. The methods used till the early 1980's are quite various, and they range from combinatorial (de Lagny, Heilbronn) to probabilistic (Dixon).

Euclidean dynamics: the average-case. Inside the Euclidean framework, most of dynamical studies concern the continuous point of view [metric properties of continued fraction expansions for instance] (see for instance [33]) and not the discrete analysis of gcd algorithms. The general reference for Euclidean dynamical analysis is [77]. Papers [24, 25] are themselves survey papers where transfer operators are used for analysing the Euclid Algorithm together some of its generalization on higher dimensions. A precise description of Euclidean dynamical analyses can be found in the already cited papers [73, 17, 12]. Papers

[6, 74] introduce the main parameters: digits, continuants, bit-complexities, and gives a panorama for their analyses. Paper [18] deeply studies (in the average case) the particular parameter "continuant at a fraction of the depth".

The Lehmer-Euclid algorithm is an improvement of the Euclid algorithm when applied for large integers. It was introduced by Lehmer [41] and first analyzed in the worst–case by Sorenson [65] This is an Interrupted Euclidean algorithm which depends on some parameter $\alpha \in [0,1]$, and, when running with an input (u,v), it performs the same steps as the usual Euclidean algorithm, but it stops as soon as the current integer is smaller than v^α. The interrupted algorithm was studied in [18]. Moreover, this point of view was later used to design a gcd algorithm based on a *Divide and Conquer* principle which has been analyzed in [13].

Distributional analysis. Concerning the standard Euclidean algorithm and the number of steps, Hensley [28] has obtained a Central Limit Theorem, and a Local Limit Theorem with speed of convergence $O((\log N)^{-1/24})$. Hensley has used the transfer operator $\mathbf{H}_s$ to obtain distributional results on rational trajectories upon aproximating discrete measures on rationals by continuous measures. In particular, his approach avoids parameters s of large imaginary parts.

To the best of our knowledge, the general framework described here and due to Baladi and Vallée [9] provides the first instance of a dynamical distributional analysis. The authors apply and extend powerful tools due to Dolgopyat to dynamical systems with infinitely many branches These principles are later used in [46] to study the Gaussian behaviour of the size of the continuant "at a fraction of the depth", together with a smooth version of the bit-complexity.

4. Lattice reduction in two dimensions.

The Gauss algorithm lies in the middle of the genealogy. It inherits many formal features from the Euclid algorithm, but it is also quite different, due to the ambient topology. The first three sections 4.1, 4.2, 4.3 describe the Gauss Algorithm in its vectorial context. Then, Section 4.4 introduces the complex framework, and Section 4.5 isolates the core of the algorithm, called the `Core-Gauss` algorithm, which can be viewed as an *exact extension* of the `C-Euclid` algorithm. The sequel of the Section is then devoted to the probabilistic analysis of the `Core-Gauss` Algorithm. Probabilistic models are introduced in Section 4.6, and in particular the notion of valuation. Then Sections 4.8 and 4.9 explain how the main probabilistic objects are expressed in terms of the (quasi-inverse of the) transfer operator of the `Core-Gauss` system previously described in Section 4.7. Section 4.10 provides the analytic properties of the transfer operator, that are used in the next three sections, which describe the average-case analysis, first in the continuous model (Sections 4.11 and 4.12) and finally in the discrete model (Section 4.13).

4.1. Lattice in two dimensions.

Up to a possible isometry, a two–dimensional lattice may always be considered as a subset of $\mathbb{R}^2$. With a small abuse of language, we use the same notation for denoting a complex number $z \in \mathbb{C}$ and the vector of $\mathbb{R}^2$ whose components are $(\Re z, \Im z)$. For a complex z, we denote by $|z|$ both the modulus of the complex z and the Euclidean norm of the vector z; for two complex numbers u, v, we denote by $\langle u, v\rangle$ the scalar product between the two vectors u and v. The following relation between the two complex numbers u, v and their quotient v/u will be very useful in the sequel

$$\frac{v}{u} = \frac{\langle u, v\rangle}{|u|^2} + i\frac{\det(u,v)}{|u|^2}. \tag{14}$$

We now restrict to a pair (u,v) of $\mathbb{R}$–linearly independent elements of $\mathbb{C}$. Then, due to (14), one has $\Im(v/u) \neq 0$, and the lattice $\mathcal{L}(u,v)$ generated by the pair (u,v) is the set of elements of $\mathbb{C}$ (also called vectors) defined by

$$\mathcal{L}(u,v) = \mathbb{Z}u \oplus \mathbb{Z}v = \{au + bv; \quad a, b \in \mathbb{Z}\}.$$

In the sequel, we focus on acute bases which for which the scalar product $\langle u, v\rangle$ is positive. This is equivalent to consider bases which satisfy $\Re(v/u) \geq 0$.

Amongst all the bases of a lattice $\mathcal{L}$, some that are called reduced enjoy the property of being formed with "short" vectors. In dimension 2, the best reduced bases are *minimal* bases that satisfy optimality properties: define u to be a first minimum of a lattice $\mathcal{L}$ if it is a nonzero vector of $\mathcal{L}$ that has smallest Euclidean norm; the length of a first minimum of $\mathcal{L}$ is denoted by $\lambda_1(\mathcal{L})$. A second minimum v is any shortest vector amongst the vectors of the lattice that are linearly independent of one of the first minimums u; the Euclidean length of a second minimum is denoted by $\lambda_2(\mathcal{L})$. Then a basis is *minimal* if it comprises a first and a second minimum. For instance, the basis on the left of Figure 1 is minimal.

The following (classical) result gives a characterization of a minimal acute basis.

Let (u, v) be an acute basis. Then the conditions (a) and (b) are equivalent:

(a) the basis (u, v) is minimal;

(b) the pair (u, v) satisfies the two simultaneous inequalities:

$$(A_1): \quad \left|\frac{v}{u}\right| \geq 1, \qquad \text{and} \qquad (A_2): \quad 0 \leq \Re\left(\frac{v}{u}\right) \leq \frac{1}{2}. \tag{15}$$

Then, the angle $\theta(u, v)$ between the two vectors u and v of a minimal basis satisfies $|\theta| \in [\pi/3, \pi/2]$ and the imaginary part $y := \Im(v/u)$ satisfies $|y| \geq \sqrt{3}/2$.

4.2. **The Gaussian reduction scheme.** Here, we focus on the reduction process which deals with acute bases. The acute reduction algorithm called `A-Gauss` takes as input an arbitrary acute basis and produces as output an acute minimal basis. It aims at satisfying simultaneously the conditions (A) described in (15). The condition (A_1) is simply satisfied by an *exchange*, and the condition (A_2) is met by an *integer translation* of the longest vector v with respect to the shortest one u. This can be viewed as a "vectorial" division of v by u, which replaces v by a shortest vector amongst all the vectors of the set

$$\{\epsilon(v - qu), \qquad q \in \mathbb{Z}, \quad \epsilon = \pm 1\}.$$

The new vector $\check{v}$ is then written as

$$\check{v} := \epsilon(v - qu) \qquad \text{with} \quad q := \lfloor \tau(v, u) \rceil, \qquad \epsilon = \operatorname{sign}\left(\tau(v, u) - \lfloor \tau(v, u) \rceil\right),$$

where $\tau(v, u)$ is defined as

$$\tau(v, u) = \Re\left(\frac{v}{u}\right) = \frac{\langle u \cdot v \rangle}{|u|^2} \tag{16}$$

The new $\tau(\check{v}, u)$ satisfies $\tau(\check{v}, u) \in [0, 1/2]$. Then, if $\check{v}$ is longest than u, the algorithm stops on an acute minimal basis. If not, the algorithm performs an exchange and continues.

`A-Gauss`(u, v)
Input. An acute basis (u, v) of $\mathbb{C}$
with $|v| \leq |u|$, $\tau(v, u) \in [0, 1/2]$ and $u \neq \lambda v, \lambda \in \mathbb{R}$.
Output. An acute minimal basis (u, v) of $\mathcal{L}(u, v)$ with $|v| \geq |u|$

```
While |v| < |u| do
    (u, v) := (v, u);
    q := ⌊τ(v, u)⌉ ; ε := sign (τ(v, u) − ⌊τ(v, u)⌉);
    v := ε(v − qu);
```

FIGURE 4. Description of the `A-Gauss` Algorithm

On the input pair $(u, v) = (u_0, u_1)$, the Gauss Algorithm computes a sequence of vectors $(u_i)_{i \in [0..p]}$ defined by the relations

$$u_{i+1} = \epsilon_i(u_{i-1} - q_i\, u_i) \quad \text{with} \quad m_i := \lfloor \tau(u_{i-1}, u_i) \rceil, \qquad \epsilon_i = \operatorname{sign}\left(\tau(u_{i-1}, u_i) - \lfloor \tau(u_{i-1}, u_i) \rceil\right). \tag{17}$$

Here, each quotient m_i is a positive integer, $p \equiv P(u,v)$ denotes the number of iterations, and the final pair (u_p, u_{p+1}) satisfies Conditions (A) of Eq (15).
Unlike the `C-Euclid` Algorithm, the `A-Gauss` algorithm *always terminates* when the input (u,v) is made with two non colinear vectors. The fact that the continuous version of the algorithm terminates is a great difference with the `C-Euclid` Algorithm. It will make possible to see the discrete data as embedded inside the continuous data (See Section 4.6).

4.3. **Main parameters of interest.** The length of an input pair $(u,v) \in \mathbb{Z}[i]\times\mathbb{Z}[i]$ is $\max(|u|^2,|v|^2) = |u|^2$ and its size is $\ell(u,v) := \max\{\ell(|u|^2),\ell(|v|^2)\} = \ell|u|^2 \approx \lg|u|^2$ where $\ell(x)$ is the binary length of the integer x. This is also the size of the Gram matrix $G(u,v)$, defined as

$$G(u,v) = \begin{pmatrix} |u|^2 & (u\cdot v) \\ (u\cdot v) & |v|^2 \end{pmatrix}.$$

In the following, we will consider subsets which gather all the (valid) inputs of length N (or size M with $M = \log N$), and introduce

$$\Omega_N := \{(u,v) \in \mathbb{Z}[i]\times\mathbb{Z}[i] \mid |v|^2 \le |u|^2 \le N,\ \tau(v,u) \in [0,1/2]\}$$

First, the `A-Gauss` algorithm is proven to always terminate on an input of Ω_N after a polynomial number of steps. Unlike the `C-Euclid` Algorithm, the first proof due to Lagarias [38] is not completely straightforward, and was followed by a more precise study due to Vallée [69]. We return to this worst-case analysis in Section 4.6. But we now focus on the average-case analysis, where the set Ω_N will be endowed with the uniform probability $\mathbb{P}_N$, and the main parameters become random variables defined on these sets.

All the computations of the Gauss algorithm are done on the Gram matrices $G(u_i,u_{i+1})$ of the pair (u_i,u_{i+1}). The *initialization* of the Gauss algorithm *computes* the Gram Matrix of the initial basis, with the three scalar products, which takes a *quadratic*[9] time $\Theta(M^2)$ with respect to the length of the input $\ell(u,v)$. After this, all the computations of the *central part* of the algorithm *are directly done* on these matrices; more precisely, each step of the process is a Euclidean division between the two coefficients of the first line of the Gram matrix $G(u_i,u_{i-1})$ of the pair (u_i,u_{i-1}) for obtaining the quotient $q_i = (m_i,\epsilon_i)$, followed with the computation of the new coefficients of the Gram matrix $G(u_{i+1},u_i)$, namely

$$|u_{i+1}|^2 := |u_{i-1}|^2 - 2m_i\,(u_i\cdot u_{i-1}) + m_i^2|u_i|^2, \qquad (u_{i+1}\cdot u_i) := m_i\,|u_i|^2 - (u_{i-1}\cdot u_i).$$

Then the cost of the i-th step is closely related to to $\ell(q_i)\cdot\log(|u_{i-1}|^2)$, and the bit-complexity of the central part of the Gauss Algorithm, one of the main parameters of interest, is closely related to

$$B(u,v) = \sum_{i=1}^{P(u,v)} \ell(m_i)\cdot\log(|u_{i-1}|^2), \tag{18}$$

where $P(u,v)$ is the number of iterations of the Gauss Algorithm. The cost B is expressed with two other costs, the quotient bit-cost $Q(u,v)$ and the cost $D(u,v)$ defined as

$$Q(u,v) = \sum_{i=1}^{P(u,v)} \ell(q_i), \qquad D(u,v) := \sum_{i=1}^{P(u,v)} \ell(m_i)\log\left|\frac{u_{i-1}}{u_0}\right|^2, \tag{19}$$

as

$$B(u,v) = Q(u,v)\,\log|u|^2 + D(u,v) \tag{20}$$

We are then led to study two types of parameters, that may be of independent interest:
(a) The additive costs, which provide a generalization of costs P and Q. They are defined as the sum of elementary costs, which only depend on the quotients $q_i := (M_I,\epsilon_i)$. More precisely, from a positive

[9] We consider the naive multiplication between integers of size M, whose bit-complexity is $O(M^2)$.

elementary cost c defined on each pair $q = (m, \epsilon)$ we consider the total cost on the input (u, v) defined as

$$C_{(c)}(u, v) = \sum_{i=1}^{P(u,v)} c(q_i)\,. \tag{21}$$

When the elementary cost c satisfies $c(q) = O(\log m)$, the cost C is said to be of moderate growth.
(b) The sequence of the i-th length decreases d_i for $i \in [1..p]$ (with $p := P(u, v)$]) and the total length decrease $d := d_p$, defined as

$$d_i := \left|\frac{u_i}{u_0}\right|^2, \qquad d := \left|\frac{u_p}{u_0}\right|^2. \tag{22}$$

4.4. **The complex framework.** We now return to a generic input made of two (non colinear) complex numbers (u, v). Many structural characteristics of lattices and bases are invariant under linear transformations –similarity transformations in geometric terms– of the form $S_\lambda : u \mapsto \lambda u$ with $\lambda \in \mathbb{C} \setminus \{0\}$.

(a) A first instance is the execution of the Gauss algorithm itself: it should be observed that translations performed by the Gauss algorithms only depend on the quantity $\tau(v, u)$ defined in (16), which equals $\Re(v/u)$. Furthermore, exchanges depend on the value of $|v/u|$. Then, if u_i is the sequence computed by the algorithm on the input (u, v), defined in (17), the sequence of vectors computed on an input pair $S_\lambda(u, v)$ coincides with the sequence $S_\lambda(u_i)$. This makes it possible to give a formulation of the Gauss algorithm entirely in terms of complex numbers.

(b) A second instance is the characterization of minimal bases given in (15) that only depends on the ratio $z = v/u$.

(c) A third instance are the main parameters of interest $X \in \{Q, D, d_i\}$ defined in Section 4.3 that satisfy $X(\lambda u, \lambda v) = X(u, v)$. Their complex version defined as $X(z) := X(1, z)$ satisfies $X(\lambda z) = X(z)$.

It is thus natural to consider lattice bases taken up to equivalence under similarity, and restrict attention to lattice bases of the form $(1, z)$. We denote by $L(z)$ the lattice $\mathcal{L}(1, z)$. In this context, the geometric transformation effected by each step of the algorithm consists of an inversion-symmetry $S : z \mapsto 1/z$, followed by a translation $z \mapsto T^{-q}z$ with $T(z) = z + 1$, and a possible sign change $J : z \mapsto -z$. The right half plane

$$\{z \in \mathbb{C};\quad \Re(z) \geq 0,\, \Im(z) \neq 0\}$$

plays a central rôle. The `A-Gauss` algorithm brings z into the vertical strip

$$\widetilde{\mathcal{B}} := \left\{ z \in \mathbb{C};\quad \Im(z) \neq 0,\quad 0 \leq \Re(z) \leq \frac{1}{2} \right\}, \tag{23}$$

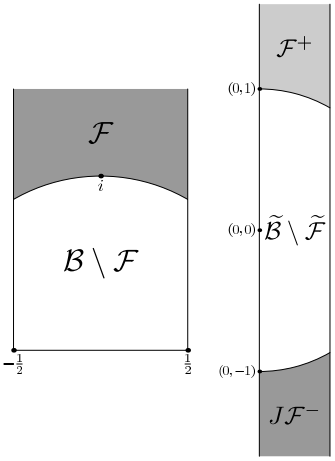
and stops as soon as z belongs to the domain $\widetilde{\mathcal{F}}$

$$\widetilde{\mathcal{F}} = \left\{ z \in \mathbb{C};\quad |z| \geq 1,\quad 0 \leq \Re(z) \leq \frac{1}{2} \right\}. \tag{24}$$

The sets $\widetilde{\mathcal{B}}$ and $\widetilde{\mathcal{F}}$ are represented in Figure 5 and compared to their usual versions.
While the complex z does not belongs to $\widetilde{\mathcal{F}}$ the algorithm reduces to the iteration of the mapping

$$U(z) = \epsilon\left(\frac{1}{z}\right)\left(\frac{1}{z} - \left\lfloor \Re\left(\frac{1}{z}\right) \right\rceil\right) \qquad \text{with} \quad \epsilon(z) := \operatorname{sign}(\Re(z) - \lfloor \Re(z) \rceil)\,. \tag{25}$$

From the definition of the two shifts described in (3) and (25), it is clear that the `A-Gauss` algorithm can be viewed as a formal extension of the `C-Euclid` algorithm. However, this not so simple, and the relations (resemblances and differences) between the two algorithms are made precise in Figure 6.



$$\mathcal{B} := \{z; \quad \Im(z) \neq 0 \quad |\Re(z)| \leq 1/2\}$$
$$\mathcal{F} := \{z; \quad |\Re(z)| \leq 1/2, \ |z| \geq 1\}$$

$$\mathcal{B}^\epsilon := \{z \in \mathcal{B}, \quad \operatorname{sign} \Re(z) = \epsilon\}$$
$$\mathcal{F}^\epsilon := \{z \in \mathcal{F}, \quad \operatorname{sign} \Re(z) = \epsilon\}$$

$$\widetilde{\mathcal{B}} := \mathcal{B}^+ \bigcup J\mathcal{B}^-, \ \widetilde{\mathcal{F}} := \mathcal{F}^+ \bigcup J\mathcal{F}^-$$

FIGURE 5. The sets $\widetilde{\mathcal{B}}$ and $\widetilde{\mathcal{F}}$ compared to the "classical" sets $\mathcal{B}$ and $\mathcal{F}$.

Euclid's algorithm	Gauss' algorithm
Division between real numbers $v = mu + \epsilon r$ with $m = \left\lfloor \frac{u}{v} \right\rceil$ and $\frac{r}{v} \leq \frac{1}{2}$	Division between complex vectors $v = mu + \epsilon r$ with $m = \left\lfloor \Re\left(\frac{u}{v}\right) \right\rceil$ and $\Re\left(\frac{r}{v}\right) \leq \frac{1}{2}$
Division + exchange $(v,u) \to (r,v)$ "read" on $x = v/u$ $U(x) = \epsilon\left(\frac{1}{x}\right)\left(\frac{1}{x} - \left\lfloor \frac{1}{x} \right\rceil\right)$	Division + exchange $(v,u) \to (r,v)$ "read" on $z = v/u$ $U(z) = \epsilon\left(\frac{1}{z}\right)\left(\frac{1}{z} - \left\lfloor \Re\left(\frac{1}{z}\right) \right\rceil\right)$
Stopping condition: $x = 0$	Stopping condition: $z \in \widetilde{\mathcal{F}}$

FIGURE 6. Comparison between the algorithms `C-Euclid` and `A-Gauss`.

Remark that the (continuous) `C-Euclid` Algorithm never stops, except on rational entries. whereas the (continuous) version of the `A-Gauss` Algorithm always stops, except for irrational flat bases z for which $\Im z = 0$ and $\Re z \notin \mathbb{Q}$. The main difference is due to the nature of the two "holes":

– The hole for the `C-Euclid` Algorithm is $\{0\}$, that is of zero measure
– The hole for the `A-Gauss` Algorithm is $\widetilde{\mathcal{F}}$, that is a fundamental domain

4.5. **The `Core-Gauss` Algorithm.** The disk $\mathcal{D}$ of diameter $\mathcal{I} = [0, 1/2]$ plays a special rôle. Figure 7 shows that the domain $\widetilde{\mathcal{B}} \setminus \mathcal{D}$ decomposes as the union of six transforms of the fundamental domain $\widetilde{\mathcal{F}}$, namely

$$\widetilde{\mathcal{B}} \setminus \mathcal{D} = \bigcup_{h \in \mathcal{K}} h(\widetilde{\mathcal{F}}) \qquad \text{with} \quad \mathcal{K} := \{I, S, STJ, ST, ST^2J, ST^2JS\}. \tag{26}$$

Then, there are two main parts in the execution of the `A-Gauss` Algorithm, according to the position of the current complex z_i with respect to the disk $\mathcal{D}$ of diameter $\mathcal{I}$ whose alternative equation is

$$\mathcal{D} := \left\{ z; \qquad \Re\left(\frac{1}{z}\right) \geq 2 \right\}.$$

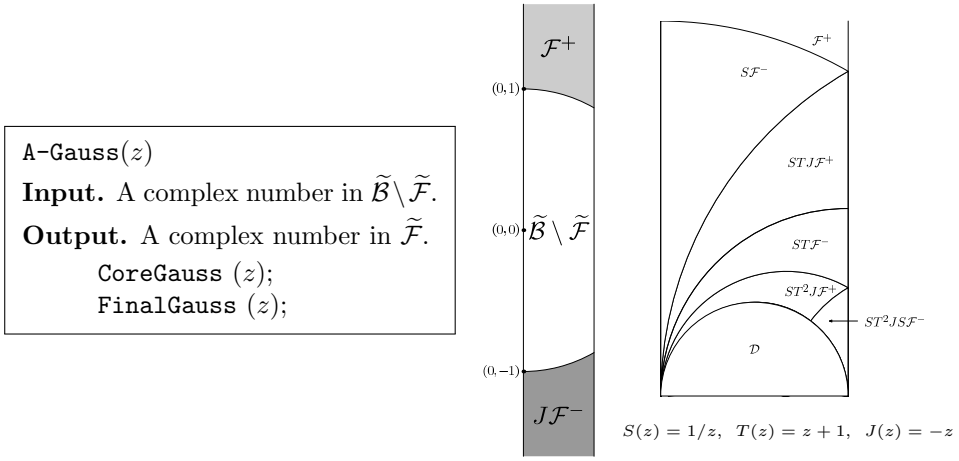


FIGURE 7. The decomposition of the A-Gauss Algorithm into two parts: its core part (the Core-Gauss Algorithm) and its final part (the Final-Gauss Algorithm). On the right, the partition of $\widetilde{\mathcal{B}}\setminus\mathcal{D}$ into six domains $K\widetilde{\mathcal{F}}$ where K belongs to the set $\mathcal{K}$ defined in (26)

While z_i belongs to $\mathcal{D}$, the quotient (m_i,ϵ_i) satisfies $(m_i,\epsilon_i)\geq(2,+1)$ (wrt the lexicographic order), and the algorithm uses at each step the set $\mathcal{H}$, that is already central in the C-Euclid Algorithm, namely

$$\mathcal{H}:=\{h_{\langle m,\epsilon\rangle};\quad (m,\epsilon)\geq(2,+1)\}\,,$$

so that $\mathcal{D}$ can be written as

$$\mathcal{D}=\bigcup_{h\in\mathcal{H}^+}h(\widetilde{\mathcal{B}}\setminus\mathcal{D})\qquad\text{with}\quad\mathcal{H}^+:=\sum_{k\geq1}\mathcal{H}^k. \tag{27}$$

The part of the A-Gauss algorithm performed when z_i belongs to $\mathcal{D}$ is called the Core-Gauss algorithm. As soon as z_i does not any longer belong to $\mathcal{D}$, there remains at most two iterations that constitutes the Final-Gauss algorithm and uses the set $\mathcal{K}$ of LFT's described in (26).

Finally, we have proven the decomposition of the A-Gauss Algorithm:

A-Gauss = Core-Gauss followed with Final-Gauss (at most 2 iterations).

The Core-Gauss algorithm has a nice structure since it uses at each step the same set $\mathcal{H}$. This set is exactly the set of LFT's which is used by the C-Euclid Algorithm, closely related to the dynamical system C-Euclid defined in the previous Section. Then, the Core-Gauss algorithm is just a lifting of this C-Euclid Algorithm, whereas the final steps of the A-Gauss algorithm use different LFT's, and are not similar to a lifting of a Euclidean Algorithm. This is why the Core-Gauss algorithm is interesting to study.

The complex numbers which intervene in the Core-Gauss algorithm on the input $z_0=u_1/u_0$ are related to the vectors (u_i) defined in (17) via the relation $z_i=u_{i+1}/u_i$. They are computed by the relation

$z_{i+1} := U(z_i)$, so that

$$z_{i-1} = h_{\langle m_i,\epsilon_i\rangle}(z_i) \qquad \text{with} \quad h_{\langle m,\epsilon\rangle}(z) := \frac{1}{m+\epsilon z}\,.$$

This creates a continued fraction expansion for the initial complex z_0, of the form

$$z_0 = \cfrac{1}{m_1 + \cfrac{\epsilon_1}{m_2 + \cfrac{\epsilon_2}{\ddots \cfrac{}{m_p + \epsilon_p z_p}}}} = h(z_p) \qquad \text{with} \quad h := h_{\langle m_1,\epsilon_1\rangle} \circ h_{\langle m_2,\epsilon_2\rangle} \circ \dots h_{\langle m_p,\epsilon_p\rangle}.$$

More generally, the i-th complex number z_i satisfies

$$z_0 = h_i(z_i) \qquad \text{with} \quad h_i := h_{\langle m_1,\epsilon_1\rangle} \circ h_{\langle m_2,\epsilon_2\rangle} \circ \dots h_{\langle m_i,\epsilon_i\rangle}. \tag{28}$$

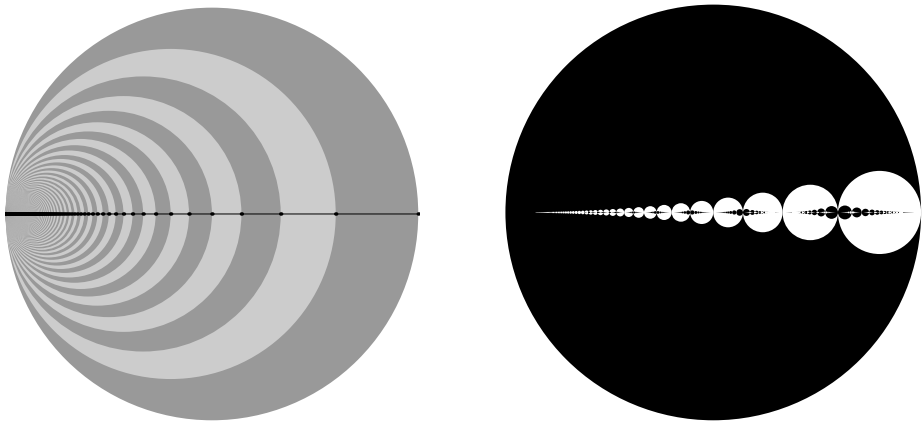


FIGURE 8. On the left, the topological partition of the `Core-Gauss` dynamical system. The intersection of this partition with the real axis gives rise to the topological partition of the `C-Euclid` dynamical system. On the right, the domains $[R = k]$ alternatively in black and white. The figure suggests that reduction of almost-collinear bases is likely to require a large number of iterations.

Consider the number of iterations R of the `Core-Gauss` algorithm. Then, the domain $[R \geq k+1]$ gathers the complex numbers z for which the transforms $U^k(z)$ are yet in $\mathcal{D}$. Such a domain admits a nice characterization, as a union of disjoint disks, namely

$$[R \geq k+1] = U^{-k}(\mathcal{D}) = \bigcup_{h\in\mathcal{H}^k} h(\mathcal{D}), \tag{29}$$

which is represented in Figure 8. The disk $h(\mathcal{D})$ for $h \in \mathcal{H}^+$ is the disk whose diameter is the interval $h(\mathcal{I})$. Inside the `C-Euclid` dynamical system, the interval $h(\mathcal{I})$ (relative to a LFT $h \in \mathcal{H}^k$) is called a fundamental interval (or a cylinder) of depth k: it gathers all the real numbers of the interval $\mathcal{I}$ which have the same continued fraction expansion of depth k. This is why the disk $h(\mathcal{D})$ is called a fundamental disk.

We now focus (just for a moment) on the worst-case complexity of the `Core-Gauss` algorithm on the set $\mathcal{D}$. The maximum number of iterations arises (in the same vein as in the `C-Euclid` algorithm) in relation with the sequence

$$A_0 = A_1 = 1, \quad A_{k+1} = 2A_k + A_{k-1}, \quad (k \geq 1), \tag{30}$$

as we shortly explain. With (29), we remark that, for a given depth k, the largest such disk $h(\mathcal{D})$ is obtained when all the quotients (m, ϵ) are the smallest ones, i.e., when all $(m, \epsilon) = (2, +1)$. In this case, the coefficients (c, d) of the LFT h are the terms A_k, A_{k+1} of the sequence defined in (30) which satisfy $A_k \geq (1+\sqrt{2})^{k-2}$. Then, the largest such disk has a radius at most equal to $(1/2)(1+\sqrt{2})^{1-2k}$. This proves the inclusion

$$[R \geq k+1] \subset \left\{ z; \quad |\Im(z)| \leq \frac{1}{2} \left(\frac{1}{1+\sqrt{2}} \right)^{2k-1} \right\}, \tag{31}$$

that entails the following worst-case bound that holds for *any complex number* $z \notin \mathbb{R}$

$$R(z) \leq \frac{1}{2} \log_{1+\sqrt{2}} \frac{1}{|\Im z|}.$$

This shows that the `Core-Gauss` and thus the `A-Gauss` Algorithm itself always terminates on a non-real complex number z. This gives an alternative proof of the result of Vallée [69].

Figure 8 also shows in a striking way the efficiency of the algorithm, and asks natural questions: Is it possible to estimate the probability of the event $[R \geq k+1]$? Is it true that it is geometrically decreasing? With which ratio? We return to these questions later on.

4.6. **Probabilistic models.** We now begin our probabilistic studies. Since we focus on the invariance of algorithm executions under similarity transformations, we consider densities on pairs of vectors (u, v) which only depend on the ratio $z = v/u$. There are two views on these functions, as functions of the complex variable $(z, \overline{z}) \mapsto F(z, \overline{z})$ or as functions defined on $\mathbb{R}^2$, of the form $(x, y) \mapsto f(x, y)$. The model $\mathcal{M}\langle f \rangle$ considers as the input set the disk $\mathcal{D}$ endowed with a density f.

Discrete models. We fix an integer N and we consider pairs (u, v) with a first vector u of the form $u = (N, 0)$. The discrete input subset $\mathcal{D}_N$ is defined as

$$\mathcal{D}_N := \left\{ z = \frac{v}{u}; \quad u = (N, 0), v = (a, b), \quad (a, b, N) \in \mathbb{N}^3, \quad z \in \mathcal{D} \right\}.$$

The discrete probabilistic model $\mathcal{M}_N\langle f \rangle$ is defined as the restriction to $\mathcal{D}_N$ of the continuous model defined on $\mathcal{D}$ via the density f. More precisely, with a given density f on $\mathcal{D}$, we associate its restriction on $\mathcal{D}_N$. Normalized by the cardinality $|\mathcal{D}_N|$, this gives rise to a density $f_{\langle N \rangle}$ on $\mathcal{D}_N$, that we extend on $\mathcal{D}$ as follows:

$$f_{\langle N \rangle}(x, y) := f_{\langle N \rangle}(\omega) \quad \text{when } (x, y) \text{ belongs to the square of center } \omega \in \mathcal{D}_N \text{ and edge } 1/N.$$

We obtain, in such a way, a family of functions $f_{\langle N \rangle}$ defined on $\mathcal{D}$, and we use the Gauss principle which compares the number of integer points in a domain $\mathcal{C}$ with the volume of $\mathcal{C}$ (see Section 4.13). When the integer N tends to ∞, this discrete model $\mathcal{M}_N\langle f \rangle := \mathcal{M}\langle f_{\langle N \rangle} \rangle$ "tends" to the continuous model $\mathcal{M}\langle f \rangle$ relative to f, as we will see.

We first remark that the inclusion (31) also entails the worst-case bound

$$\max\{R(u, v) \mid (u, v) \in \mathcal{D}_N\} = \log_{1+\sqrt{2}} N + O(1).$$

This proves that the `Core-Gauss` algorithm and thus the `A-Gauss` algorithm performs a linear number of iterations, exactly of the same type as the `Euclid` Algorithm.

The model with valuation. It is clear that the two variables $x := \Re z, y := \Im z$ do not play a role of the same importance. The main actor is the imaginary part y, and the algorithm aims increasing it. The real part x plays an auxilliary role, and, during the execution of the algorithm, x remains bounded, whereas y does not. This is why we now present a family of densities which only depend on y: the density f_r *of valuation* r (with $r > 0$) is proportional to $|y|^r$, namely

$$f_r(x, y) = \frac{1}{A(r)} |y|^{r-1} \qquad \text{with} \quad A(r) = \iint_{\mathcal{D}} |y|^{r-1} dx dy = \frac{1}{2r+2} \frac{1}{2^{r+1}} \frac{[\Gamma(r/2)]^2}{\Gamma(r)} \tag{32}$$

When r tends to 0, the density is more and more concentrated near the real axis, where y is small. Inputs with a small imaginary part represent (informally speaking) hard instances for reducing the lattice: indeed, such a small y is related to a small modulus $|z|$ or a small angle $|\theta|$, whereas the output $\widehat{z}$ corresponds to a modulus $|\widehat{z}| \geq 1$ and an angle $|\widehat{\theta}| \geq \pi/3$. We will see in the sequel that the valuation r is indeed a good tool for quantifying the difficulty of the input instances. When $r = 1$, we recover the uniform distribution on $\mathcal{D}$ with $A(1) = \pi/16$ and we observe $A(r) \sim 1/r$ for $r \to 0$.

The (continuous) model relative to the density of valuation r is denoted with an index of the form (r) and is written as $\mathcal{M}_{(r)}$. The discrete models of valuation r are defined by two indices, the integer length N and the index of the valuation, and are written as $\mathcal{M}_{(r,N)}$.

What can be expected about the probabilistic behavior of the `Core-Gauss` Algorithm? On one hand, there is a strong formal similarity between the two algorithms, since the `Core-Gauss` Algorithm can be viewed as a lifting of the `C-Euclid` Algorithm. On the other hand, important differences appear when we consider algorithms: the `C-Euclid` algorithm never terminates, except on rational inputs which fall in the hole $\{0\}$, whereas the `Core-Gauss` Algorithm always terminates, except for irrational real inputs. We will see that the model of valuation r is an interesting tool for explaining the "transition" between the behaviour of the two algorithms.

4.7. **Transfer operators.** For $h \in \mathcal{H}^\star$, we consider the function $\underline{h}$ of two real variables which is induced by the map $h : \mathbb{C} \to \mathbb{C}$. It is defined as $(x, y) \mapsto \underline{h}(x, y) = (\Re h(x + iy), \Im h(x + iy))$. It is conjugated to the map (h, h) defined on $\mathbb{C}^2$ by the relation $(u, v) \mapsto (h(u), h(v))$ via the map Φ, namely $\underline{h} = \Phi^{-1} \circ (h, h) \circ \Phi$, where mappings Φ, Φ^{-1} are linear mappings $\mathbb{C}^2 \to \mathbb{C}^2$ defined as

$$\Phi(x, y) = (z = x + iy, \overline{z} = x - iy), \qquad \Phi^{-1}(z, \overline{z}) = \left(\frac{z + \overline{z}}{2}, \frac{z + \overline{z}}{2i} \right).$$

Since Φ and Φ^{-1} are linear mappings, and h has real coefficients, the Jacobian $J(\underline{h})$ satisfies

$$J(\underline{h})(x, y) = |h'(z) \cdot h'(\overline{z})| = |h'(z)|^2 = \check{h}(z) \cdot \check{h}(\overline{z}), \tag{33}$$

where $\check{h}$ is the analytic extension of $x \mapsto |h'(x)|$ already used in Section 3. Consider now a density on $\mathcal{D}$. Viewed as a function of real variables, it is denoted as $f : (x, y) \mapsto f(x, y)$ and, viewed as a function of the complex variable, it is denoted as $F(z, \overline{z})$. The density transformer related to the `Core-Gauss` system is then defined by

$$f(x, y) \mapsto \sum_{h \in \mathcal{H}} |h'(x + iy)|^2 \, f(\underline{h}(x, y)) \qquad \text{or} \quad F(z, \overline{z}) \mapsto \sum_{h \in \mathcal{H}} |h'(z)| \, |h'(\overline{z})| \, F(h(z), h(\overline{z})).$$

It is then convenient to introduce a more general operator which depends on a parameter s and acts on analytic functions of two variables, namely

$$\begin{aligned} \mathbb{H}_s[F](z, u) &= \sum_{h \in \mathcal{H}} \check{h}(z)^{s/2} \, \check{h}(u)^{s/2} \, F(h(z), h(u)) \\ &= \sum_{(m,\epsilon) \geq (2,1)} \left(\frac{1}{m + \epsilon z} \right)^s \left(\frac{1}{m + \epsilon u} \right)^s F\left(\frac{1}{m + \epsilon z}, \frac{1}{m + \epsilon u} \right). \end{aligned} \tag{34}$$

With (33), the density transformer exactly coincides with $F(z, \overline{z}) \mapsto \mathbb{H}_2[F](z, \overline{z})$, and the operator $\mathbb{H}_s$ also provides an extension of the density transformer $\mathbf{H}_s$ related to the `C-Euclid` system, via the "diagonal" relation

$$\mathbb{H}_s[F](x, x) = \mathbf{H}_s[\operatorname{diag} F](x), \qquad \text{where } \operatorname{diag} F \text{ is defined as } \operatorname{diag} F(x) = F(x, x). \tag{35}$$

When applied to the density F_r of valuation r, the equality $\Im h(z) = \Im z \cdot (\det h) \cdot |h'(z)|$ proves the relation

$$\mathbb{H}_s[F_r](z, \overline{z}) = \sum_{h \in \mathcal{H}} |h'(z)|^s |\Im(h(z))|^{r-1} = |\Im(z)|^{r-1} \sum_{h \in \mathcal{H}} |h'(z)|^{s+r-1} = |\Im(z)|^{r-1} \mathbb{H}_{s+r-1}[1](z, \overline{z}). \tag{36}$$

In particular, when r tends to 0, the density F_r is concentrated near the real axis; moreover, if $s = 2$, the parameter $s+r-1 = 1+r$ tends to 1, and the operator $\mathbb{H}_{s+r-1}$ "tends to" the operator $\mathbb{H}_1$ which is itself an extension of the density transformer $\mathbf{H}_1$ of the `C-Euclid` algorithhm. This explains the interest of the notion of valuation to study the transition between the `A-Gauss` algorithm and the `C-Euclid` algorithm. We return to this fact later on.

4.8. **Output and conditional densities.** There are two kinds of interesting densities related to the execution of the algorithm, the output density and the conditional densities.

Output density. We recall that the set $\mathcal{H}^+ = \cup_{k>0}\mathcal{H}^k$ is the set of the transformations describing the whole executions of the `Core-Gauss` Algorithm. Then, in the same vein as Section 2, we introduce the transfer operators $\mathbb{G}_s^+$ relative to $\mathcal{H}^+$ together with $\mathbb{G}_s$ relative to $\mathcal{H}$

$$\mathbb{G}_s^+ = \mathbb{H}_s \circ (I - \mathbb{H}_s)^{-1} = (I - \mathbb{H}_s)^{-1} - I, \qquad \mathbb{G}_s = (I - \mathbb{H}_s)^{-1}\,, \tag{37}$$

and we describe the output density:

Proposition 2. *Consider the* `Core-Gauss` *algorithm with the complex version F of its input density on $\mathcal{D}$. Then, the complex version of the output density is expressed as $\widehat{F} = \mathbb{G}_2^+[F]$. When the input density is the density F_r of valuation r, then the ouput density is expressed as $\widehat{F}_r(z,\overline{z}) = |y|^{r-1}\mathbb{G}_{1+r}^+[1](z,\overline{z})$.*

The Hurwitz characterization gives rise to a nice expression for the output density $\widehat{F}_r$

$$\widehat{F}_r(z,\overline{z}) = \frac{1}{A(r)}\frac{1}{\zeta(2r+2)} \sum_{\substack{c,d\geq 1\\ d\phi<c<d\phi^2}} \frac{y^{r-1}}{|cz+d|^{2r+2}}.$$

Conditional densities. We have already mentioned the importance of the invariant density in the `C-Euclid` algorithm. No such invariant measure can exist here as the reduction algorithm terminates. However, a rôle quite similar to the invariant density is played by a function that describes the limit distribution of successive transforms of the input as the reduction algorithms proceeds.

We begin with an input z_0 which is distributed with a density $f(x,y) = F(z,\overline{z})$ inside the disk $\mathcal{D}$. *Assume* now that the algorithm performs *at least $k+1$ iterations*. Then the kth iterate z_k is yet an element of $\mathcal{D}$. A natural question is to determine its distribution inside $\mathcal{D}$. We call it the *conditional* density at depth k, and we denote it by $f^{[k]}(x,y) = F^{[k]}(z,\overline{z})$. The definition of the density transform $\mathbb{H}_2$ leads to an alternative expression for $F^{[k]}(z,\overline{z})$, namely

$$F^{[k]}(z,\overline{z}) = \frac{1}{\mathbb{P}_{\langle F\rangle}[R\geq k+1]}\,\mathbb{H}_2^k[F](z,\overline{z})\,,$$

the normalization factor being the probability that the algorithm performs at least $k+1$ iterations, which is the measure (relative to density F) of the set $[R\geq k+1]$ described in (29) which admits itself an alternative expression

$$\mathbb{P}_{\langle F\rangle}[R\geq k+1] = \iint_{\mathcal{D}} \mathbb{H}_2^k[F](z,\overline{z})dxdy\,.$$

Proposition 3. *Consider the* `Core-Gauss` *algorithm with the complex version F of its input density on $\mathcal{D}$. Then, the complex version of the k-th conditional density is expressed in terms of the k-th iterate of the density transformer $\mathbb{H}_2$ as*

$$F^{[k]}(z,\overline{z}) = \frac{1}{\iint_{\mathcal{D}}\mathbb{H}_2^k[F](z,\overline{z})dxdy}\,\mathbb{H}_2^k[F](z,\overline{z})\,,$$

When the input density is the density F_r of valuation r, then the conditional density is expressed in terms of the operator $\mathbb{H}_{1+r}$ as

$$F_r^{[k]}(z,\overline{z}) = \frac{1}{\iint_{\mathcal{D}}|y|^{r-1}\mathbb{H}_{1+r}^k[F](z,\overline{z})dxdy}\,|y|^{r-1}\mathbb{H}_{1+r}^k[F](z,\overline{z})\,.$$

As we will see later, this conditional density is closely related to the dominant eigenfunction of the operator $\mathbb{H}_{1+r}$.

We now begin the probabilistic analysis of the `Core-Gauss` algorithm. But we insist on a great difference with the analysis of the `C-Euclid` algorithm. This last algorithm admits a continuous version which never terminates (except on its discrete inputs). It makes the transfer *continuous* $\leftrightarrow$ *discrete* more difficult, and generating functions were an (indirect) tool which operate this transfer. Here, the situation is more simpler, as the continuous version of the `A-Gauss` algorithm always terminates. This allows a more direct transfer, and we will see the discrete data as *embedded* inside the continuous data, as explained in Section 4.6. We thus begin with the analysis of the `Core-Gauss` in the continuous model.

4.9. **Generating operators for costs C and D.** As in the study of the `C-Euclid` algorithm, we now modify the transfer operator $\mathbb{H}_s$ defined in (34) in such a way that it becomes a "generating operator" for costs of interest. In fact, these operators generate themselves ... the main objects needed in our probabilistic study.

Additive costs. An additive cost $C_{(c)}$, defined in (21), is related to an elementary cost c defined on quotients q. In the same vein as in the Euclid case, such a cost can be defined on $\mathcal{H}$ and linearly extended to the total set $\mathcal{H}^\star$. This gives rise to another definition for the complex version of cost defined by $C(z) := C(1, z)$. If an input $z \in \mathcal{D}$ leads to an output $\widehat{z} \in \widetilde{\mathcal{B}} \setminus \mathcal{D}$ by using the LFT $h \in \mathcal{H}^+$ with $z = h(\widehat{z})$, then $C(z)$ equals $c(h)$.

It is natural to add a new parameter w inside the transfer operator $\mathbb{H}_s$ for "marking" the cost, and we consider the two-parameters operator

$$\mathbb{H}_{s,w,(c)}[F](z,u) = \sum_{h\in\mathcal{H}} e^{wc(h)}\, \check{h}(z)^{s/2}\, \check{h}(u)^{s/2}\, F(h(z), h(u)) \tag{38}$$

together with its associated quasi-inverse. As for the `C-Euclid` algorithm, the operator $\mathbb{H}_{s,w,(c)}$ generates the moment generating function of the cost $C_{(c)}$, as we will see now. The moment generating function $\mathbb{E}_{\langle f\rangle}(\exp[wC_{(c)}])$ is defined as

$$\mathbb{E}_{\langle f\rangle}(\exp[wC_{(c)}]) := \sum_{h\in\mathcal{H}^+} \exp[wc(h)] \cdot \mathbb{P}_{\langle f\rangle}[C(z) = c(h)] = \sum_{h\in\mathcal{H}^+} \exp[wc(h)] \iint_{h(\widetilde{\mathcal{B}}\setminus\mathcal{D})} f(x,y)dxdy.$$

Using a change of variables and the expression of the Jacobian, leads to

$$\mathbb{E}_{\langle f\rangle}(\exp[wC_{(c)}]) = \sum_{h\in\mathcal{H}^+} \exp[wc(h)] \iint_{\widetilde{\mathcal{B}}\setminus\mathcal{D}} |h'(z)|^2 F(h(z), h(\overline{z}))dxdy = \iint_{\widetilde{\mathcal{B}}\setminus\mathcal{D}} \mathbb{G}^+_{2,w,(c)}[F](z,\overline{z})dxdy.$$

Now, when the density F is of valuation r, Relation (36) leads to the expression

$$\mathbb{E}_{(r)}(\exp[wC_{(c)}]) = \iint_{\widetilde{\mathcal{B}}\setminus\mathcal{D}} y^{r-1}\ \mathbb{G}^+_{1+r,w,(c)}[1](z,\overline{z})dxdy. \tag{39}$$

The expectation $\mathbb{E}_{\langle f\rangle}[C_{(c)}]$ is just obtained by taking the derivative with respect to w (at $w = 0$). There appears the operator

$$\mathbb{G}_{s,C} := \mathbb{G}_s \circ \mathbb{H}_s^{[c]} \circ \mathbb{G}_s \tag{40}$$

This provides an alternative expression for the expectation of any additive cost:

$$\mathbb{E}_{\langle f\rangle}[C_{(c)}] = \iint_{\widetilde{\mathcal{B}}\setminus\mathcal{D}} \mathbb{G}_{2,C}[F](z,\overline{z})dxdy = \iint_{\widetilde{\mathcal{B}}\setminus\mathcal{D}} |y|^{r-1}\, \mathbb{G}_{1+r,C}[1](z,\overline{z})dxdy, \tag{41}$$

the last equality holding for a density F of valuation r.

Cost D. In the same vein as in Section 3.4, the i-th length decrease d_i defined in (22) can be expressed with the derivative of the LFT $g_i := h_i^{-1}$ (with h_i defined in (28)) as

$$\frac{|u_i|^2}{|u_0|^2} = \frac{1}{|g_i'(z)|} = |c_i z - a_i|^2 \quad \text{so that} \quad 2\log\left(\frac{|u_i|}{|u_0|}\right) = -\log|g_i'(z)| = -\log|c_i z - a_i|^2,$$

where a_i, c_i are coefficients of the LFT h_i. Finally, the complex versions of cost D is

$$D(z) = \sum_{i=1}^{P(z)} \ell(|q_i|) \log|h_{i-1}'(z_{i-1})| = -2\sum_{i=1}^{P(z)} \ell(|q_i|)\, \log|c_{i-1}z - a_{i-1}|\,. \tag{42}$$

Then, the operator

$$\mathbb{G}_{s,D} := \mathbb{G}_s \circ \mathbb{H}_s' \circ \mathbb{G}_s \circ \mathbb{H}_s^{[c]} \circ \mathbb{G}_s$$

is the "generating operator" of the cost $D(z)$, and the equality holds,

$$\mathbb{E}_{\langle f\rangle}[D] := \iint_{\mathcal{D}} D(z)\, F(z,\overline{z}) dxdy, = \iint_{\widetilde{\mathcal{B}}\setminus\mathcal{D}} \mathbb{G}_{2,D}[F](z,\overline{z})\, dxdy = \iint_{\widetilde{\mathcal{B}}\setminus\mathcal{D}} |y|^{r-1}\, \mathbb{G}_{1+r,D}[1](z,\overline{z})\, dxdy, \tag{43}$$

the last equality holding for a density F of valuation r

The present Section may be summarized as follows:

Proposition 4. *Consider the* `Core-Gauss` *algorithm when its inputs are distributed inside the disk $\mathcal{D}$ with the density of valuation $r > 0$ Then, the main probabilistic objects described in* (39), (41), (43) *are expressed with the operator $\mathbb{H}_{1+r}$, its iterates or its quasi-inverse $\mathbb{G}_{1+r} = (I - \mathbb{H}_{1+r})^{-1}$*

4.10. **Functional analysis.** We thus need precise information on the operator $\mathbb{H}_s$ and its the quasi-inverse $\mathbb{G}_s = (I - \mathbb{H}_s)^{-1}$ which is omnipresent in the expressions of our probabilistic studies as the quasi-inverse $\mathbf{G}_s = (I - \mathbf{H}_s)^{-1}$ was already omnipresent in the probabilistic analyses of the `C-Euclid` Algorithm.

It is first needed to find a convenient functional space where the operator $\mathbb{H}_s$, and its variants $\mathbb{H}_{s,w,(c)}$ will possess good spectral properties. In the same vein as in Section 3.5, we consider the open disk $\mathcal{V}$ of diameter $[-1/2, 1]$ and the functional space $B_\infty(\mathcal{V})$ of all functions F (of two variables) that are holomorphic in the domain $\mathcal{V}\times\mathcal{V}$ and continuous on the closure $\overline{\mathcal{V}}\times\overline{\mathcal{V}}$. Endowed with the sup-norm, this is a Banach spaces; for $\Re(s) > (1/2)$, the transfer operator $\mathbb{H}_s$ acts on $B_\infty(\mathcal{V})$, and is compact. Furthermore, when weighted by a cost of moderate growth [i.e., $c(h_{\langle q\rangle}) = O(\log q)$], for w close enough to 0, and $\Re s > (1/2)$, the operator $\mathbb{H}_{s,w,(c)}$ also acts on $B_\infty(\mathcal{V})$, and is also compact.

The operator $\mathbb{H}_{s,w,(c)}$ also possesses nice spectral properties (see for instance [75]): for a complex number s close enough to the real axis, with $\Re s > (1/2)$ and $|w|$ small enough, it has a unique dominant eigenvalue, denoted by $\lambda_{(c)}(s,w)$, which is separated from the remainder of the spectrum by a spectral gap. This implies the following: for any fixed s close enough to the real axis, the quasi–inverse $w \mapsto (I - \mathbb{H}_{s,w,(c)})^{-1}$ has a dominant pôle located at $w = w_{(c)}(s)$ defined by the implicit equation $\lambda_{(c)}(s, w_{(c)}(s)) = 1$.

When $w = 0$, one recovers the plain operator $\mathbb{H}_s$, and the diagonal relation (35) shows that the operator $\mathbb{H}_s$ has the same dominant eigenvalue $\lambda(s)$ as the operator $\mathbf{H}_s$. The dominant eigenfunction Ψ_s extends the dominant eigenfunction ψ_s of $\mathbf{H}_s$ and coincides with ψ_s on the diagonal. For $s = 1$, it has a dominant eigenvalue $\lambda(1) = 1$ with a dominant eigenfunction Ψ, which is an extension of the invariant density ψ of the `C-Euclid` Algorithm, and admits an exact expression,

$$\Psi(z,u) = \frac{1}{\log\phi}\frac{1}{u-z}\left(\log\frac{\phi+u}{\phi+z} + \log\frac{\phi^2-u}{\phi^2-z}\right) \quad \text{for } z \neq u, \qquad \Psi(z,z) = \psi(z). \tag{44}$$

Near $s = 1$, the quasi-inverse satisfies

$$(I - \mathbb{H}_s)^{-1}[F](z,u) \sim_{s\to 1} \frac{1}{s-1}\frac{1}{h(\mathcal{E})}\, I[F]\, \Psi(z,u) \qquad \text{with} \quad I[F] := \int_{\widetilde{\mathcal{I}}} F(x,x)dx. \tag{45}$$

4.11. **Average-case analysis in the continuous model.** We now consider the `Core-Gauss` algorithm with an initial density of valuation r. We begin with Proposition 4 which provides expressions in terms of the transfer operator $\mathbb{H}_s$, its iterates or its quasi-inverse. And we apply the results of the previous Section. This gives rise to two types of results for the average-case analysis, the first type being devoted to the particular case of the number of iterations.

Number of iterations and conditional densities. We begin with the description of the event $[R \geq k+1]$ given in (29). This leads to the expression of the probability of the event $[R \geq k+1]$, namely

$$\mathbb{P}_{(r)}[R \geq k+1] = \frac{1}{A(r)} \sum_{h \in \mathcal{H}^k} \iint_{h(\mathcal{D})} |y|^{r-1} dxdy = \frac{1}{A(r)} \iint_{\mathcal{D}} |y|^{r-1} \, \mathbb{H}_{1+r}^k \, [1](z,\overline{z}) \, dxdy,$$

where $A(r)$ is defined in (32). Using the characterisation due to Hurwitz given in (4) leads to the following result:

Theorem 3. *Consider the* `Core-Gauss` *algorithm, when its inputs are distributed inside the disk $\mathcal{D}$ with the density of valuation $r > 0$. Then the following holds:*

(*a*) *The expectation of the number R of iterations admits the following expression*

$$\mathbb{E}_{(r)}[R] = \frac{2^{2r+2}}{\zeta(2r+2)} \sum_{\substack{c,d \geq 1 \\ d\phi < c < d\phi^2}} \frac{1}{(cd)^{1+r}}.$$

(*b*) *The number R of iterations asymptotically follows a geometric law of ratio $\lambda(1+r)$ where $\lambda(s)$ is the dominant eigenvalue of the transfer operator $\mathbb{H}_s$, and*

$$\mathbb{P}_{(r)}[R \geq k+1] \sim_{k \to \infty} b(r) \, \lambda(1+r)^k$$

where $b(r)$ is a strictly positive constant which depends on the valuation r.

(*c*) *The conditional density $F_r^{[k]}(z\overline{z})$ on $\mathcal{D}$ tends to $|y|^{r-1}\Psi_{1+r}(z,\overline{z})$ where Ψ_{1+r} is the dominant eigenfunction of the operator $\mathbb{H}_{1+r}$.*

It seems that there does not exist any close expression for the dominant eigenvalue $\lambda(s)$. However, this dominant eigenvalue is polynomial–time computable, as it is proven by Lhote [45]. In [25], numerical values are computed in the case of the uniform density, i.e., for $\lambda(2)$ and $\mathbb{E}_{(1)}[R]$,

$$\mathbb{E}_{(1)}[R] \approx 1.08922, \qquad \lambda(2) \approx 0.0773853773.$$

For $r \to 0$, the dominant eigenvalue $\lambda(1+r)$ tends to $\lambda(1) = 1$ and $\lambda(1+r) - 1 \sim r\lambda'(1)$. This explains the evolution of the behavior of the Gauss Algorithm when the data become more and more concentrated near the real axis.

Distribution of additive costs in the continuous model. We now study any additive cost and wish to prove that the sequence $k \mapsto \mathbb{P}_{(r)}[C_{(c)} = k]$ has a geometrical decreasing, with a precise estimate for the ratio. For this purpose, we use the moment generating function $\mathbb{E}_{(r)}(\exp[wC_{(c)}])$ of the cost $C_{(c)}$, for which we have provided an alternative expression in Eq. (39). The probability $\mathbb{P}_{(r)}[C_{(c)} = k]$ (for $k \to \infty$) is obtained by extracting the coefficient of $\exp[kw]$ in the moment generating function. Then the asymptotic behavior of $\mathbb{P}_{(r)}[C_{(c)} = k]$ is related to singularities of $\mathbb{E}_{(r)}(\exp[wC_{(c)}])$. This series has a pôle at $e^{w_{(c)}(r+2)}$ where $w = w_{(c)}(s)$ is defined by the spectral equation $\lambda_{(c)}(s,w) = 1$ that involves the dominant eigenvalue $\lambda_{(c)}(s,w)$ of the operator $\mathbb{H}_{s,w,(c)}$ which is described in (38). Then, with classical methods of analytic combinatorics, we obtain:

Theorem 4. *Consider the* `Core-Gauss` *algorithm, when its inputs are distributed inside the disk $\mathcal{D}$ with the density of valuation r. Then, any additive cost $C_{(c)}$ defined in (21), associated to a step-cost c of moderate growth asymptotically follows a geometric law.*

The ratio of this law, equal to $\exp[-w_{(c)}(r+1)]$ *is related to the solution* $w_{(c)}(s)$ *of the spectral relation* $\lambda_{(c)}(s,w)=1$ *which involves the dominant eigenvalue of the transfer operator* $\mathbb{H}_{s,w,(c)}$*, and*

$$\mathbb{P}_{(r)}[C_{(c)}=k] \sim_{k\to\infty} a(r)\exp[-kw_{(c)}(r+1)], \qquad \text{for } k\to\infty. \tag{46}$$

where $a(r)$ *is a strictly positive constant which depends on cost* c *and valuation* r*. When* $r\to 0$*, the solution* $w_{(c)}(r+1)$ *satisfies* $w_{(c)}(r+1)=\Theta(r)$ *for any cost* c *of moderate growth.*

We now state three main results: the first one describes the evolution in the continuous model, when the valuation r tends to 0; the second one describes the evolution of the discrete model, when the integer length N tends to ∞, the valuation being fixed; finally, the third one describes the evolution of the discrete model when the valuation r tends to 0 *and* the integer length N tends to ∞.

4.12. **Average-case analysis in the continuous model when** $r\to 0$**.**

Theorem 5. *Consider the* `Core-Gauss` *Algorithm, where its inputs are distributed inside the input disk* $\mathcal{D}$ *with the density of valuation* $r>0$*. Then, the mean value* $\mathbb{E}_{(r)}[C]$ *of any additive cost* C *of moderate growth, and the mean value* $E_{(r)}[D]$ *of cost* D *satisfy when* $r\to 0$*,*

$$\mathbb{E}_{(r)}[C]\sim \frac{1}{r}\frac{\mathbb{E}[c]}{h(\mathcal{E})}, \qquad \mathbb{E}_{(r)}[D]\sim -\frac{1}{r^2}\,\frac{\mathbb{E}[\ell]}{h(\mathcal{E})}.$$

When r *tends to 0, the output density, associated to the initial density of valuation* r*, tends to* $\dfrac{1}{h(\mathcal{E})}\dfrac{1}{y}\Psi$*, where* Ψ *is the invariant density for* $\mathbb{H}_1$ *described in Eq. (44).*

Remark that the constants which appear here are closely related to those which appear in the analysis of the Euclid algorithm described in Section 3. More precisely, the asymptotics are almost the same when we replace $1/r$ in the previous theorem by $\log N$. Next theorems will make precise this observation.

4.13. **Analyses in the discrete model.** It is now possible to transfer this analysis to the discrete model $\mathcal{M}_{(r,N)}$, that we have already described in Section 4.6. This transfer *continuous* $\to$ *discrete* is not of the same type as in the previous Section. For the Euclid Algorithm, we need generating functions, as the behaviour of the algorithm is very different on discrete data (on which it stops) and on generic data (where it does not stop almost surely). Here, the behaviour of the algorithm is essentially the same on discrete data and on continuous data. This is why it is natural to consider discrete data *embedded* in continuous data. We then proceed as described in Section 4.6 and deal with the set $\mathcal{D}_N$. We are led to mainly use the Gauss Principle that relates the number of points of $\mathcal{D}_N$ in a domain of $\mathcal{C}\subset\mathcal{D}$ with both the area of $\mathcal{C}$ and the length of its frontier $\partial\mathcal{C}$. For instance, a disk of radius ρ included in $\mathcal{D}$ contains $\pi N^2\rho^2+O(\rho N+1)$ points of $\mathcal{D}_N$.

We first consider the model $\mathcal{M}_{(r,N)}$ for a fixed valuation $r>0$, and let $N\to\infty$. Then, we let $N\to\infty$ together with $r\to 0$.

For fixed r **and** $N\to\infty$**.** We describe the behaviour of the `Core-Gauss` algorithm in the discrete model $\mathcal{M}_{(n,r)}$ when the length N of the data becomes large.

Theorem 6. *Consider the* `Core-Gauss` *Algorithm, where its integer inputs* (u,v) *are distributed in* $\mathcal{D}$ *according to the model* (r,N)*, and denote by* X *any additive cost* C *of moderate growth, or cost* D*. Then, when* $N\to\infty$*, the mean value* $\mathbb{E}_{(r,N)}[X]$ *of cost* X *tends to the mean value* $\mathbb{E}_{(r)}[X]$*. More precisely, with* $M:=\log N$*, one has*

$$\mathbb{E}_{(r,N)}[X]=\mathbb{E}_{(r)}[X]+M^{e(X)}e^{-Mr}\,O\left(\max\{1,Mr\}\right),$$

where the exponent $e(X)$ *depends on cost* X *and satisfies* $e(C)=1, e(D)=2$*.*
The mean value $\mathbb{E}_{(r,N)}[B]$ *of the bit-complexity* B *satisfies,*

$$\mathbb{E}_{(r,M)}[B]\sim \mathbb{E}_{(r)}[Q]\cdot M.$$

In particular, the mean bit-complexity is linear with respect to the size M.

For $r \to 0$ and $N \to \infty$. Finally, the last result describes the transition between the `Core-Gauss` algorithm and the `C-Euclid` Algorithm, obtained when the valuation r tends to 0, and the integer length N tends to ∞:

Theorem 7. *Consider the* `Core-Gauss` *Algorithm, where its integer inputs (u, v) where its integer inputs (u, v) are distributed in $\mathcal{D}$ according to the model (r, N), and denote by X any additive cost C of moderate growth, or cost D. Let $M := \log N$. When the integer length N tends to ∞ and the valuation r tends to 0, with $Mr = \Omega(1)$, the mean value $\mathbb{E}_{(r,N)}[X]$ of cost X satisfies*

$$\mathbb{E}_{(r,N)}[X] = \mathbb{E}_{(r)}[X] \left[1 + O\left(M^{e(X)} e^{-Mr}\right)\right] O\left(\frac{1}{1 - e^{-Mr}}\right),$$

where the exponent $e(X)$ depends on cost X and satisfies $e(C) = 1, e(D) = 2$.
Then, if we let $rM =: M^\alpha \to \infty$ (with $0 < \alpha < 1$), then the mean values satisfy

$$\mathbb{E}_{(r,N)}[C] \sim \frac{\mathbb{E}[c]}{h(\mathcal{E})} M^{1-\alpha}, \qquad \mathbb{E}_{(r,N)}[D] \sim -\frac{\mathbb{E}[\ell]}{h(\mathcal{E})} \frac{1}{\log 2} M^{2-2\alpha} \qquad \mathbb{E}_{(r,M)}[B] \sim \frac{\mathbb{E}[\ell]}{h(\mathcal{E})} M^{2-\alpha}.$$

If now rM is $\Theta(1)$, then

$$\mathbb{E}_{(r,N)}[C] = \Theta(M), \qquad \mathbb{E}_{(r,M)}[D] = \Theta(M^2), \qquad \mathbb{E}_{(r,M)}[B] = \Theta(M^2).$$

Open question. *Provide a precise description of the phase transition for the behavior of the bit-complexity between the* `A-Gauss` *algorithm for a valuation $r \to 0$ and the* `C-Euclid` *algorithm: determine the constant hidden in the Θ term as a function of $M\, r$.*

4.14. **Historical and bibliographic notes.** The Gauss algorithm seems to be first described by Lagrange. Lagarias gave in [38] a first polynomial bound for the number iterations of the discrete algorithm. This was improved later by Vallée who gave in [69] the best possible bounds. The general complex framework was described in [20]. The same paper also introduces the `Core-Gauss` algorithm and provides a detailed analysis of the Gaussian algorithm, both in the average case and in probability, in the case where the inputs are uniformly distributed. Then, these results have been extended to study additive costs, in a general model with valuation (see [78], [80]).
Paper [75] introduces for the first time transfer operators with two variables, together with their functional analysis. The notion of notion of valuation appeared for the first time in [72].

5. The LLL algorithm

This last section is devoted to higher dimensions. We begin to describe the general framework, with some elements in geometry of numbers (Section 5.1) together with complexity issues for the basic problem (the Shortest Vector problem, in Section 5.2). We explain in Section 5.3 how the lattice reduction problem arises in many application areas. Then, we give a precise algorithmic description of the LLL algorithm; we begin in Section 5.4 with general principles which explain the role of the Gauss Algorithm inside the LLL algorithm. Then, we describe the algorithm with some of its variants (Sections 5.5 and 5.6), we explain the role plated by the potential in the worst-case analysis (Section 5.7), and finally describe the properties of the output basis (Section 5.8). The remainder of the Section is devoted to the probabilistic analysis of the algorithm. We begin with analyses that are performed in the uniform model (Section 5.9). Unfortunately, this model is not well-adapted to most of applications, and we design in Section 5.10 a general modelling of the inputs, based on the notion of valuation, that seems to be both manageable and realistic enough. As the underlying dynamical system is too complex to deal with, we design three simplified models which describe particular executions : model `M1` in Section 5.11, model `M2` in Section 5.12 and finally model `M3` in Section 5.13. The model `M1` is above all pedagogical, and is used as a reference model. The model `M2` is more realistic, and we prove an asymptotic geometric

law for the number of iterations when the inputs are distributed along the model described in Section 5.10. The model `M3` is just mentioned for a possible future work.

5.1. **Geometry of numbers.** We now study the case of general dimensions $d \geq 3$, What can be expected about a good basis? As a good basis has to contain short vectors, we begin to focus on short vectors in a lattice, and we recall the following definitions, about *successive minima*:

(*i*) *The first minimum $\lambda(\mathcal{L})$ is the norm of a shortest non-zero vector of the lattice.*
(*ii*) *More generally, the k-th minimum $\lambda_k(\mathcal{L})$ is the radius of the smallest Euclidean ball that contains at least k independent vectors of the lattice.*
(*iii*) *A minimal system of the lattice $\mathcal{L}$ is a system $\boldsymbol{b}$ of independent vectors whose k-th vector satisfies $\|b_k\| = \lambda_k(\mathcal{L})$.*

The first difficulty immediately arises as it is not always possible to choose as a basis of the lattice $\mathcal{L}$ a minimal system: generally speaking, a lattice $\mathcal{L}$ does not always have a *minimal* basis. Then, we restrict our wishes: we only consider the first minimum, and compare it to another parameter of the lattice, namely the determinant.

For a basis $\boldsymbol{b}$ of the lattice $\mathcal{L}$, the determinant $G(\boldsymbol{b})$ of the Gram matrix $\boldsymbol{b}$, whose coefficients are the scalar products $(b_i \cdot b_j)$, equals the square of the d-dimensional volume of the parallelotop built on the system b. As any other basis $\boldsymbol{b}'$ is written as $U\boldsymbol{b}$ with a unimodular matrix U, the determinant $G(\boldsymbol{b})$ does not depend on the basis $\boldsymbol{b}$ of the lattice $\mathcal{L}$: it is then called the determinant of the lattice and denoted as $\det \mathcal{L}$. This is an integer number.

The main result of geometry of numbers, the Minkowski Theorem, relates these two parameters $\lambda(\mathcal{L})$ and $\det \mathcal{L}$: *For any d, there is a constant γ_d, such that, for any $\mathcal{L}$ of dimension d,*

$$\lambda(\mathcal{L})^2 \leq \gamma_d \left[\det \mathcal{L}\right]^{1/d},$$

and the Hermite constant γ_d has a polynomial growth with respect to d.

5.2. **The shortest vector problem.** However, the two parameters (determinant and shortest vector) do not play the same algorithmic role. When a lattice $\mathcal{L}$ of dimension d is given by an integer basis $\boldsymbol{b}$ of length $M := \max \|b_i\|^2$, the input size is $O(d \log M)$, and it is *easy* to compute the determinant $\det \mathcal{L}$, namely in polynomial-time with respect to the size $O(d \log M)$. However, it is (probably) *difficult* to compute[10] a shortest non zero vector, as we now see. We consider the following problem:

Shortest Vector Problem [SVP]
Given a basis $\boldsymbol{b}$ of a lattice $\mathcal{L}$, find a non-zero vector v of $\mathcal{L}$ that satisfies $\|v\| = \lambda(\mathcal{L})$.

This problem is only known to be NP–hard for randomized reductions. But it is closely surrounded by problems that are proven to be NP–hard, and it is thus conjectured to be NP-hard. It is then probably not possible to compute a shortest non-zero vector in polynomial-time, and this leads to consider approximate versions of the SVP Problem:

Problem γ–SVP.
Given a basis $\boldsymbol{b}$ of a lattice $\mathcal{L}$, find a short enough vector v that satisfies $\|v\| \leq \gamma\lambda(\mathcal{L})$.

There exist algorithms that solve this problem in polynomial-time when the approximation factor is $\gamma = 2^{O(d)}$. The LLL algorithm can be viewed as such an approximation algorithm, as we see in the sequel.

[10] As most of the results in the classical geometry of numbers, the proof of Minkowski's theorem is not constructive, and does not provide any way to obtain a *short vector* in the lattice.

5.3. **Lattice reduction and applications.** A lattice is probably the simplest structure of the discrete linear algebra, and it is a natural objet to model linear structures which arise in algorithmics. Here, the structure is viewed via its embedding in the vectorial space $\mathbb{R}^n$ endowed with its Euclidean structiure, and the lattice reduction problem is a central problem in the interplay between *algebra* and *metric Euclidean topology*. Here, we five some examples which are well- described in the book : integer programming (see [1]), cryptography (see [52, 53]), computational number theory (see [31, 35]), etc...

Integer Programming. Given a lattice $\mathcal{L} \subset \mathbb{R}^n$ and a domain $\mathcal{C} \subset \mathbb{R}^n$, we wish to describe in an efficient way the intersection $\mathcal{C} \cap \mathcal{L}$. This is the basic problem in integer programming, and this was also a strong historical motivation to design efficient algorithms for the lattice reduction problem. The two problems are indeed closely related, as we now see. Consider a lattice $\mathcal{L}$ with a *good* basis $\boldsymbol{b}$; then, the lattice $\mathcal{L}$ can be defined as a sequence $\mathcal{L}_n$ of *affine* parallel lattices with a large enough spacing between them. Then there are *few* such $\mathcal{L}_k$ which intersect $\mathcal{C}$, and the intersection $\mathcal{C} \cap \mathcal{L}$ is just the union of few intersections $\mathcal{L}_n \cap \mathcal{C}$; one just proceeds in a recursive manner on each intersection $\mathcal{L}_n \cap \mathcal{C}$.

Computational number theory. We describe three applications in this area.

Diophantine approximations. Given a d-uple $(\alpha_1, \alpha_2, \ldots, \alpha_d)$ of real numbers, one wishes to find d integers $(p_1, p_2, \ldots, p_d)$ and an integer q such that the rational d-uple $(p_1/q, p_2/q, \ldots, p_d/q)$ be a good approximation of the d-uple $(\alpha_1, \alpha_2, \ldots, \alpha_d)$. A non-constructive answer is known due to Dirichlet and based on Minkowski's theorem. But, it is possible to give an approximate but constructive version of this theorem when applying lattice reduction to a particular lattice whose matrix is a *bordered unit matrix*

Factoring polynomials in $\mathbb{Z}[X]$. The original LLL algorithm [43] was designed in a paper which was mainly devoted to factorization of polynomials of $\mathbb{Z}[X]$. The main basic idea is the following: given a very good approximation $\overline{\alpha}$ of a root α of a polynomial f, it is possible to determine the minimal polynomial of α, and thus an irreducible factor of f. In the context of the paper [43], the approximation $\overline{\alpha}$ is a p-adic number obtained by the factorization $\mod p$ due to Berlekamp, following a lifting by Hensel's lemma (see also [35])

Guessing roots mod p. Consider a polynomial $f \mod p$ for which we know an approximate value $\overline{\alpha}$ of a root $\alpha \mod p$. We wish to recover the exact value α. As soon as the approximation is good enough, applying the lattice reduction to a convenient lattice allows to recover the exact root. (see [70, 16])

Cryptology. There are two areas in cryptology: a positive side *(crytography)*, where one builds cryptosystems, and a negative side *(cryptanalysis)*, where one breaks cryptosystems. In cryptology, one needs difficult problems which may become easy in well–specified frameworks. Via the status of the SVP Problem (a difficult problem, which remains hard on average, and admits good approximation algorithms), lattices play a central role in both sides.

Cryptanalysis via lattice reduction. The LLL algorithm easily broke all the cryptosystems built on linear problems (for instance the cryptosystems based on the knapsack problem). But, it was also succcessful to break cryptosystems based on polynomial problems (after a suitable *linearization*), and for instance particular instances of the celebrated RSA cryptosystem.

Lattice-based cryptography. This is a more recent area where lattices allow to create new classes of cryptosystems, with new interesting features. First, lattices provide an interesting alternative *algorithmic* area, where the problems probably remain hard even against quantum computers, unlike more widely used and known public key cryptography such as the RSA cryptosystem. Second, they made possible to obtain stronger proofs of security [30], based on the average-case hardness of problems. Finally, they allow a desirable feature in modern communication system architectures, namely homomorphic encryption : computations to be carried out on ciphertext, thus generating an encrypted result which, when decrypted, matches the result of operations performed on the plaintext.

5.4. **Principles of the LLL algorithm.** The algorithm first computes the Gram-Schmidt orthogonal basis $\boldsymbol{b}^\star = (b_1^\star, \dots, b_d^\star)$ of $\boldsymbol{b}$ and the Gram-Schmidt matrix Π for which $\boldsymbol{b}$ is written as $\Pi \boldsymbol{b}^\star$. The $d \times d$ square matrix $\Pi = (p_{i,j})_{1 \le i,j \le d}$ is defined as (see also Figure 9)

$$p_{i,i} = 1, \quad p_{i,j} = 0 \ \text{ for } j > i, \qquad p_{i,j} = \frac{(b_i \cdot b_j^\star)}{\|b_j^\star\|^2} \ \text{ for } i > j\,.$$

$$\Pi := \begin{array}{c} \\ b_1 \\ b_2 \\ \vdots \\ b_{i-1} \\ b_i \\ b_{i+1} \\ \vdots \\ b_d \end{array} \begin{array}{c} \begin{array}{cccccccc} b_1^\star & b_2^\star & \dots & b_{i-1}^\star & b_i^\star & b_{i+1}^\star & \dots & b_d^\star \end{array} \\ \begin{pmatrix} 1 & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\ p_{2,1} & 1 & \dots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{i-1,1} & p_{i-1,2} & \dots & 1 & 0 & 0 & 0 & 0 \\ p_{i,1} & p_{i,2} & \dots & p_{i,i-1} & 1 & 0 & 0 & 0 \\ p_{i+1,1} & p_{i+1,2} & \dots & p_{i+1,i-1} & p_{i+1,i} & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{d,1} & p_{d,2} & \dots & p_{d,i-1} & p_{d,i} & p_{d,i+1} & \dots & 1 \end{pmatrix} \end{array}$$

Main steps of the LLL algorithm = Gauss' reduction steps on systems U_i

$$U_i := \begin{array}{c} \\ u_i \\ v_i \end{array} \begin{array}{c} \begin{array}{cc} b_i^\star & b_{i+1}^\star \end{array} \\ \begin{pmatrix} 1 & 0 \\ p_{i+1,i} & 1 \end{pmatrix} \end{array}$$

FIGURE 9. Matrix P and systems U_i.

The LLL algorithm deals with the systems U_i "built on the diagonal" of matrix Π described in Figure 9 and performs the `A-Gauss` algorithm on these two-dimensional systems U_i, with three differences,

(*a*) The output test is *weaker*: with a fixed parameter $\tau \le 1$, the test $|v_i| > |u_i|$ is replaced by the test $|v_i| > \tau |u_i|$. Then, the output domain for the complex number z_i associated with the system U_i is the pseudo fondamental domain

$$\mathcal{F}_\tau := \{z \mid 0 \le \Re z \le 1/2, \ \ |z| \ge \tau\}\,,$$

which coincides with $\widetilde{\mathcal{F}}$ for $\tau = 1$.

(*b*) The operations are "decided" on the system U_i, then *reflected* on the system (b_i, b_{i+1}).

(*c*) The algorithm is performed on systems U_i *step by step*. This is due to the fact that the systems U_i are *intersecting* and modifications on the system U_i change the systems U_{i-1} and U_{i+1}.

5.5. **Description of the LLL algorithm.** The norms $\ell_i = \|b_i^\star\|$ of the orthogonal vectors $b_i^\star$ and the subdiagonal coefficients $p_{i+1,i}$ of matrix Π describe the geometry of the system U_i. Using the complex number z_i associated to U_i, we mainly deal with the two vectors $\boldsymbol{x} = (x_i) := (\Re z_i)$ and $\boldsymbol{y} = (y_i) := (\Im z_i)$ which satisfy

$$y_i := \ell_{i+1}/\ell_i, \qquad x_i := \{\{p_{i+1,i}\}\}, \qquad \text{for } 1 \le i \le d-1,$$

(where $\{\{t\}\}$ is the centered fractional part of t) and play a fundamental role in the LLL algorithm. With a parameter $\tau \le 1$, the LLL(τ) algorithm builds a basis $\widehat{\boldsymbol{b}}$ that satisfies the following conditions and is then called *Lovász–reduced*:

(*a*) For $i \in [1..d-1]$, it fulfills all the Lovász conditions $\mathcal{L}_\tau(i) : \widehat{x}_i^2 + \widehat{y}_i^2 \ge \tau^2$.

(*b*) It is size-reduced: the coefficients $\widehat{p}_{i,j}$ satisfy $|\widehat{p}_{i,j}| \le 1/2$ for $1 \le j < i \le d$.

The LLL algorithm performs two main operations, of the same type as the `A-Gauss` Algorithm.

(*i*) *Translations.* For $j < i$, the translations $b_i := b_i - \lfloor p_{i,j} \rceil b_j$ (where $\lfloor x \rceil$ denotes the nearest integer to x), modify the Gram-Schmidt matrix P whose coefficients now satisfy $|p_{i,j}| \le 1/2$ for $j < i$. These translations ensure that the basis $\boldsymbol{b}$ is size-reduced and do not modify $\boldsymbol{b}^\star$. They can be performed at any time of the execution.

(*ii*) *Exchanges between vectors.* This is the main operation, which is performed when the Lovász condition $\mathcal{L}_\tau(i)$ is not satisfied for some i. More precisely, the algorithm chooses an index i in the set

$$\mathcal{J}_\tau(\boldsymbol{b}) := \{i \in [1..d-1];\ \ \mathcal{L}_\tau(i) \text{ is not fulfilled}\} = \{i \in [1..d-1];\ \ x_i^2 + y_i^2 < \tau^2\}\,,$$

and it exchanges the two vectors b_i and b_{i+1}. As the Gram-Schmidt process depends on the order on the family $\boldsymbol{b}$, the exchange modifies the components $b_i^\star$ and $b_{i+1}^\star$ of the system $\boldsymbol{b}^\star$ (See Figure 10). The new $b_i^\star$ denoted by $\check{b}_i^\star$ equals the vector $b_{i+1}^\star + x_i\, b_i^\star$, and the decreasing factor ρ between the new $\|\check{b}_i^\star\|$ and the old $\|b_i^\star\|$ satisfies

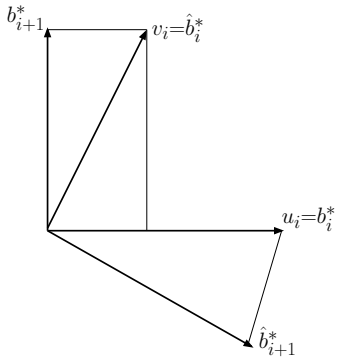
$$\rho^2 := x_i^2 + y_i^2 < \tau^2 \leq 1\,. \tag{47}$$

Due to the invariance of the determinant, the product $\ell_i \cdot \ell_{i+1}$ does not change, and the new values $\check{y}_{i-1}, \check{y}_i, \check{y}_{i+1}$ satisfy

$$\check{y}_{i-1} = \rho\, y_{i-1}, \qquad \check{y}_i = \rho^{-2}\, y_i, \qquad \check{y}_{i+1} = \rho\, y_{i+1}\,. \tag{48}$$

Remark that, for extreme indices $i = 1$ or $i = d-1$, the part of the computation that involves y_0 or y_d is not performed. During an exchange performed at $i \notin \{1, d-1\}$, the total product $\prod_{i=1}^{d-1} y_j$ remains constant, whereas y_i is increasing, and both y_{i-1} and y_{i+1} are decreasing. When the exchange is performed at $i = 1$ or $i = d-1$, the total product increases with the factor ρ^{-1}. We will see that during any exchange the potential $P(\boldsymbol{y})$ defined in (50) is itself increasing with the factor ρ^{-1}.

After any exchange, the matrix Π must be updated and size-reduced. There are in particular three new values $\check{x}_{i-1}, \check{x}_i, \check{x}_{i+1}$, defined as centered fractional parts, and one has for instance $\check{x}_i = \{\{\rho^{-2} x_i\}\}$.



LLL(τ) Algorithm $\quad(\tau \leq 1)$

Input. A basis $\boldsymbol{b} = (b_1, \ldots, b_d)$ of a lattice $\mathcal{L}$.
Output. A LLL(τ)-reduced basis $\widehat{\boldsymbol{b}}$ of $\mathcal{L}$

Compute the vector $\boldsymbol{b}^\star$ and the matrix P;
Size reduce $\boldsymbol{b}$;
While the set $\mathcal{J}_\tau(\boldsymbol{b})$ is not empty, **do**
  Choose an index $i \in \mathcal{J}_\tau(\boldsymbol{b})$;
  Exchange b_i and b_{i+1};
  Update $\boldsymbol{b}^\star$ and P;
  Size-reduce $\boldsymbol{b}$

FIGURE 10. On the left, description of the exchange $b_i \leftrightarrow b_{i+1}$ viewed on the basis U_i. On the right, description of the LLL algorithm.

There are various possible strategies for the choice of the index $i \in \mathcal{J}_\tau(\boldsymbol{b})$.

- (C) The classical strategy chooses $i := \min \mathcal{J}_\tau(\boldsymbol{b})$.
- (R) The random strategy chooses i uniformly at random in $\mathcal{J}_\tau(\boldsymbol{b})$.
- (G) The greedy strategy chooses the index $i := \text{argmin}\{x_j + y_j\}$.

A general description of the LLL algorithm is given in Figure 10.

$\Sigma(\sigma)$ Algorithm $\quad(\sigma \leq \sqrt{3}/2)$

Input. A basis $\boldsymbol{b} = (b_1, \ldots, b_d)$ of a lattice $\mathcal{L}$.
Output. A Siegel(σ)-reduced basis $\widehat{\boldsymbol{b}}$ of $\mathcal{L}$

Compute the vector $\boldsymbol{b}^\star$ and the matrix P;
Size reduce $\boldsymbol{b}$;
While the set $\mathcal{K}_\sigma(\boldsymbol{b})$ is not empty, **do**
  Choose an index $i \in \mathcal{K}_\sigma(\boldsymbol{b})$;
  Exchange b_i and b_{i+1};
  Update $\boldsymbol{b}^\star$ and P;
  Size-reduce $\boldsymbol{b}$

With an additive point of view
$t_i = -\log_\sigma y_i, \quad \alpha = -\log_\sigma \rho,$
and only viewed on the vector t,
the $\Sigma(\sigma)$ algorithm is written as:
While $\exists t_i > 1$, **do**
$\check{t}_i := t_i - 2\alpha;$
$\check{t}_{i-1} := t_{i-1} + \alpha;$
$\check{t}_{i+1} := t_{i+1} + \alpha;$

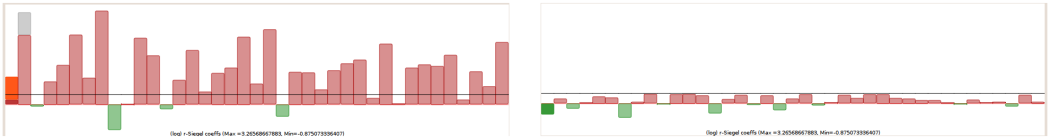


FIGURE 11. On the left, description of the Siegel version of the LLL algorithm. On the right, description of the additive point of view. On the bottom, example of the additive point of view on an input basis (on the left) and the associated output basis via the Siegel version of the LLL algorithm.

5.6. **Other versions of the LLL algorithm.** There exist *weaker* versions of the LLL algorithm which are perhaps easier to understand: we introduce the Siegel conditions

$$\mathcal{S}_\sigma(i): \quad \widehat{y}_i \geq \sigma.$$

With $\sigma^2 := \tau^2 - (1/4)$, together with inequalities $x_i^2 \leq 1/4$, the conditions $\mathcal{L}_\tau(i)$ entail the Siegel conditions $\mathcal{S}_\sigma(i)$. Then, with a parameter $\sigma \leq \sqrt{3}/2$, we introduce the Siegel algorithm $\Sigma(\sigma)$.
The algorithm $\Sigma(\sigma)$ builds a basis $\widehat{\boldsymbol{b}}$ that satisfies the following conditions and is called *Siegel–reduced*:

(*a*) For $i \in [1..d-1]$, it fulfills all the Siegel conditions $\mathcal{S}_\sigma(i) : \widehat{y}_i \geq \sigma$.
(*b*) It is size-reduced: the coefficients $\widehat{p}_{i,j}$ satisfy $|\widehat{p}_{i,j}| \leq 1/2$ for $1 \leq j < i \leq d$.

The Siegel algorithm has exactly the same structure as the LLL algorithm. It performs translations which size-reduce the matrix Π, deals with the set

$$\mathcal{K}_\sigma(\boldsymbol{b}) := \{i \in [1..d-1]; \quad \mathcal{S}_\sigma(i) \text{ is not fulfilled}\} = \{i \in [1..d-1]; \quad y_i < \sigma\},$$

and, with an index $i \in \mathcal{K}_\sigma(\boldsymbol{b})$, performs the exchange between b_i and b_{i+1}. This exchange is "transferred" on the vector $\boldsymbol{y}$ by the same transform as in (48) which involves the same decreasing factor ρ defined via the equality $\rho^2 = x_i^2 + y_i^2 \leq \sigma^2 + (1/4) = \tau^2$.
The output complex domain for each complex number associated with the local basis U_i is now the domain $\mathcal{R}_\sigma$ that is the union of two rectangles, namely

$$\mathcal{R}_\sigma := \{z \mid 0 \leq \Re z \leq 1/2, \quad |\Im z| \geq \sigma\},$$

and replaces the pseudo-fondamental domain $\mathcal{F}_\tau$ in associated with the LLL(τ) algorithm.

We then adopt an *additive point of view* that is perhaps more pedagogical; it is described in Figure 11. We may view the components t_i as sandpiles. The main action of the Σ algorithm is to increase the components y_i and thus decrease the components t_i; the algorithm ends when the vector t has all its components t_i at most equal to 1. At each step of the algorithm, the positive number $\alpha := \log_\sigma \rho$ is computed, and an amount of sand equal to 2α is removed from the sandpile t_i, divided into two equal

parts, each of them being placed on the left and on the right of t_i. The total amount of sand remains thus constant, except for indices $i = 1$ and $i = d - 1$, where a sand amount equal to α is placed on t_0 or t_d and thus "disappears". Then the general strategy can be informally described as follows :

> *Push the sand towards the borders, where it disappears. Stop when all the sandpiles are small enough.*

It is important to remark that the sand is "more difficult to push" when it is far from the borders.

5.7. **Potential of a basis.** It is not completely clear that the algorithm terminates, since, when it "improves" the system U_i, it also "deteriorates" the systems U_{i-1} and U_{i+1}. In fact, the algorithm terminates due to the evolution of the two (related) types of potentials. The first potential $Q(\boldsymbol{b})$ involves the beginning lattices $\mathcal{L}_i$, where $\mathcal{L}_i$ is generated by the system $\boldsymbol{b}_i := (b_1, b_2, \ldots, b_i)$,

$$Q(\boldsymbol{b}) := \prod_{i=1}^{d-1} \ell_i^{2(d-i)} = \prod_{i=1}^{d-1} \det \mathcal{L}_i, \tag{49}$$

whereas the second potential is a monomial which involves each component y_i of $\boldsymbol{y}$, with an exponent equal to $i(d-i)$, namely,

$$P(\boldsymbol{y}) := \prod_{i=1}^{d-1} y_i^{i(d-i)} = Q(\boldsymbol{b})^{-1} \det(\boldsymbol{b})^{d-1}. \tag{50}$$

These potentials are not modified during the size-reduction steps. During each step of the While loop that deals with some index j, the potential $P(\boldsymbol{y})$ is increasing with the factor $1/\rho^{(j)} > 1/\tau$, where $\rho^{(j)}$ is the decreasing factor defined in (47), whereas the potential $Q(\boldsymbol{b})$ is decreasing with the factor $(\rho^{(j)})^2 < \tau^2$. Then, as soon as $\tau < 1$, there is an upper bound for the number K_τ of iterations of the LLL(τ) algorithm on the basis $\boldsymbol{b}$ which involves the geometric mean ρ of the factors $\rho^{(j)}$, at most equal to τ, via the relations

$$K_\tau(\boldsymbol{b}) = \frac{1}{|\log \rho|} \log \frac{P(\widehat{\boldsymbol{y}})}{P(\boldsymbol{y})} = \frac{2}{|\log \rho|} \log \frac{Q(\boldsymbol{b})}{Q(\widehat{\boldsymbol{b}})} \qquad \text{with} \quad P(\widehat{\boldsymbol{y}}) \geq \sigma^D, \qquad D := d(d^2-1)/12.$$

The potential $Q(\boldsymbol{b})$ of an integer basis is an integer. Then, for an integer lattice, the output potential $Q(\widehat{\boldsymbol{b}})$ is at least equal to 1, while the input potential $Q(\boldsymbol{b})$ is an integer of size $O(d^2 \log M)$. Then, for $\tau < 1$, the number of iterations is $O(d^2 \log M)$, and thus polynomial with respect to the input size $O(d \log M)$. Moreover, it is also proven (see [43]) that all the rational[11] numbers which arise along the execution also remain of polynomial size. However, it will be much more efficient to deal with *floating* numbers. Even if the idea is completely natural, the *needed precision* is difficult to deal with (see [66]).

Limit case $\tau = 1$**.** What happens when the parameter τ equals 1? The potential $Q(\boldsymbol{b})$ is now only stricly decreasing, and the number of iterations is only proven to be finite. This an important question. Experiments reported by Lagarias and Odlyzko [39] hint that the number of iterations of the algorithm LLL(1) seem to remain polynomial with respect to the input size. However, such polynomial bounds are not proven, and there exist only very crude bounds, exponential with respect to dimension d, exhibited in [3] or [44] for instance.

Particular case $\boldsymbol{x} = 0$**.** In the general case when the component x_i is not zero, the exchange between b_i and b_{i+1} does not induce an exchange between the orthogonalized vector $b_i^\star$ and $b_{i+1}^\star$ (see Figure 10). This is why the LLL algorithm can be viewed as a kind of *skew sorting* process, where the exchange ... modifies the data. However, in the case $x_i = 0$, the exchange between b_i and b_{i+1} induces an exact exchange between the orthogonalized vectors $b_i^\star$ and $b_{i+1}^\star$. So, when all the used coefficients x_i are always zero, the LLL algorithm can be viewed as a *sorting algorithm*, which sorts the lengths ℓ_i. This is a exact sorting algorithm when $\tau = 1$ and a kind of "approximate sorting" algorithm when $\tau < 1$.

[11] Even for an integer input, the Gram-Schmidt process produces rational numbers.

When furthermore the classical strategy (C) is used, this leads for $\tau = 1$ to a classical sorting algorithm, the `InsertionSort` algorithm.

5.8. **Properties of the output basis.** As already mentioned, the Lovász properties for parameter τ yield the Siegel properties for parameter σ with $\sigma^2 := \tau^2 - (1/4) \leq 3/4$, namely lower bounds on ratios y_i of the form $y_i > \sigma$, These lower bounds together with the size-reduction entail two main upper bounds that hold on the output[12] basis $\widehat{\boldsymbol{b}}$, and involve the ratio $(1/\sigma) \geq 2/\sqrt{3}$,

$$\begin{array}{lll} \text{Length defect.} & \|\widehat{b}_1\| & \leq \left(\dfrac{1}{\sigma}\right)^{d-1} \lambda(\mathcal{L}); \\ \text{Orthogonality defect.} & \displaystyle\prod_{i=1}^{n-1} \frac{\|\widehat{b}_i\|}{\|\widehat{b}_i^\star\|} & \leq \left(\dfrac{1}{\sigma}\right)^{d(d-1)}. \end{array}$$

The first inequality compares the first vector of the output basis to the length $\lambda(\mathcal{L})$ of a shortest non-zero vector of the lattice. Then, the first vector is not *too long*. Then, with the previous Section, and for any $\tau < 1$, the LLL(τ) is a polynomial time algorithm which solves the γ-SVP problem with an approximation rate $\gamma = 2^{O(d)}$.
The second inequality compares the length of the vectors of the two output systems, the system $\widehat{\boldsymbol{b}}$, and its orthogonalized $\widehat{\boldsymbol{b}}^\star$. It proves the output basis to be not *too far* from an orthogonalized basis.
As the factor $(1/\sigma)$ is at least equal to $2/\sqrt{3} > 1$, the ratios which occur in the previous bounds are exponentially increasing with respect to d.

The remainder of this section is devoted to the probabilistic analysis of the LLL algorithm: we aim to study in a precise probabilistic way parameters that describe the execution of the algorithm or the geometry of the output. In small dimensions (1 or 2), such a probabilistic analysis is performed in the previous two Sections (Sections 3 and 4) and strongly uses the dynamical system that underlies the algorithm: In the one-dimensional case, this is the Euclidean dynamical system `C-Euclid` on the interval $\mathcal{I} = [0, 1/2]$ with the centered version of the Gauss map. In two dimensions, this is an extension of the previous dynamical system to the right complex plane, which is the `A-Gauss` dynamical system or its core-part, the `Core-Gauss` system. In higher dimensions, the dynamical system that underlies the LLL algorithm is much more involved, and such a precise dynamical analysis seems not possible to perform.

5.9. **Probabilistic analyses of the LLL algorithm in the uniform model.** There are first two natural questions, asked when the ambient space is $\mathbb{R}^n$ and $\boldsymbol{b}$ a basis of cardinality $d \leq n$:

- (i) What is the probability $\pi(d, n, \sigma)$ that $\boldsymbol{b}$ be already that σ- Siegel reduced?
- (ii) What is the mean number of iterations of the LLL(τ) algorithm? of the $\Sigma(\sigma)$ algorithm? What about the distribution of the number of iterations?

These questions were answered in a spherical model, which is an extension of the uniform model. Here, we focus on the uniform model $\mathcal{U}(n, d)$, where each vector b_i is uniformly drawn in the n-dimensional Euclidean unit ball $\mathcal{B}_n = \{x \in \mathbb{R}^n : \|x\| \leq 1\}$. This model is manageable due to two main properties satisfies by the random variables ℓ_j^2: first, they are independent; second, they follow β laws.

Probability $\pi(n, d, \sigma)$**.** This first result, due to Akhavi, Marckert and Rouault [4, 5], exhibits an interesting threshold phenomenon:

Theorem 8. *Consider the model $\mathcal{U}(n, d)$, denote $g = n - d$ and let $n \to \infty$. Then the following holds for the probability $\pi(n, d, \sigma)$ for any $\sigma < 1$*

- (a) *For $g \to \infty$, the probability $\pi(n, d, \sigma)$ tends to 1.*
- (b) *If g is constant, then $\pi(n, d, \sigma)$ tends to a constant in the interval $]0, 1[$.*

[12] These upper bounds are the same for both algorithms LLL(τ) and $\Sigma(\sigma)$

In particular, in the model $\mathcal{U}(n,\theta n)$ with $\theta<1$, any basis $\boldsymbol{b}$ is almost surely $\Sigma(\sigma)$-reduced for any value of σ. Then, in the uniform model, the Siegel algorithm has "not much work to do" as soon as the dimension of the lattice is far from the ambient dimension.

Number of iterations. The second result, due to Daudé and Vallée [19], provides *upper bounds* for the mean number of iterations.

Theorem 9. *In the model $\mathcal{U}(n,d)$, the mean number of iterations K_τ of the algorithm LLL(τ) on a basis $\boldsymbol{b}$ satisfies*

$$\mathbb{E}_{n,d}[K_\tau]\leq \frac{d(d-1)}{n-d+1}\left(\frac{1}{|\log\tau|}\right)\left[\frac{1}{2}\log n+2\right].$$

For $d=\theta n$, with $\theta<1$, one has

$$\mathbb{E}_{n,d}[K_\tau]\leq \frac{cn}{1-c}\left(\frac{1}{|\log\tau|}\right)\left[\frac{1}{2}\log n+2\right]..$$

When $n-d$ is of order $\Omega(n^a)$ with $a\in[0,1[$, then $\mathbb{E}_{n,d}[K_\tau]=O(n^{2-a}\log n)$,

There also exist estimates for the distribution of K_τ, here described in the case of a full-dimensional-lattice ($d=n$) , which show that this distribution admits an exponential tail of geometric type, namely,

$$\mathbb{P}_{n,n}[K\geq k]\leq\left(2\sqrt{n}\,\tau^{-1/n}\right)\tau^{k/n^2}.$$

We recall that asymptotic geometric laws are proven to hold in two dimensions (see Section 4). For higher dimensions, we return later on to the study of this distribution in the case of one of our simplified models (see Section 5.12).

5.10. **Towards a more realistic modelling of inputs: the valuation model.** The results of the previous Section are both interesting (from a theoretical point of view) ... and useless from a more concrete point of view. Anyone who uses the LLL algorithm in the "real algorithmic life" knows that the previous results do not depict the experimental facts he observes (see also [54]): it is clear that the uniform model is not well-adapted to most of the applications areas of the algorithm (except perhaps integer programming).

We have seen in the previous Section 4 that the behaviour of the `A-Gauss` algorithm strongly depends on the distribution of the inputs. The situation is probably the same for the LLL algorithm, and it is important to design particular input models dedicated to particular applications. Here, as previously, the probabilistic input model is related to the geometry of the input basis $\boldsymbol{b}$, mainly described by the geometry of the systems (U_i) which is itself defined via the complex number z_i; as in Section 4, difficult instances are related to complex numbers z_i with a small imaginary part y_i which lead to high sandpiles $t_i:=-\log y_i$.

Then, as in Section 4.6, we deal with *valuations*, and, here, with a *valuation vector* $\boldsymbol{r}:=(r_1,r_2,\ldots,r_{d-1})$ where the component r_i is the valuation attached to the system U_i (or complex z_i) and satisfies $r_i>0$. For any given $a>0$, we consider the hypercube $H_a:=[0,a]^{d-1}$ and an input density $f_{\boldsymbol{r}}$ on H_a, called the density of valuation $\boldsymbol{r}$, which is proportional to $\boldsymbol{y}^{\boldsymbol{r}-1}:=\prod_{i=1}^{d-1}y_i^{r_i-1}$. This defines the model $\mathcal{M}_{\boldsymbol{r},a,d}$.

This model has two advantages: it is both *manageable* and sufficiently *realistic*. We have already seen the expressivity of the valuation in Section 4. We now explain how it may provide a good modelling in many of applications of the LLL algorithm. We insist in Section 5.3 on the particular shape of the input bases that are used in applications. They give rise to particular sandpiles, where the distribution of the sand may vary a lot with the position of the sandpile. For instance, the cryptographic lattices are given by sandpiles which have very often only one pile; this is the case for the knapsack lattices, or lattices which model the NTRU system (see [29] for a description of cryptographic lattices). On the opposite, the lattices used in Ajtai's proof [2] give rise to sandpiles whose all piles are very high (associated to completely non reduced lattices) which will be modelled by a vector $\boldsymbol{r}$ with very small components.

With this valuation vector $\boldsymbol{r}$, it proves convenient to associate the *scaled valuation vector* $\boldsymbol{s}$ whose components are $s_i = r_i/(i(d-i))$. Letting $j := |d/2 - i|$, we see that the scaling coefficient $i(d-i) = (d^2/4) - j^2$ which already appears in the expression of the potential P given in (50) is maximal when the index i is close to $d/2$. With our description via sandpiles, and the definition of the potential given in (50), we know that the sand contained in the central sandpiles (with an index close to $d/2$) is more difficult to be taken out. Then the *scaled valuation* $r_i/i(d-1)$ is a good measure of the difficulty of the system (U_i), that comes both from the valuation r_i (which measures the height of the sandpile) and the position of the index i (which locates the dsandpile).

The model $\mathcal{M}_{\boldsymbol{r},a,d}$ appears to be manageable: it is possible to describe there the distribution of the potential $P(\boldsymbol{y})$ which plays a central role in the algoritthm. This distribution in the model $\mathcal{M}_{\boldsymbol{r},a,d}$ involves the minimum component of the scaled valuation vector.

Proposition 5. [14] *Consider the model $\mathcal{M}_{(\boldsymbol{r},a,d)}$ and let $D := D := d(d^2-1)/6$. Denote by $\boldsymbol{s}$ the scaled valuation vector $\boldsymbol{s}$ whose components are $s_i := r_i/(i(d-i))$, and by $m(\boldsymbol{s})$ the minimum of its components; consider the set $\mathcal{B}$ of indices i for which $i := \operatorname{argmin}\{s_j\}$ and denote by b its cardinality. Then, the distribution of the d-dimensional potential $P(\boldsymbol{y})$ satisfies, for any $C \in [0,1]$*

$$\mathbb{P}_{\boldsymbol{r}}[P(\boldsymbol{y}) \leq C\, a^D] = \Theta\left(C^{m(\boldsymbol{s})}(\log C)^{b-1}\right) \qquad (C \to 0)\,,$$

where the constants hidden in the Θ only depend on $\boldsymbol{s}$. When the components of vector $\boldsymbol{s}$ are all distinct, the valuation vector is called irreducible, and there is an exact expression that involves the monic polynomial $S(x)$ whose roots are the components s_i, namely

$$\mathbb{P}_{\boldsymbol{r}}[P(\boldsymbol{y}) \leq C\, a^D] = -S(0) \sum_{i=1}^{d-1} \frac{1}{s_i S'(s_i)} C^{s_i}.$$

We now introduce simplified models for the execution, which would describe particular executions of the LLL algorithm. We then analyse the behaviour of the LLL algorithm, when it gives rise to *particular trajectories*. The underlying system which "produces" such particular trajectories is *not* defined as a *restriction* of the LLL algorithm to *particular* inputs. It is just a *simplified model* which is expected to remain close enough to the "true" algorithm but become more manageable and easier to analyze. We now describe two possible simplified models for the execution.

5.11. **The sandpile model `M1`.** The main *observable* is the decreasing factor ρ defined in (47). The first simplified model proposed in [47] is the model `M1`(ρ) where the factor ρ defined in (47) is assumed to remain *constant* all along the execution. If one adopts the additive point of view defined in Figure 11 and consider the logarithms t and α in Figure 11, we are led to a very classical dynamical system, called sandpile or chip firing game. The main features of these models are very well known, at least when the input t and the parameter α are integers, and easy to extend to the continuous model. In this case, it is well-known that both the number of steps and the output configuration are independent of the strategy. Even if this model is too simple to model all the features of the LLL system, it exhibits various qualitative phenomena of the "true" algorithm and has many pedagogical advantages. It also helps to propose some conjectures. The sandpile model was further used in [32] for the analysis of another lattice reduction algorithm, the BKZ algorithm.

5.12. **The model `M2`.** In the second model, called `M2`, the factor ρ defined in (47) is no longer assumed to be constant all along the execution. As it is seen in (47), this factor ρ depends on two parameters, namely $y_i = \Im z_i$ and $x_i = \Re z_i$. These two parameters do not play the same role, as we already observed in our study of the two-dimensional case: The main actor is the vector $\boldsymbol{y}$, and the whole execution of the algorithm aims increasing the components of $\boldsymbol{y}$, whereas the vector $\boldsymbol{x}$ (whose all components always belongs to the interval $[0, 1/2]$) only plays an auxiliary role. Moreover, the transformations of the

vector $\boldsymbol{x}$ involve fractional parts and seem to be difficult to deal with, whereas the transformations on the vector $\boldsymbol{y}$ appear to be more readable and more manageable.
Finally, we make the two following assumptions (see [29, 14]):

(i) The coefficients x_i are constant along the execution and equal to $\mu > 0$;
(ii) The strategy is the greedy strategy, and then $i := \text{argmin}\{y_i\}$, so that the decreasing factor ρ always satisfies $\rho^2 = m(\boldsymbol{y})^2 + \mu^2$ where $m(\boldsymbol{y}) = \min(y_i)$.

We first recall two notations: $m(\boldsymbol{y}) := \min(y_j)$, $D := d(d^2-1)/6$.
Then, with a pair (μ, τ) with $\mu \in [0, 1/2]$ and $\tau \in [\mu, 1]$, we associate two hypercubes in $\mathbb{R}^{d-1}$, namely

$$\mathcal{I}_\mu = [0, 1/(2\mu)]^{d-1}, \quad \mathcal{O}_{\mu,\tau} = \Big[(\tau^2-\mu^2)^{1/2}, +\infty\Big[^{d-1} . \tag{51}$$

The exchange between the two vectors b_i and b_{i+1} is then described by the map $T_{i,\mu} : \mathbb{R}^{d-1} \to \mathbb{R}^{d-1}$ which associates to the vector $\boldsymbol{y}$ the vector $\check{\boldsymbol{y}}$ which satisfies $\check{y}_j = y_j$ for $j \notin \{i-1, i, i+1\}$, and

$$\check{y}_{i-1} = y_{i-1}\ (y_i^2+\mu^2)^{1/2}, \quad \check{y}_i = \frac{y_i}{y_i^2+\mu^2}, \quad \check{y}_{i+1} = y_{i+1}\ (y_i^2+\mu^2)^{1/2}. \tag{52}$$

The function $f_\mu : y \mapsto y/(y^2+\mu^2)$ is positive on $\mathbb{R}^+$ and attains its maximum at $x = \mu$, with $f_\mu(\mu) = 1/(2\mu)$; then, all the maps $T_{i,\mu}$ act on the hypercube $\mathcal{I}_\mu$ defined in (51). Moreover, in this context, a basis $\boldsymbol{b}$ is LLL(τ)-reduced if all the components y_i of $\boldsymbol{y}$ satisfy $y_i^2 \geq \tau^2 - \mu^2$. It is also $\Sigma(\theta)$-reduced with $\theta := (\tau^2-\mu^2)^{1/2}$ so that the algorithm stops as soon as $\boldsymbol{y}$ belongs to the hypercube $\mathcal{O}_{\mu,\tau}$ defined in (51).

Finally, the executions of the LLL(τ) algorithm, which are governed by the greedy strategy and where the vector $\boldsymbol{x}$ has all its components equal to a given value μ can be described with the trajectories of the dynamical system $\texttt{M2}(\mu, \tau)$. This system acts on the hypercube $\mathcal{I}_\mu$, and use maps $T_{i,\mu}$ described in (52) according to the greedy strategy. Its hole is the hypercube $\mathcal{O}_{\mu,\tau}$ defined in (51). The algorithm $\texttt{M2}(\mu, \tau)$ is described in Figure 12.

For instance, for $d = 3$, the system $\texttt{M2}(\mu, \tau)$ uses two maps T_1 and T_2, and, with $\theta := (\tau^2-\mu^2)^{1/2}$, one has

$$T_1(y_1, y_2) = \left(\frac{y_1}{y_1^2+\mu^2},\ \ y_2(y_1^2+\mu^2)^{1/2}\right) \qquad T_2(y_1, y_2) = \left(y_1(y_2^2+\mu^2)^{1/2},\ \ \frac{y_2}{y_2^2+\mu^2}\right).$$

The map T_1 is used for $y_1 < y_2$ and $y_1 < \theta$ and T_2 when $y_2 < y_1$ and $y_2 < \theta$.
The algorithm stops when $\min(y_1, y_2) > \theta$. Figure 12 shows an instance of a trajectory. At the beginning, the point (y_1, y_2) is close to $(0, 0)$, and this corresponds to a bad-reduced basis. When the point (y_1, y_2) attains the hole, the basis is reduced. When the point goes close to the fixed point $(1-\mu, 1-\mu)$, the trajectory becomes "slower".

The last result studies the distribution of the algorithm $\texttt{M2}(\mu, \tau)$ in the model $\mathcal{M}_{(\boldsymbol{r},a,d)}$ introduced in Section 5.10. Generally speaking, the distribution is geometric, with an explicit ratio, which depends on μ and the scaled valuation vector $\boldsymbol{s}$.

Proposition 6. [14] *Consider the model $\mathcal{M}_{(\boldsymbol{r},a,d)}$ associated with a irreducible valuation vector $\boldsymbol{s}$. For any $\mu > 0$ and $\tau < 1$, the number of iterations of the algorithm $\texttt{M2}(\mu, \tau)$ is asymptotically geometric, namely*

$$\mathbb{P}_{(\boldsymbol{r},a,d)}[K \geq k] = \Theta\left(\mu^{km(\boldsymbol{s})}\right) \qquad m(\boldsymbol{s}) := \min_i \frac{r_i}{i(d-i)}, \qquad (k \to \infty),$$

where the constant hidden in the Θ depends on μ and τ. It is uniform for $\mu \in [\mu_0, 1/2]$ and $\tau \in [0, \tau_0]$ for any $\mu_0 > 0$ and $\tau_0 < 1$.

Open question. *Describe the transition of the algorithm $\texttt{M2}(\mu, \tau)$ when $\mu \to 0$ and $\tau \to 1$.*

Input. A vector $\boldsymbol{y} \in \mathcal{I}_\mu$;
Output. A vector $\widehat{\boldsymbol{y}} \in \mathcal{O}_{\tau,\mu}$

While $\boldsymbol{y} \notin \mathcal{O}_{\tau,\mu}$ **do**
$i := \text{argmin}\{y_i\}$;
$\boldsymbol{y} := T_{i,\mu}(\boldsymbol{y})$;

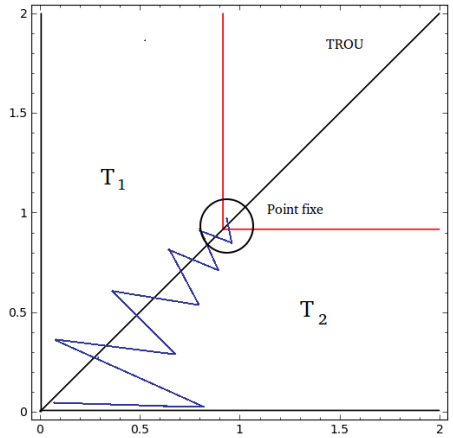


FIGURE 12. On the left, description of the dynamical system $\texttt{M2}(\mu, \tau)$ with parameters $\tau \leq 1$ and $\mu \in]0, 1/2[$. The sets $\mathcal{I}_\mu, \mathcal{O}_{\mu,\tau}$ are defined in (51). The map $T_{i,\mu}$ is defined in (52). On the right, an instance of a trajectory for $d = 3$.

5.13. **The simplified model** `M3`**.** There exists a natural extension of the model `M2` where the real parts x_i are no longer assumed to be constant. But the component x_i is assumed to be *independent* of the vector $\boldsymbol{y}$ and *randomly* chosen in the interval $[0, 1/2]$. Then, the model $\texttt{M2}(\mu, \tau)$ gives rise in this way to a random dynamical system $\texttt{M3}(\tau)$ which seems to be a good model for the LLL(τ) algorithm, when the real part $\boldsymbol{x}_i$ is no longer computed but randomly chosen. Unfortunately, this system seems also more difficult to deal with. It is probably convenient in a first study to consider that x_i is randomly drawn in some interval $[\mu_0, 1/2]$ (with $\mu_0 > 0$) and deal with $\tau < 1$.

5.14. **Historical and bibliographic notes.** There exist algorithms, described in [68] and [61] which exactly extend the Gauss algorithm for dimensions $d = 3$ or $d = 4$: they build directly a minimal basis and perform a linear number of iterations. The LLL algorithm was created in 1982 and described in [43]. The book [55] is completely devoted to the algorithm and its numerous applications. There are other lattice reduction algorithms which provide output bases of better quality, (with of course a worst complexity). In particular, Schnorr describes in [60] a complete hierarchy of such lattice reduction algorithms.

Probabilistic analysis of the algorithm is yet in its infancy. It began in the uniform input model, with the paper [19] which was followed and extended by the works [4, 5]. The new tentatives, with the introduction of the model with valuation, and the simplified models of execution are described in [47] [29] and [14].

REFERENCES

[1] K. AARDAL and F. EISENBRAND. *The LLL algorithm and integer programming*, Chapter 9 (293–314) in the book [55]
[2] M. AJTAI. *The worst-case behavior of Schnorr's algorithm approximating the shortest nonzero vector in a lattice*, Proceedings of the 35th Symposium on the Theory of Computing (STOC 2003), ACM, 2003, 396–406
[3] A. AKHAVI. *The optimal LLL algorithm is still polynomial in fixed dimension*, Theor. Comput. Sci. 297(1-3): 3-23 (2003)
[4] A. AKHAVI. *Random lattices, threshold phenomena and efficient reduction algorithms*, *Theoretical Computer Science*, 287 (2002) 359–385
[5] A. AKHAVI, J.-F. MARCKERT and A. ROUAULT. *On the Reduction of a Random Basis*, ESAIM Probability and Statistics Volume 13, January 2009, 437 - 458
[6] A. AKHAVI and B. VALLÉE. *Average bit-complexity of Euclidean algorithms*, in Proceedings of ICALP'2000 - Genève, 14 pages, Lecture Notes in Computer Science 1853, 373–387.

[7] K. I Babenko On a problem of Gauss. *Soviet Mathematical Doklady 19*, 1 (1978), pp 136–140.

[8] V. Baladi. *Positive Transfer operators and decay of correlations,* Advanced Series in non linear dynamics, World Scientific, 2000.

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

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

[11] V. Berthé, L. Lhote and B. Vallée *Probabilistic analyses of the plain multiple GCD algorithm*, accepted to Journal of Symbolic Computation, [50 p] to appear in 2016. http://dx.doi.org/10.1016/j.jsc.2015.08.007

[12] J. Bourdon, B. Daireaux, and B. Vallée. *Dynamical analysis of α-Euclidean Algorithms*, Journal of Algorithms 44 (2002) 246–285.

[13] E. Cesaratto, J. Clément, B. Daireaux, L. Lhote, V. Maume-Deschamps, and B. Vallée. *Regularity of the Euclid Algorithm: application to the analysis of fast gcd Algorithms*, Journal of Symbolic Computation, 44 (2009) 726–767

[14] J. Clément, M. Georgieva, L. Lhote and B. Vallée. *Modelling particular trajectories of the greedy LLL algorithm* in preparation

[15] P. Collet. Some ergodic properties of maps of the interval, *Dynamical systems, Proceedings of the first UNESCO CIMPA School on Dynamical and Disordered Systems* (Temuco, Chile, 1991), Hermann, 1996.

[16] D. Coppersmith. *Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities*, Journal of Cryptology, vol 10(4), 1997, 233–260

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

[18] B. Daireaux and B. Vallée. *Dynamical Analysis of the Parametrized Lehmer-Euclid Algorithm*, *Combinatorics, Probability, Computing*, pp 499–536 (2004).

[19] H. Daudé and B. Vallée. *An upper bound on the average number of iterations of the LLL algorithm.* TCS 123 (1994) 95–115.

[20] H. Daudé, P. Flajolet and B. Vallée. *An average-case analysis of the Gaussian algorithm for lattice Reduction.* Combinatorics, Probability and Computing (1997) 6, 397–433.

[21] H. Delange. *Généralisation du Théorème d'Ikehara, Ann. Sc. ENS*, (1954) 71, pp 213–242.

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

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

[24] P. Flajolet and B. Vallée. *Continued fraction algorithms, functional operators, and structure constants*, Theoretical Computer Science (1998), vol 194, 1–2, 1–34.

[25] P. Flajolet and B. Vallée. *Continued fractions, Comparison algorithms and Fine structure constants* Constructive, Experimental and Non-Linear Analysis, Michel Thera, Editor, Proceedings of Canadian Mathematical Society, Vol 27 (2000), 53–82

[26] P. Flajolet and R. Sedgewick. *Analytic combinatorics,* Cambridge University Press (2009).

[27] H. Heilbronn. *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.

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

[29] M. Georgieva *Analyse probabiliste de la réduction des réseaux euclidiens cryptographiques,* PhD, University of Caen, December 2013.

[30] C. Gentry. *The Geometry of Provable Security: some proofs of Security in which Lattices make a Surprise Appearance,* Chapter 12 (391–426) in the book [55]

[31] G. Hanrot. *LLL: a tool for Effective Diophantine Approximation*, Chapter 6 (215–264) in the book [55]

[32] G. Hanrot, X. Pujol and D. Stehl. *Analyzing Blockwise Lattice Algorithms using Dynamical Systems*, see the webpage of Damien Stehlé.

[33] Iosifescu, M. and Kraaicamp, C. *Metrical Theory of Continued Fractions.* (2002)

[34] A. I Khinchin. *Continued Fractions.* University of Chicago Press, Chicago, 1964. A translation of the Russian original published in 1935.

[35] J. Klüners. *The van Hoeij algorithm for factoring polynomials,* Chapter 8 (283-292) in the book [55]

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

[37] R. O Kuzmin. *Sur un problème de Gauss,* Atti del Congresso Internationale dei Matematici 6 (Bologna, 1928) pp 83-89.

[38] J. C. Lagarias. *Worst–case complexity bounds for algorithms in the theory of integral quadratic forms,* Journal of Algorithms 1, 2 (1980), 142–186.

[39] J. C. Lagarias and A. M. Odlyzko.*Solving low-density subset sum problems,* Journal ACM, 32(1) (1985), 229-246.

[40] A. Lasota and M. Mackey. *Chaos, Fractals and Noise; Stochastic Aspects of Dynamics,* Applied Mathematical Science 97, Springer (1994)
[41] D. H Lehmer. *Euclid's algorithm for large numbers.* Am. Math. Mon. (1938) 45 pp 227–233.
[42] P. Lévy. *Sur les lois de probabilité dont dépendent les quotients complets et incomplets d'une fraction continue.* Bull. Soc. Math. France 57 (1929) pp 178-194
[43] A. K. Lenstra, H. W. Lenstra, and L. Lovász. *Factoring polynomials with rational coefficients.* Mathematische Annalen 261 (1982), 513–534.
[44] H. W. Lenstra. *Flags and lattice basis reduction* European Congress of Mathematics, Barcelona, July 10-14, 2000 Volume II, Progress in mathematics, Springer, 37–53
[45] L. Lhote. *Computation of a class of Continued Fraction constants.* Proceedings of ALENEX-ANALCO'04, 199–210
[46] L. Lhote and B. Vallée. *Gaussian laws for the main parameters of the Euclid Algorithm*, Algorithmica (2008) 50, 497–554
[47] M. Madritsch and B. Vallée. *Modeling the LLL algorithm by sandpiles*, Proceedings of LATIN 2010, LNCS 6034 (2010) 267–281
[48] D. H Mayer. *On a ζ function related to the continued fraction transformation*, *Bulletin de la Société Mathématique de France 104* (1976), pp 195–203.
[49] D. H Mayer. *Continued fractions and related transformations*, In [10] pp. 175–222.
[50] D. H Mayer. *On the thermodynamic formalism for the Gauss Map*, *Commun. Math. Phys.* 130, pp 311-333 (1990)
[51] D. Mayer and G. Roepstorff. *On the relaxation time of Gauss's continued fraction map.* I. The Hilbert space approach, *Journal of Statistical Physics* 47, 1/2 (Apr. 1987), pp 149–171. II. The Banach space approach (transfer operator approach, *Journal of Statistical Physics* 50, 1/2 (Jan. 1988), pp 331–344.
[52] A. May. *Using LLL Reduction for solving RSA and Factorization Problems,* Chapter 10 (315-348) in the book [55]
[53] J. Hoffstein, N. Howgrave-Graham, J. Pipher, and W. Whyte. *NTRUEncrypt and NTRUSign*, Chapter 11 (349–390) in the book [55]
[54] P. Nguyen and D. Stehlé, *LLL on the average*, Proceedings of the 7th Algorithmic Number Theory Symposium (ANTS VII), Springer LNCS vol. 4076, (2006), 238–256
[55] P. Q. Nguyen and B. Vallée. *The LLL algorithm - Survey and Applications*, Springer, collection "Information Security & Cryptography series", 2010, 496 pages.
[56] G. J. Rieger. *Über die mittlere Schrittanzahl bei Divisionalgorithmen, Math. Nachr.* (1978) pp 157–180.
[57] G. J. Rieger. *Über die Schrittanzahl beim Algorithmus von Harris und dem nach nächsten Ganzen*, Archiv der Mathematik 34 (1980), pp 421–427.
[58] D. Ruelle. *Thermodynamic formalism,* Addison Wesley (1978)
[59] D. Ruelle. *Dynamical Zeta Functions for Piecewise Monotone Maps of the Interval*, vol. 4 of *CRM Monograph Series*, American Mathematical Society, Providence, 1994.
[60] C. P. Schnorr. *A Hierarchy of Polynomial Lattice Basis Reduction Algorithms*, Theoretical Computer Science, vol 53, (1987), 201–224
[61] I. Semaev. *A 3-dimensional lattice reduction algorithm*, Proceedings of the 2001 Cryptography and Lattices Conference (CALC'01), Springer LNCS 2146, 2001, 181–193
[62] Shallit, J. *Origins of the analysis of the Euclidean Algorithm*, Historia Mathematica 21 (1994) pp 401-419
[63] Schweiger, F. *Multidimensional Continued Fractions,* Oxford University Press, (2000)
[64] C.L. Siegel. *A mean value theorem in geometry of numbers*, Annals in Mathematics, 46(2) 340–347, 1945.
[65] Sorenson, J. *An analysis of Lehmer's Euclidean GCD Algorithm,* Proceedings of ISSAC 1995, pp 254–258
[66] D. Stehlé. *Floating Point LLL: Theoretical and Practical aspects*, Chapter 5 (179–215) in the book [55]
[67] B. Vallée. *Un problème central en géométrie algorithmique des nombres: la réduction des réseaux. Autour de l'algorithme LLL,* Informatique Théorique et Applications (RAIRO), 1989-3, pp 345-376. (English translation by E. Kranakis, CWI Quaterly, 1990-3, Amsterdam.)
[68] B. Vallée. *An affine point of view on minima finding in integer lattices of lower dimensions.* Proceedings of EUROCAL'87, Springer LNCS 378 (1987) 376–378
[69] B. Vallée. *Gauss' algorithm revisited.* Journal of Algorithms 12 (1991), 556–572.
[70] B. Vallée, M. Girault and P. Toffin. *How to guess ℓ-th roots modulo n by reducing lattices bases,* Proceedings of AAECC-88, Rome, Lectures Notes in Computer Science (357), 427–442 (1988)
[71] B. Vallée. *Generation of elements with small modular squares and provably fast integer factoring algorithms,* Mathematics of Computation, vol 56, 194, 823–849, 1991.
[72] B. Vallée. *Algorithms for computing signs of 2×2 determinants: dynamics and average–case analysis*, Proceedings of ESA'97 (5th Annual European Symposium on Algorithms) (Graz, September 97), LNCS 1284, 486–499 (1997)
[73] B. Vallée, *Dynamics of the Binary Euclidean Algorithm: Functional analysis and operators,* Algorithmica, vol 22 (4) (1998), 660–685.
[74] B. Vallée. *Digits and Continuants in Euclidean Algorithms. Ergodic Versus Tauberian Theorems*, Journal de Théorie des Nombres de Bordeaux 12 (2000) pp 531-570.

[75] B. Vallée. *Opérateurs de Ruelle-Mayer généralisés et analyse des algorithmes d'Euclide et de Gauss*, Acta Arithmetica 81.2 (1997) pp 101–144.
[76] B. Vallée. *Dynamical Analysis of a Class of Euclidean Algorithms*, Theoretical Computer Science 297 1-3, 2003, 447–486
[77] B. Vallée. *Euclidean Dynamics*, Discrete and Continuous Dynamical Systems, 15 (1) May 2006, 281–352.
[78] B. Vallée and A. Vera. *Lattice Reduction in two dimensions: analyses under realistic probabilistic models*, Proceedings of the AofA'07 conference, Discrete Mathematics and Theoretical Computer Science, Proc. AH, 2007, 181-216.
[79] B. Vallée and A. Vera. *Probabilistic analyses of lattice reduction algorithms*, chapter 3 (71-144) of the book [55]
[80] A. Vera, *Analyse en moyenne de l'algorithme de Gauss, et applications à l'analyse de l'algorithme LLL*, PhD Thesis, University of Caen, 2009.
[81] G. Villard. *Parallel lattice basis reduction.* Proceedings of International Symposium on Symbolic and Algebraic Computation, Berkeley California USA. ACM Press, July 1992.
[82] E. Wirsing. *On the theorem of Gauss–Kusmin–Lévy and a Frobenius–type theorem for function spaces.* Acta Arithmetica 24 (1974) pp 507–528.