ANALYSIS of EUCLIDEAN ALGORITHMS

An Arithmetical Instance of Dynamical Analysis

Dynamical Analysis :=

Analysis of Algorithms + Dynamical Systems

Brigitte Vallée (CNRS and Université de Caen, France)

Results obtained with :

Ali Akhavi, Viviane Baladi, Jérémie Bourdon, Eda Cesaratto, Julien Clément, Benoît Daireaux, Hervé Daudé, Philippe Flajolet, Loïck Lhote, Véronique Maume, Antonio Vera.

Dynamical Analysis –main principles.

Input.- A discrete algorithm.

Step 1.- Extend the discrete algorithm into a continuous process, i.e. a dynamical system. (X, V) X compact, $V : X \to X$, where the discrete alg. gives rise to particular trajectories.

Step 2.- Study this (continuous) dynamical system, via its generic trajectories. A main tool: the transfer operator.

Step 3.- Coming back to the algorithm: Use the transfer operator as a generating operator, and prove that the particular trajectories due to the algorithm behave as the generic trajectories.

Output.- Probabilistic analysis of the Algorithm.

Plan of the talk.

I – Four types, six instances of Euclidean algorithms

II – The average-case analysis: The results.

III – The dynamical systems underlying the algorithms.

IV – The method: Dynamical Analysis

V – Two or three instances of the extension of the method.

I – Four types, six instances of Euclidean algorithms

The Euclid Algorithm: the grand father of all the algorithms.

On the input (u, v), it computes the gcd of u and v, together with the Continued Fraction Expansion of u/v. $u_0 := v;\ u_1 := u; u_0 \geq u_1$

$$\left\{\begin{array}{ccccc} u_0 & = & m_1 u_1 & + & u_2 \qquad\qquad 0 < u_2 < u_1 \\ u_1 & = & m_2 u_2 & + & u_3 \qquad\qquad 0 < u_3 < u_2 \\ \ldots & = & \ldots & + & \\ u_{p-2} & = & m_{p-1} u_{p-1} & + & u_p \qquad\qquad 0 < u_p < u_{p-1} \\ u_{p-1} & = & m_p u_p & + & 0 \qquad\qquad u_{p+1} = 0 \end{array}\right\}$$

u_p is the gcd of u and v, the m_i's are the digits. p is the depth.

CFE of $\dfrac{u}{v}$:

$$\frac{u}{v} = \cfrac{1}{m_1 + \cfrac{1}{m_2 + \cfrac{1}{\ddots + \cfrac{1}{m_p}}}},$$

The extended Euclid Algorithm

also returns the Bezout pair (r, s) for which $\gcd(u, v) = rv + su$.

It computes the sequence s_i defined by

$$s_0 = 0, \qquad s_1 = 1, \qquad s_i = s_{i-2} - s_{i-1} \cdot m_{i-1}, \quad 0 \leq i < p.$$

The last element s_p is the Bezout coefficient s.

Used for computing modular inverses: crucial in cryptography.

A Euclidean algorithm:=

any algorithm which performs a sequence of divisions $v = mu + r$.

Various possible types of Euclidean divisions

– MSB divisions [directed by the Most Significant Bits]

shorten integers on the left,

and provide a remainder r smaller than u,

(w.r.t the usual absolute value), i.e. with more zeroes on the left.

– LSB divisions [directed by the Least Significant Bits]

shorten integers on the right,

and provide a remainder r smaller than u

(w.r.t the dyadic absolute value), i.e. with more zeroes on the right.

– Mixed divisions

shorten integers both on the right and on the left,

with new zeroes both on the right and on the left.

Instances of MSB Algorithms.

– Variants according to the position of remainder r,

By Default:	$v = mu + r$	with	$0 \le r < u$
By Excess:	$v = mu - r$	with	$0 \le r < u$
Centered:	$v = mu + \epsilon r$	with	$\epsilon = \pm 1, \quad 0 \le r \le u/2$

– Subtractive Algorithm :

A division with quotient m can be replaced by m subtractions

`While` $v \geq u$ `do` $v := v - u$

An instance of a Mixed Algorithm.

The Subtractive Algorithm,

where the zeroes on the right are removed from the remainder defines the Binary Algorithm.

Subtractive Gcd Algorithm.	Binary Gcd Algorithm.
Input. u, v; $v \geq u$	**Input.** u, v odd; $v \geq u$
`While` $(u \neq v)$ `do`	`While` $(u \neq v)$ `do`
`While` $v > u$ `do`	`While` $v > u$ `do`
	$k := \nu_2(v - u)$;
$v := v - u$	$v := \dfrac{v - u}{2^k}$;
Exchange u and v.	Exchange u and v.
Output. u (or v).	**Output.** u (or v).

The 2-adic valuation ν_2 counts the number of zeroes on the right

An instance of a LSB Algorithm.

On a pair (u, v) with v odd and u even,

with $\nu_2(u) = k$, of the form $u := 2^k\, u'$

the LSB division produces

– a quotient a odd, with $|a| < 2^k$

– and a remainder r with $\nu_2(r) > k$, of the form $r := 2^k\, r'$,

and writes $\qquad v = a \cdot u' + 2^k \cdot r'.$

The pair (r', u') satisfies

$$\nu_2(r') > \nu_2(u') = 0 \text{ and } \gcd(u, v) = \gcd(r', u').$$

It will be the new pair for the next step.

An execution of the LSB Algorithm on $(72001, 2011176)$

i	u_i [base 2]	a_i	k_i
1	111101011000000101000	-3	3
2	11001001101101010000	1	1
3	110000110001010000000	1	3
4	10011000111100000000	-1	1
5	111010010101000000000	-1	1
6	110000010010000000000	1	1
7	100010001100000000000	-1	1
8	1000001011000000000000	1	1
9	1100000000000000	1	2
10	1000001000000000000000	-1	1
11	100010000000000000000	1	1
12	110000000000000000000	-5	3
13	10000000000000000000000	3	2

Comparison for five algorithms on the input $(2011176, 72001)$

Evolution of the remainders

Standard	Centered	By-Excess	Binary	LSB
67149	4852	4852	44849	51637
4852	779	779	1697	12485
4073	178	601	1697	2447
779	67	423	125	3733
178	23	245	125	1545
67	2	67	9	547
44	–	23	9	523
23	–	2	5	3
19	–	–	–	65
4	–	–	–	17
3	–		–	3

I – Four types, Six instances of Euclidean algorithms

II – The average-case analysis: The results.

III – The dynamical systems underlying the algorithms.

IV – The method: Dynamical Analysis

V – Two or three instances of the extension of the method.

A general framework.

Each division–step of each algorithm uses a "digit" $d = (m, \epsilon, a, b)$, changes the old pair (u, v) into the new pair (r', u') as

$$u = 2^a \cdot u', \qquad v = m \cdot u' + \epsilon \cdot 2^b \cdot r'.$$

On integer pairs, it uses the matrix transformation $M_{[d]}$

$$\begin{pmatrix} u \\ v \end{pmatrix} = M_{[d]} \begin{pmatrix} r' \\ u' \end{pmatrix}, \qquad \text{with} \qquad M_{[d]} := \begin{pmatrix} 0 & 2^a \\ \epsilon\, 2^b & m \end{pmatrix}$$

and, on rationals (the old $x = u/v$ and the new $y = r'/u'$), it uses the LFT $h_{[d]}$,

$$x = h_{[d]}(y) \qquad \text{with} \quad h_{[d]}(y) = \frac{2^a}{m + \epsilon\, 2^b y}.$$

Then $|\det h_{[d]}| = 2^{a+b}$ involves the total number $a + b$ of binary shifts.

A generic execution.

On the input pair $(u, v) = (u_1, u_0)$, it is of the form

$$\left\{\begin{array}{lllllll} u_1 & := & 2^{-a_1} u_1, & u_0 & = & m_1\, u_1 & + & \epsilon_1\, 2^{b_1}\, u_2, \\ u_2 & := & 2^{-a_2} u_2, & u_1 & = & m_2\, u_2 & + & \epsilon_2\, 2^{b_2}\, u_3, \\ & & \ldots, & & & \ldots & & \\ u_i & := & 2^{-a_i} u_i, & u_{i-1} & = & m_i\, u_i & + & \epsilon_i\, 2^{b_i}\, u_{i+1} \\ & & \ldots, & & & \ldots & & \\ u_p & := & 2^{-a_p} u_p, & u_{p-1} & = & m_p\, u_p & + & \epsilon_p\, 2^{b_p}\, u_{p+1} \end{array}\right\},$$

and uses the sequence of digits $d_i := (m_i, \epsilon_i, a_i, b_i)$.

It stops at the p–th iteration with $u_{p+1} = \eta \cdot u_p$ $[\eta = 0$ or $\eta = 1]$.

Then $\gcd(u, v) = u_p$.

Cost of an execution: the additive case.

Given a positive step–cost c defined on the set $\mathcal{D}$ of digits, consider the total cost C defined on the input (u, v) in an additive way as

$$C(u, v) := \sum_{i=1}^{p} c(d_i), \qquad\qquad d_i := (m_i, \epsilon_i, a_i, b_i)$$

The step–cost c is of moderate growth, when $c(d) = O(\log m)$

Main costs of moderate growth.

- if $c \equiv 1$, then $C = P$ is the number of iterations
- if c is the characteristic function $\mathbf{1}_{d_0}$ of a given digit d_0, then C is the number of occurrences of d_0 in the CF.
- if $c(d) = a + b$, then C is the total number of binary shifts.
- if $c(d) = \ell(m)$, the binary length of digit m, then C is the encoding length of the continued fraction.

An important (non additive) cost.

The most precise cost: the (naive) bit–complexity

$$B(u,v) := \sum_{i=1}^{p} \ell(m_i) \cdot \ell(u_i)$$

which involves digit sizes $\ell(d_i)$ together with remainder sizes $\ell(u_i)$...
in a multiplicative way ...

An Important Question.

Compare the behaviour of these various Euclidean algorithms with respect to various costs, and particularly the bit–complexity.

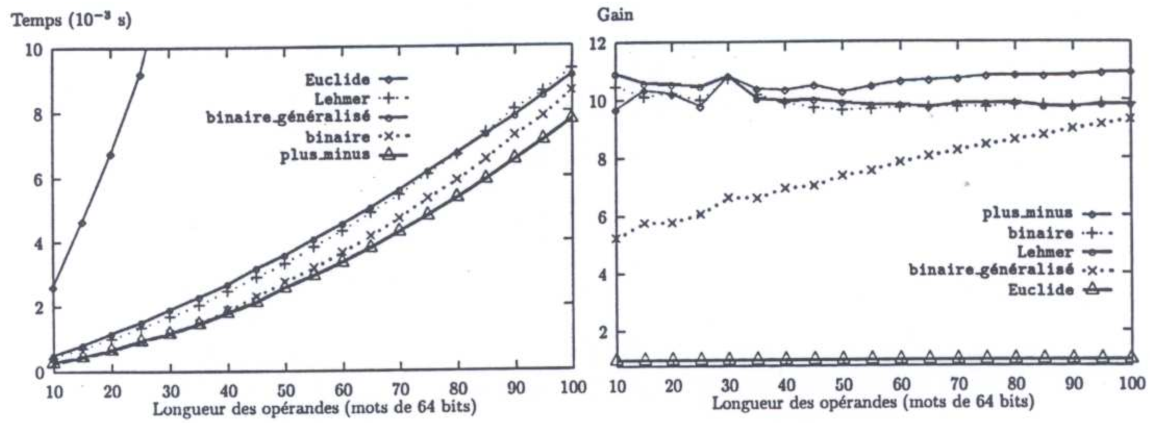


FIG. 9.20 – *Temps et gain par rapport à* **Euclide** *(station DEC).*

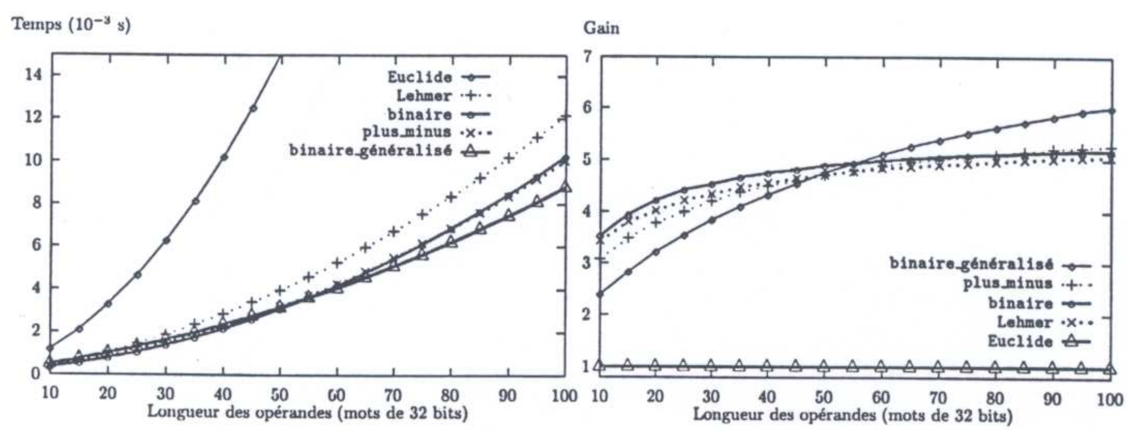


FIG. 9.21 – *Temps et gain par rapport à* **Euclide** *(station SUN).*

The analysis of the Euclidean Algorithms.

For an input (u, v), the length $|(u, v)|$ is defined by $|(u, v)|^2 := (u^2 + v^2)$,

Its size is $L(u, v) := \ell(|(u, v)|)$

When the set of all possible inputs (u, v) of the algorithm is Ω,

the algorithm is studied on the set

$\Omega_n := \{(u, v) \in \Omega; \quad L(u, v) = n\}$ for $n \to \infty$.

Previous results, mostly in the average-case,

only for parameter P, and specific to particular algorithms...

well–described in Knuth's book (Tome II)

Heilbronn, Dixon, Rieger (70): Standard and Centered Alg.

Yao and Knuth (75): Subtractive Alg.

Brent (78): Binary Alg (partly heuristic),

Hensley (94) : A distributional study for the Standard Alg.

Stehlé and Zimmermann (05) : LSB Alg (experiments)

Then Dynamical Analysis [Our group, 1995 $\rightarrow$?] provides

– a complete classification into two classes,

  – the Fast Class ={Standard, Centered, Binary, LSB},

  – the Slow Class = {By-Excess, Subtractive}.

– an average-case analysis of a broad class of costs,

  – all the additive costs,

  – and also the bit–complexity.

– a distributional analysis of a subclass of the Fast Class,

the Good Class = {Standard, Centered}.

Asymptotic gaussian laws hold for:

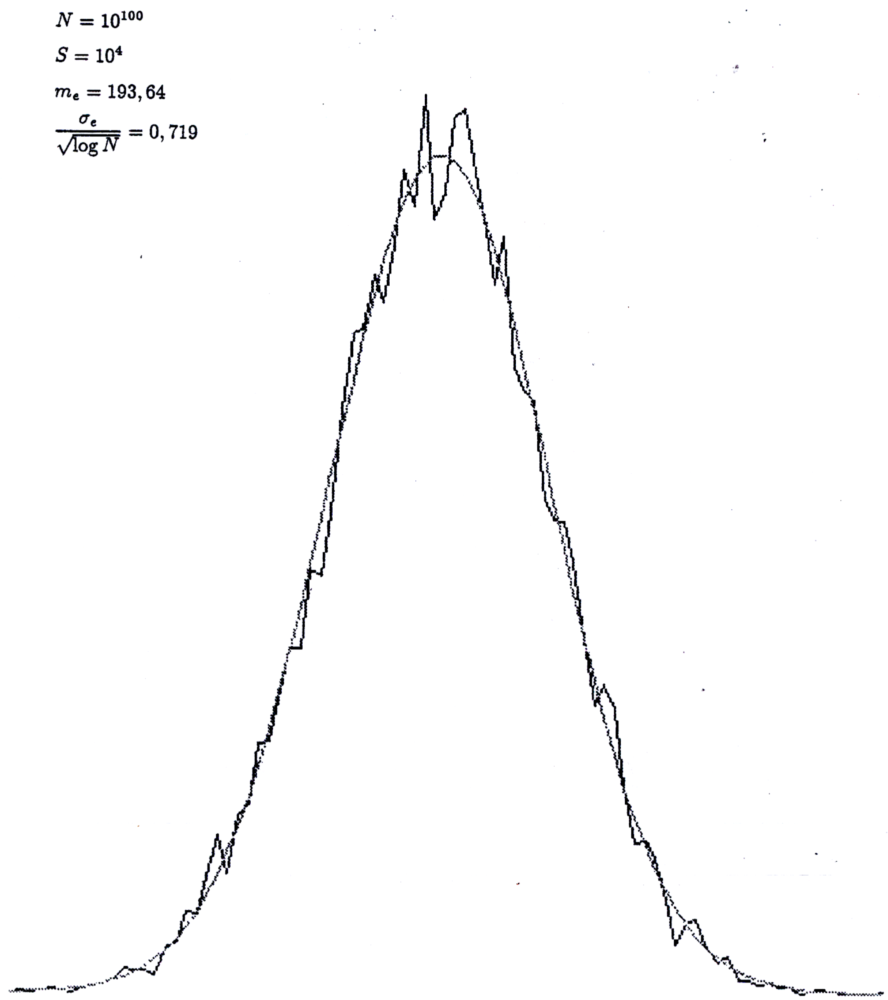
– P, and additive costs of moderate growth,

– the remainder size $\log u_i$ for $i \sim \delta P$, the stopping time P_δ

– the bit-complexity of the extended Alg.

We “prove” experimental results.

Here, an histogram of the number of iterations of the Standard Alg...



Here, focus on average-case results ($n :=$ input size)

– For the Fast Class ={Standard, Centered, Binary, LSB } ,

– the mean values of costs P, C are linear wrt n,

– the mean bit-complexity is quadratic.

$$\mathbb{E}_n[P] \sim \frac{2\log 2}{h(\mathcal{S})} n, \qquad \mathbb{E}_n[C] \sim \frac{2\log 2}{h(\mathcal{S})} \mu[c]\, n, \qquad \mathbb{E}_n[B] \sim \frac{\log 2}{h(\mathcal{S})} \mu[\ell]\, n^2.$$

$h(\mathcal{S})$ is the entropy of the system, $\mu[c]$ the mean value of step–cost c.

– Moreover, these costs are concentrated: $\mathbb{E}_n[C^k] \sim \mathbb{E}_n[C]^k$

– For the Slow Class = {By-Excess, Subtractive},

– the mean values of costs P, C are quadratic,

– the mean bit-complexity of B is cubic,

– the moments of order $k \geq 2$ are exponential: $\mathbb{E}_n[C^k] = \Theta(2^{n(k-1)})$.

The main constant $h(\mathcal{S})$ is the entropy of the Dynamical System.

A well-defined mathematical object, computable.

– Related to classical constants for the first two algs

$$h(\mathcal{S}) = \frac{\pi^2}{6\log 2} \sim 2.37 \text{ [Standard]}, \quad h(\mathcal{S}) = \frac{\pi^2}{6\log\phi} \sim 3.41 \text{ [Centered]}.$$

– For the LSB alg, $h(\mathcal{S}) = 4 - 2\gamma \sim 3.91$ involves the Lyapounov exponent γ of the set of random matrices, where

$$N_{a,k} = \frac{1}{2^k}\begin{pmatrix} 0 & 2^k \\ 2^k & a \end{pmatrix} \text{ with } k \geq 1, a \text{ odd}, |a| < 2^k \text{ is taken with prob. } 2^{-2k},$$

– For the Binary alg, $h(\mathcal{S}) = \pi^2 f(1) \sim 3.6$ involves the value $f(1)$ of the unique density which satisfies the functional equation

$$f(x) = \sum_{k\geq 1} \sum_{\substack{a \text{ odd} \\ 1\leq a<2^k}} \left(\frac{1}{2^k x + a}\right)^2 f\left(\frac{1}{2^k x + a}\right)$$

Precise comparisons between the four Fast Algorithms

Algs	Nb of iterations	Bit-complexity
Standard	$0.584\, n$	$1.242\, n^2$
Centered	$0.406\, n$	$1.126\, n^2$
(Ind.) Binary	$0.381\, n$	$0.720\, n^2$
LSB	$0.511\, n$	$1.115\, n^2$

I – Four types, Six instances of Euclidean algorithms

II – The average-case analysis: The results.

III – The dynamical systems underlying the algorithms.

IV – The method: Dynamical Analysis

V – Two or three instances of the extension of the method.

The Euclidean dynamical system (I).

The trace of the execution of the Euclid Algorithm on (u_1, u_0) is:

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

Replace the integer pair (u_i, u_{i-1}) by the rational $x_i := \dfrac{u_i}{u_{i-1}}$.

The division $u_{i-1} = m_i u_i + u_{i+1}$ is then written as

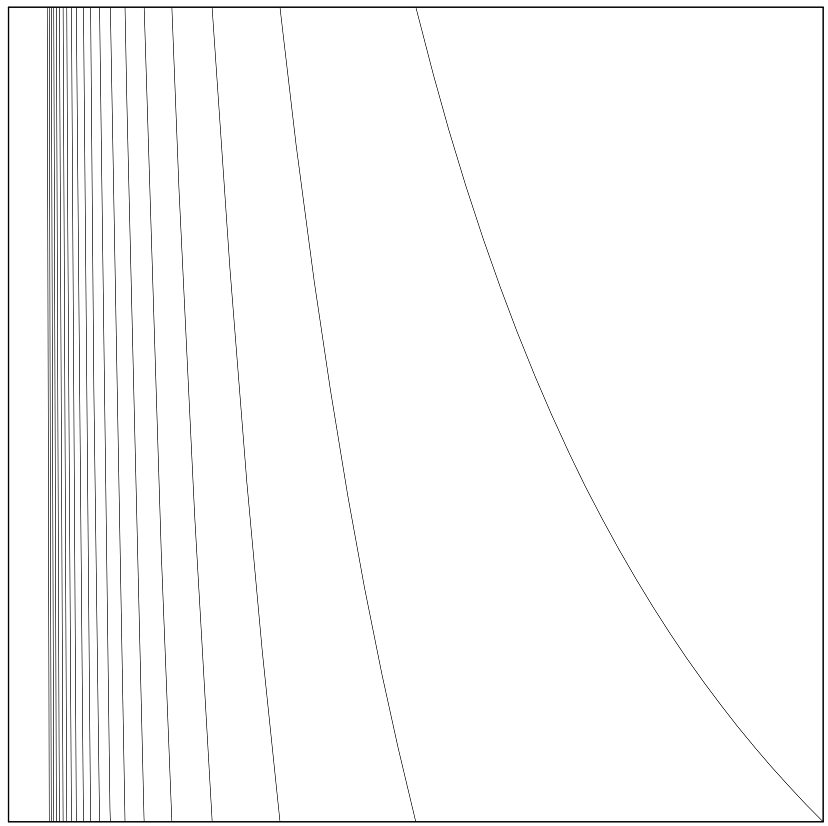
$$x_{i+1} = \frac{1}{x_i} - \left\lfloor \frac{1}{x_i} \right\rfloor \qquad \text{or} \qquad x_{i+1} = T(x_i), \qquad \text{where}$$

$$T : [0,1] \longrightarrow [0,1], \quad T(x) := \frac{1}{x} - \left\lfloor \frac{1}{x} \right\rfloor \quad \text{for} \ \ x \neq 0, \quad T(0) = 0$$

An execution of the Euclidean Algorithm $(x, T(x), T^2(x), \ldots, 0)$

$=$ A rational trajectory of the Dynamical System $([0,1], T)$

$=$ a trajectory that reaches 0.



The Euclidean dynamical system (II).

A dynamical system with a denumerable system of branches $(T_{[m]})_{m\geq 1}$,

$$T_{[m]} :]\frac{1}{m+1}, \frac{1}{m}[\longrightarrow]0,1[, \qquad T_{[m]}(x) := \frac{1}{x} - m$$

The set $\mathcal{H}$ of the inverse branches of T is

$$\mathcal{H} := \{ \ h_{[m]} :]0,1[\longrightarrow]\frac{1}{m+1}, \frac{1}{m}[; \qquad h_{[m]}(x) := \frac{1}{m+x} \}$$

The set $\mathcal{H}$ builds one step of the CF's.

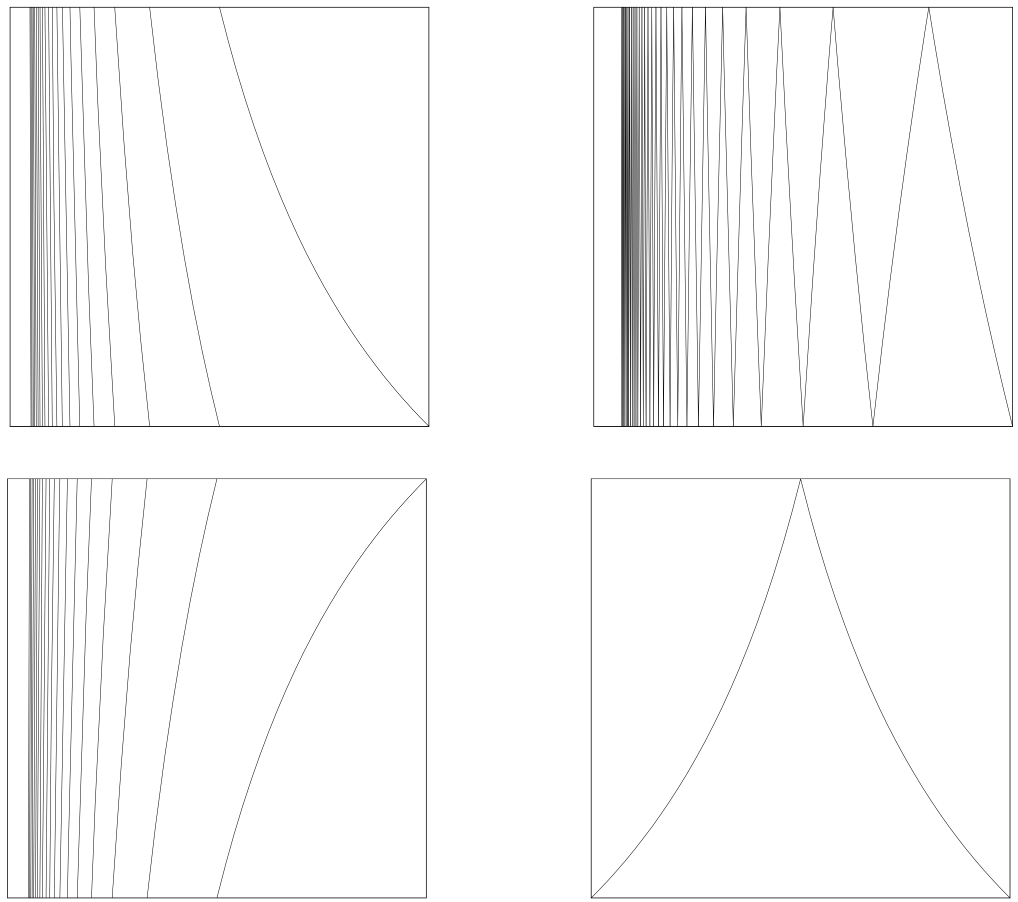
The set $\mathcal{H}^n$ of the inverse branches of T^n builds CF's of depth n.

The set $\mathcal{H}^\star := \bigcup \mathcal{H}^n$ builds all the (finite) CF's.

$$\frac{u}{v} = \cfrac{1}{m_1 + \cfrac{1}{m_2 + \cfrac{1}{\ddots + \cfrac{1}{m_p}}}} = h_{[m_1]} \circ h_{[m_2]} \circ \ldots \circ h_{[m_p]}(0)$$

For each MSB Alg., replace the rational u/v by a generic real x:

A continuous dynamical system extends each discrete division



Above, Standard and Centered; On the bottom, By-Excess and Subtractive.

On the bottom, there are indifferent points : $x = 1$ or 0, for which $T(x) = x, |T'(x)| = 1$.

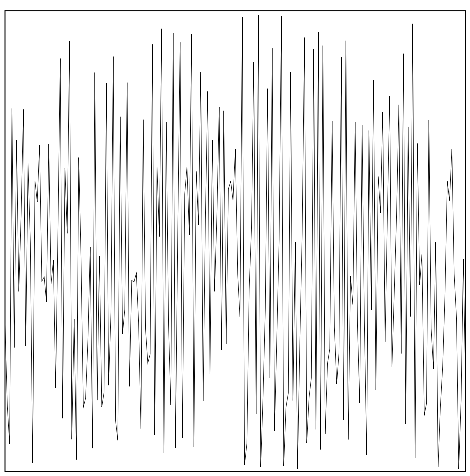
Dynamical Systems relative to MSB Algorithms.

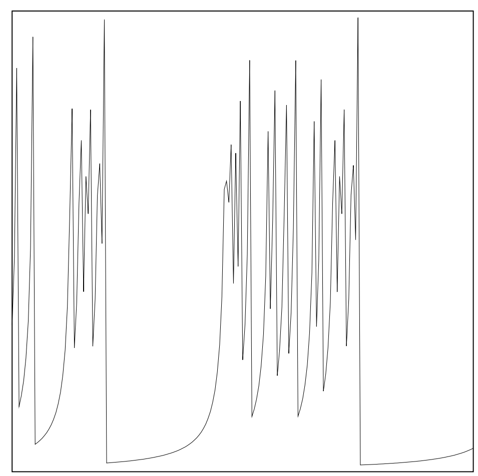
Key Property : Expansiveness of branches

$$|T'(x)| \geq \rho > 1 \text{ for all } x \text{ in } \mathcal{I}$$

When true, this implies a chaotic behaviour for trajectories. The associated algos are Fast and belong to the Good Class

When this condition is violated at only one indifferent point, this leads to intermittency phenomena. The associated algos are Slow.





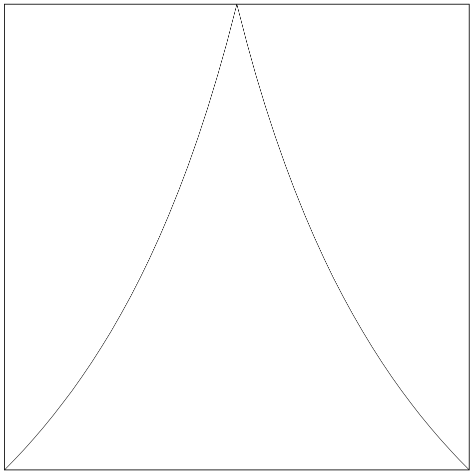
Chaotic Orbit [Fast Class], Intermittent Orbit [SlowClass].

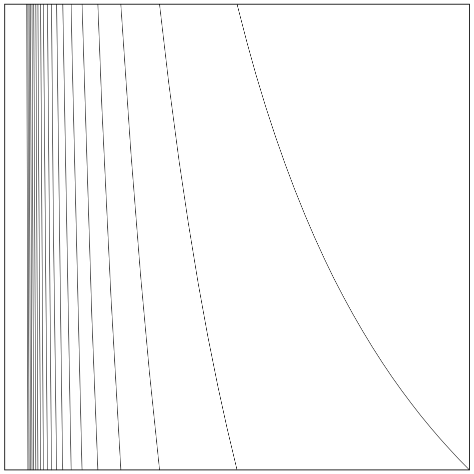
Induction Method

For a DS (I,T) with a "slow" branch relative to a slow interval J, contract each part of the trajectory which belongs to J into one step.

This (often) transforms the slow DS (I,T) into a fast one (I,S):

$$\text{While } x \in J \text{ do } x := T(x);$$
$$S(x) := T(x);$$





The Induced DS of the Subtractive Alg = the DS of the Standard Alg.

The Dynamical Systems relative to the other two algorithms, the Binary Algorithm and the LSB Algorithm.

These algorithms use the 2–adic valuation ν

.... only defined on rationals.

The 2–adic valuation ν is extended to a real random variable ν with

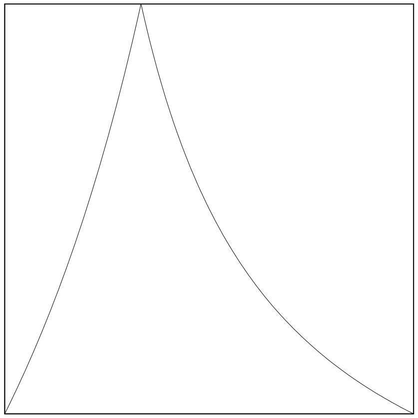
$$\mathbb{P}[\nu = k] = 1/2^k \qquad \text{for} \qquad k \geq 1.$$

This gives rise to probabilistic dynamic systems.

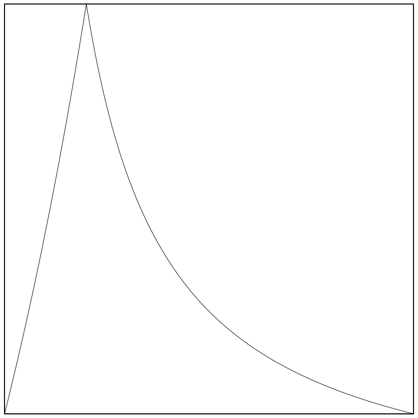
The Binary Dynamical System.

First, the probabilistic version of the Algorithm with

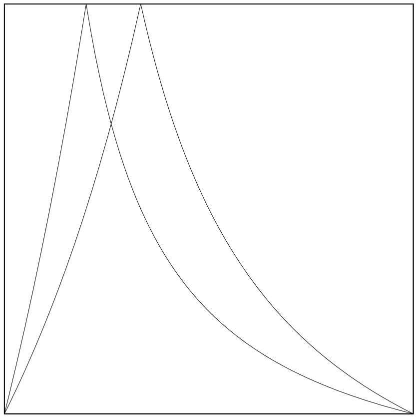
$$\mathbb{P}[\nu = k] = 1/2^k \qquad \text{for} \quad k \geq 1.$$



$k = 1$



$k = 2$



$k = 1$ and $k = 2$

Second, the induced dynamical system, where, now,
the probabilistic choice depends on the position of real x.

Subtractive Gcd Algorithm.	Binary Gcd Algorithm.
Input. u, v; $v \geq u$	**Input.** u, v odd; $v \geq u$
`While` $(u \neq v)$ `do`	`While` $(u \neq v)$ `do`
`While` $v > u$ `do`	`While` $v > u$ `do`
	$k := \nu_2(v-u)$;
$v := v - u$	$v := \dfrac{v-u}{2^k}$;
Exchange u and v.	Exchange u and v.
Output. u (or v).	**Output.** u (or v).

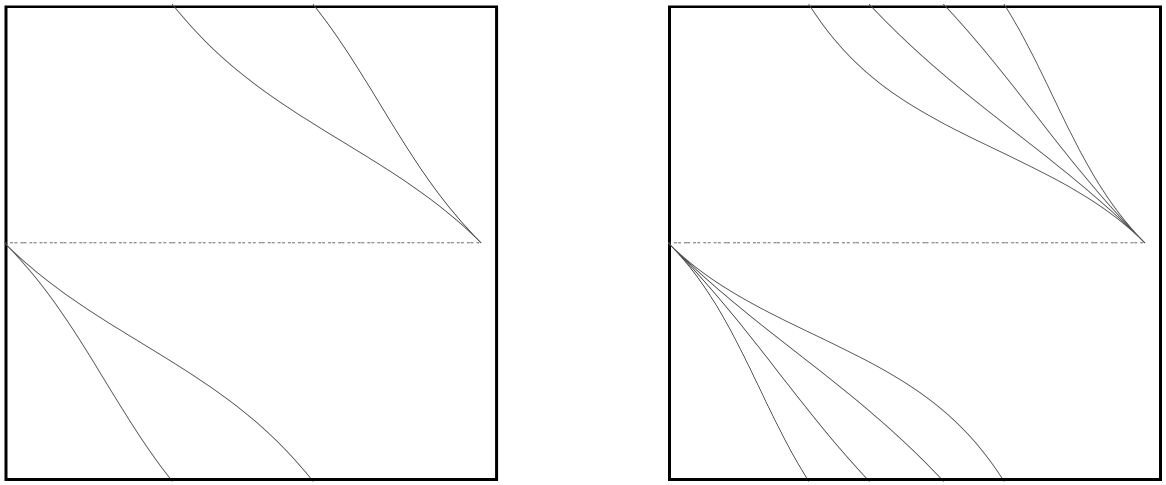
The 2-adic valuation ν_2 counts the number of zeroes on the right

The LSB Dynamical System.

Here, the remainders are not decreasing,

so that the rationals $x = u/v$ may belong to the whole $\mathbb{R}$.

Using the tangent map leads to work inside $J = [-\pi/2, +\pi/2]$...



The DS relative to the LSB Alg.

On the left, for $k = 1[a = \pm 1]$ – On the right, for $k = 2[a = \pm 1, \pm 3]$.

The probabilistic choice does not depend on the position of x.

This defines a system of iterated functions.

I – Four types, Six instances of Euclidean algorithms

II – The average-case analysis: The results.

III – The dynamical systems underlying the algorithms.

IV – The method: Dynamical Analysis

V – Two or three instances of the extension of the method.

General principles of Dynamical Analysis.

Two objects:

The (discrete) Algorithm, the (continuous) Dynamical System

Two tools:

The generating function, The transfer operator

And their relations:

Geometric properties of the Dynamical System
$\Downarrow$
Spectral properties for the Transfer Operator in a convenient functional space.
$\Downarrow$
Analytical properties of the (Dirichlet) Gen. Functions
$\Downarrow$
Asymptotic Analysis of the Algorithm

The Dirichlet series.

If Ω is the whole set of inputs, the Dirichlet generating function

$$S_C(s) = \sum_{(u,v)\in\Omega} \frac{C(u,v)}{|(u,v)|^{2s}} = \sum_{m\geq 1} \frac{c_m}{m^{2s}} \qquad \text{with } c_m := \sum_{\substack{(u,v)\in\Omega \\ |(u,v)|=m}} C(u,v)$$

is used for expressing the mean value $\mathbb{E}_n[C]$ of C on Ω_n, since

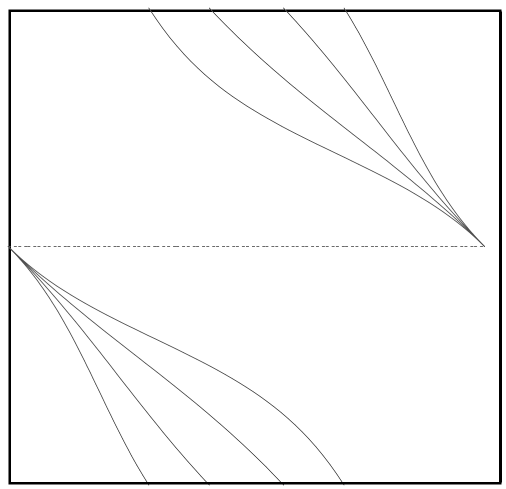
$$\mathbb{E}_n[C] = \frac{1}{|\Omega_n|} \sum_{m|\ell(m)=n} c_m.$$

For the asymptotics of $\mathbb{E}_n[C]$....

we need to obtain an alternative expression for $S_C(s)$, from which the position and the nature of singularities of $S_C(s)$ become apparent.

The density transformer $\mathbf{H}$ expresses the new density f_1 as a function of the old density f_0, as $f_1 = \mathbf{H}[f_0]$. It involves the set $\mathcal{H}$

$$\mathbf{H}[f](x) := \sum_{h \in \mathcal{H}} \delta_h \cdot |h'(x)| \cdot f \circ h(x) \qquad (\text{here, } \delta_h = \mathbb{P}[h])$$



With a cost $c : \mathcal{H} \to \mathbf{R}^+$, and a parameter s,

it gives rise to the weighted transfer operator

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

Relation between the transfer operator and the Dirichlet series.

Due to the fact that branches are LFT's,

There is an alternative expression for the Dirichlet series

$$S_C(s) := \sum_{(u,v)\in\Omega} \frac{C(u,v)}{|(u,v)|^{2s}} = (I - \mathbf{H}_s)^{-1} \circ \mathbf{H}_s^{[c]} \circ (I - \mathbf{H}_s)^{-1}[1](\eta)$$

as a function of two transfer operators : the weighted one

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

and the quasi-inverse $(I - \mathbf{H}_s)^{-1}$ of the plain transfer operator $\mathbf{H}_s$,

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

Singularities of $s \mapsto (I - \mathbf{H}_s)^{-1}$ related to spectral properties of $\mathbf{H}_s$

..... on a convenient functional space .... which depends on the DS (and the algo)...

Expected spectral properties of $\mathbf{H}_s$

(i) *UDE* and *SG* for s near 1:

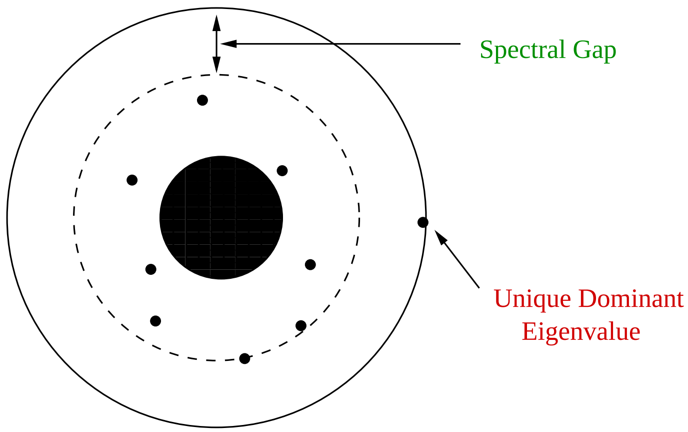
UDE – Unique dominant eigenvalue $\lambda(s)$

with $\lambda(1) = 1$

SG – Existence of a spectral gap

(ii) Aperiodicity: for $\Re s = 1, s \neq 1$,

the spectral radius of $\mathbf{H}_s$ is < 1



On which functional space?

The answer depends on the DS,

and thus on the algorithm....

The functional spaces where the triple $UDE + SG + \text{Aperiodicity}$ holds.

Algs	Geometry of branches	Convenient Functional space
Good Class (Standard, Centered)	Contracting	$\mathcal{C}^1(\mathcal{I})$
Binary	Not contracting	The Hardy space $\mathcal{H}(\mathcal{D})$
LSB	Contracting on average	Various spaces: $\mathcal{C}^0(J), \mathcal{C}^1(J)$ Hölder $\mathbb{H}_\alpha(J)$
Slow Class (Subtractive, By-Excess)	An indifferent point	Induction $+ \mathcal{C}^1(\mathcal{I})$

In each case, the aperiodicity holds since the branches have not " all the same form".

The triple $UDE + SG +$ Aperiodicity entails good properties for $(I - \mathbf{H}_s)^{-1}$, sufficient for applying Tauberian Theorems to $S_C(s)$.

$s = 1$ is the only pole

on the line $\Re s = 1$

Expansion near the pole $s = 1$

$$(I - \mathbf{H}_s)^{-1} \sim \frac{a}{s-1}$$

s=1

Half–plane of convergence $\Re s > 1$

No hypothesis needed

on the half–plane $\Re s < 1$.

We have then described the general framework

Geometric properties of the Dynamical System
$\Downarrow$
Spectral properties for the Transfer Operator
in a convenient functional space.
$\Downarrow$
Analytical properties of the (Dirichlet) Gen. Functions
$\Downarrow$
Asymptotic Analysis of the Algorithm

and applied it to the average-case analysis
of a class of EUCLID Algorithms....

Here, we have used the transfer operator $\mathbf{H}_s$ of the underlying DS
and studied it for complex numbers s with $\Re s \geq 1$.

I – Four types, Six instances of Euclidean algorithms

II – The average-case analysis: The results.

III – The dynamical systems underlying the algorithms.

IV – The method: Dynamical Analysis

V – Two or three instances of the extension of the method.

Three extensions.

– Distributional analysis of the Euclid algorithms

– Analysis of Fast variants of the Euclid Algorithms

Use the same transfer operator $\mathbf{H}_s$, with its behaviour for $\Re s < 1$

A vertical strip free of poles with polynomial growth for $(I - \mathbf{H}_s)^{-1}$

– Study of the Gauss algorithm (for reducing lattices) for $n = 2$

Use of an extension of the transfer operator $\mathbf{H}_s$,

which operates on functions of two variables, for $s \sim 2$

A central tool for reducing lattices in general dimensions n

Mean bit–complexity of fast variants of the Euclid Algorithm

Main principles of Fast Euclid Algorithms:

– Based on a Divide and Conquer principle with two recursive calls.

– Study "slices" of the original Euclid Algorithm

– begin when the data has already lost δn bits.

– end when the data has lost γn additional bits.

– Replace large divisions by small divisions and large multiplications.

– Use fast multiplication algorithms (based on the FFT)

of complexity $n \log n \, a(n)$

with $a(n) = \log \log n$ [Schönhage Strassen]

now $a(n) = 2^{O(\log^\star n)}$ [Fürer, 2007]

with $\log^\star n =$ the smallest integer k for which $\log^{(k)} n < 1$

We obtain the mean bit-complexity of (variants of) these algorithms

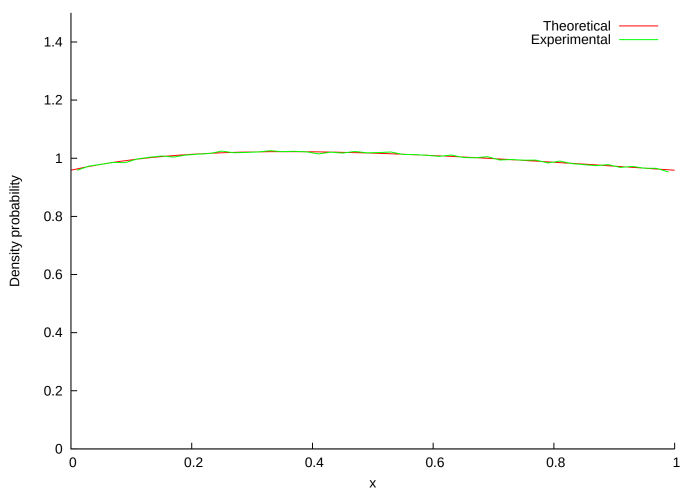
$$\Theta(n(\log n)^2 a(n))$$

with a precise estimate of the hidden constants

Analysis based on the answer to the question:

What is the distribution of the data

when they have already lost a fraction δ of its bits?



Unexpected occurrence

of a particular density ψ

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

distinct of the Gauss density

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

The general problem of lattice reduction

A lattice of $\mathbb{R}^n$ = a discrete additive subgroup of $\mathbb{R}^n$.

A lattice $\mathcal{L}$ possesses a basis $B := (b_1, b_2, \ldots, b_p)$ with $p \leq n$,

$$\mathcal{L} := \{x \in \mathbb{R}^n; \quad x = \sum_{i=1}^{b} x_i b_i, \qquad x_i \in \mathbb{Z}\}$$

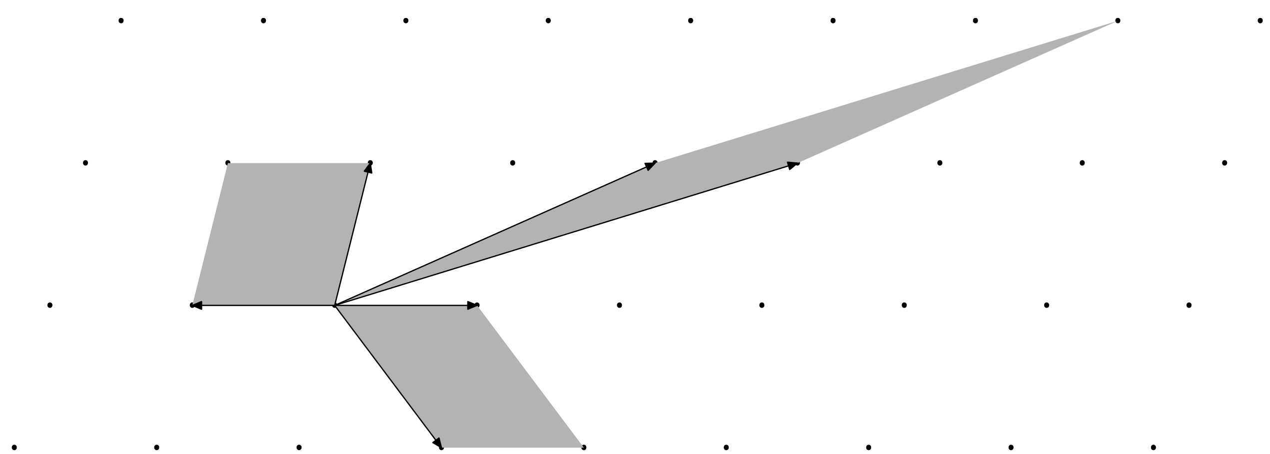
... and in fact, an infinite number of bases....

If now $\mathbb{R}^n$ is endowed with its (canonical) Euclidean structure, there exist bases (called reduced) with good Euclidean properties: their vectors are short enough and almost orthogonal.

Lattice reduction Problem : From a lattice $\mathcal{L}$ given by a basis B, construct from B a reduced basis $\hat{B}$ of $\mathcal{L}$.

Many applications of this problem in various domains:
number theory, arithmetics, discrete geometry..... and cryptology.

Lattice reduction algorithms in the two dimensional case.



Lattice Reduction in two dimensions.

Up to an isometry, the lattice $\mathcal{L}$ is a subset of $\mathbb{R}^2$ or..... $\mathbb{C}$.

To a pair $(u, v) \in \mathbb{C}^2$, with $u \neq 0$, we associate a unique $z \in \mathbb{C}$:

$$z := \frac{v}{u} = \frac{(u \cdot v)}{|u|^2} + i\frac{\det(u, v)}{|u|^2}.$$

Up to a similarity, the lattice $\mathcal{L}(u, v)$ becomes $\mathcal{L}(1, z) =: L(z)$.

Bad bases (u, v) correspond to complex z with small $|\Im z|$.

Three main facts in two dimensions.

– The existence of an optimal basis = a minimal basis

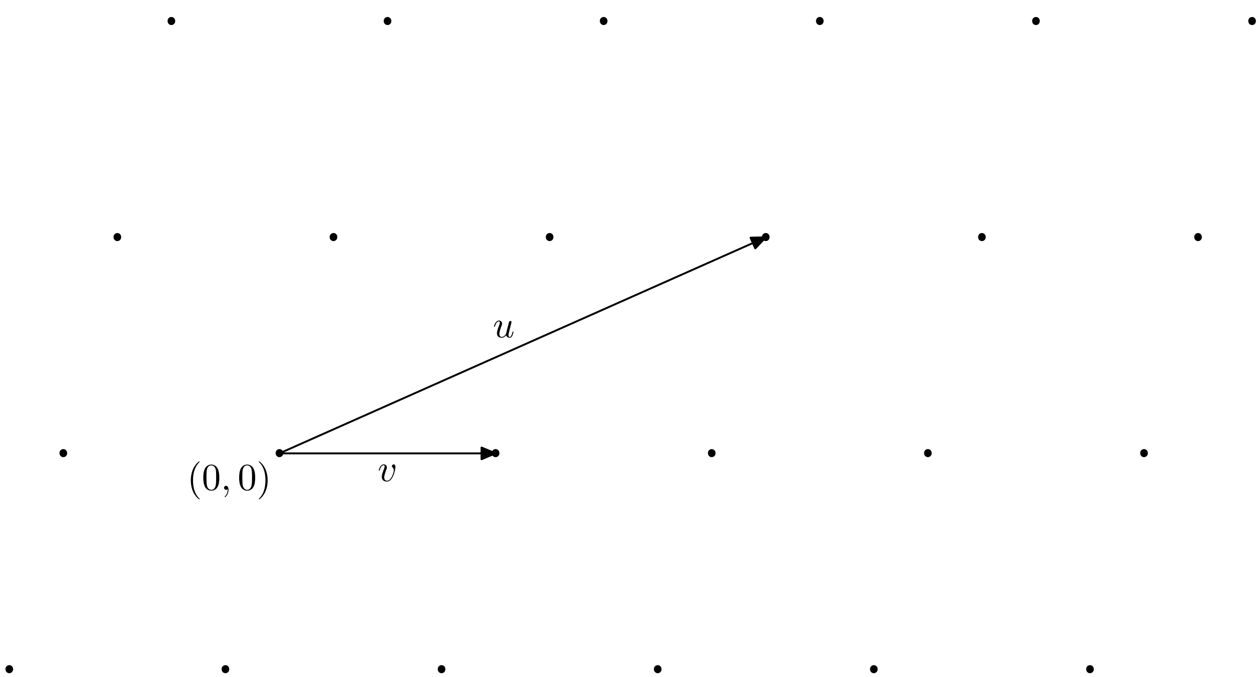
– A characterization of an optimal basis.

– An efficient algorithm which finds it = The Gauss Algorithm.

The Gauss algorithm is an extension of the Euclid algorithm.

It performs integer translations – seen as "vectorial" divisions–

$$u = qv + r \quad \text{with} \quad q = \left\lfloor \Re\left(\frac{u}{v}\right) \right\rceil = \left\lfloor \frac{u \cdot v}{|v|^2} \right\rceil, \quad \left| \Re\left(\frac{r}{v}\right) \right| \leq \frac{1}{2}$$



Here $q = 2$

The Gauss algorithm is an extension of the Euclid algorithm.
It performs integer translations – seen as "vectorial" divisions–, and exchanges.

Euclid's algorithm	Gauss' algorithm
Division between real numbers $u = qv + r$ with $q = \left\lfloor \frac{u}{v} \right\rceil$ and $\left\lvert \frac{r}{v} \right\rvert \leq \frac{1}{2}$	Division between complex vectors $u = qv + r$ with $q = \left\lfloor \Re\left(\frac{u}{v}\right) \right\rceil$ and $\left\lvert \Re\left(\frac{r}{v}\right) \right\rvert \leq \frac{1}{2}$
Division + exchange $(v, u) \to (r, v)$ "read" on $x = v/u$ $T(x) = \frac{1}{x} - \left\lfloor \frac{1}{x} \right\rceil$	Division + exchange $(v, u) \to (r, v)$ "read" on $z = v/u$ $T(z) = \frac{1}{z} - \left\lfloor \Re\left(\frac{1}{z}\right) \right\rceil$
Stopping condition: $x = 0$	Stopping condition: $z \in \mathcal{F}$

Analysis of the Gauss Algorithm: Instance of a Dynamical Analysis.

The analysis of the Euclid Algorithm uses the transfer operator

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

where $\mathcal{H}$ is the set of the inverse branches of (I, T)

The analysis of the Gauss Algorithm uses the transfer operator

$$\underline{\mathbf{H}}_s[F](x, y) := \sum_{h \in \mathcal{H}} |h'(x)|^{s/2} |h'(y)|^{s/2} \cdot F(h(x), h(y))$$

which acts on functions of two variables and extends $\mathbf{H}_s$, since

$$\underline{\mathbf{H}}_s[F](x, x) = \mathbf{H}_s[f](x), \qquad \text{with} \quad f(x) := F(x, x)$$

The dynamics of the EUCLID Algorithm is described with $s = 1$.
The (uniform) dynamics of the GAUSS Algorithm is described with $s = 2$.
The (general) dynamics of the GAUSS algorithm is described with $s = 2 + r$.
When $r \to -1$, the GAUSS Algorithm tends to the EUCLID Algorithm.

THE END....