**Programming Research Group**

# FURTHER ANALYSIS OF THE BINARY EUCLIDEAN ALGORITHM

Richard P. Brent

November 1999

PRG-TR-7-99

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD

# Further analysis of the Binary Euclidean algorithm

Richard P. Brent[1]
Oxford University

rpb@comlab.ox.ac.uk

4 November 1999

**Abstract**

The binary Euclidean algorithm is a variant of the classical Euclidean algorithm. It avoids multiplications and divisions, except by powers of two, so is potentially faster than the classical algorithm on a binary machine.

We describe the binary algorithm and consider its average case behaviour. In particular, we correct some errors in the literature, discuss some recent results of Vallée, and describe a numerical computation which supports a conjecture of Vallée.

## 1 Introduction

In §2 we define the *binary* Euclidean algorithm and mention some of its properties, history and generalisations. Then, in §3 we outline the heuristic model which was first presented in 1976 [4]. Some of the results of that paper are mentioned (and simplified) in §4.

Average case analysis of the binary Euclidean algorithm lay dormant from 1976 until Brigitte Vallée's recent analysis [29, 30]. In §§5–6 we discuss Vallée's results and conjectures. In §8 we give some numerical evidence for one of her conjectures. Some connections between Vallée's results and our earlier results are given in §7.

Finally, in §9 we take the opportunity to point out an error in the 1976 paper [4]. Although the error is theoretically significant and (when pointed out) rather obvious, it appears that no one noticed it for about twenty years. The manner of its discovery is discussed in §9. Some open problems are mentioned in §10.

### 1.1 Notation

$\lg(x)$ denotes $\log_2(x)$. $N, n, a, k, u, v$ are positive integers.

$\mathrm{Val}_2(u)$ denotes the dyadic valuation of the positive integer $u$, i.e. the greatest integer $j$ such that $2^j \mid u$. This is just the number of trailing zero bits in the binary representation of $u$.

$f, g, F, \widetilde{F}, G$ are functions of a real or complex variable, and usually $f(x) = F'(x)$, $g(x) = G'(x)$ etc. Often $f, g$ are probability densities and $F, G$ are the corresponding probability distributions.

**Warning**: Brent [4], Knuth [20], and Vallée [27, 29, 30] use incompatible notation. Knuth uses $G(x)$ for our $\widetilde{F}(x)$, and $S(x)$ for our $G(x)$. Vallée sometimes interchanges our $f$ and $g$.

 rpb183tr

## 2 The Binary Euclidean Algorithm

The idea of the *binary* Euclidean algorithm is to avoid the "division" operation $r \leftarrow m \bmod n$ of the classical algorithm, but retain $O(\log N)$ worst (and average) case.

We assume that the algorithm is implemented on a binary computer so division by a power of two is easy. In particular, we assume that the "shift right until odd" operation

$$u \leftarrow u/2^{\mathrm{Val}_2(u)}$$

or equivalently

$$\text{while even}(u) \text{ do } u \leftarrow u/2$$

can be performed in constant time, although time $O(\mathrm{Val}_2(u))$ would be sufficient.

### 2.1 Definitions of the Binary Euclidean Algorithm

There are several almost equivalent ways to define the algorithm. It is easy to take account of the largest power of two dividing the inputs, using the relation

$$\mathrm{GCD}(u,v) = 2^{\min(\mathrm{Val}_2(u),\mathrm{Val}_2(v))}\ \mathrm{GCD}\left(u/2^{\mathrm{Val}_2(u)}, v/2^{\mathrm{Val}_2(v)}\right) ,$$

so for simplicity we assume that $u$ and $v$ are *odd* positive integers. Following is a simplified version of the algorithm given in Knuth [20, §4.5.2].

**Algorithm B**

**B1.** $t \leftarrow |u - v|$;
if $t = 0$ terminate with result $u$

**B2.** $t \leftarrow t/2^{\mathrm{Val}_2(t)}$

**B3.** if $u \geq v$ then $u \leftarrow t$ else $v \leftarrow t$;
go to B1.

### 2.2 History

The binary Euclidean algorithm is usually attributed to Silver and Terzian [25] or (independently) Stein [26] in the early 1960s. However, it seems to go back much further. Knuth [20, §4.5.2] quotes a translation of a first-century AD Chinese text *Chiu Chang Suan Shu* on how to reduce a fraction to lowest terms:

> If halving is possible, take half.
>
> Otherwise write down the denominator and the numerator,
> and subtract the smaller from the greater.
>
> Repeat until both numbers are equal.
>
> Simplify with this common value.

This is essentially Algorithm B. Hence, the binary algorithm is almost as old as the classical Euclidean algorithm [11].

## 2.3 The Worst Case

Although this paper is mainly concerned with the average case behaviour of the binary Euclidean algorithm, we mention the worst case briefly. At step B1, $u$ and $v$ are odd, so $t$ is even. Thus, step B2 always reduces $t$ by at least a factor of two. Using this fact, it is easy to show that $\lg(u+v)$ decreases by at least one each time step B3 is executed, so this occurs at most

$$\lfloor \lg(u+v) \rfloor$$

times [20, exercise 4.5.2.37]. Thus, if $N = \max(u, v)$, step B3 is executed at most

$$\lg(N) + O(1)$$

times.

Even if step B2 is replaced by single-bit shifts

$$\text{while even}(t) \text{ do } t \leftarrow t/2$$

the overall worst case time is still $O(\log N)$. In fact, it is easy to see that $(\lg(u)+\lg(v))$ decreases by at least one for each right shift, so the number of right shifts is at most $2\lg(N)$.

## 2.4 The Extended Binary Algorithm

It is possible to give an *extended* binary GCD algorithm which computes integer multipliers $\alpha$ and $\beta$ such that

$$\alpha u + \beta v = \mathrm{GCD}(u, v) \ .$$

Let $n = \lceil \lg u \rceil + \lceil \lg v \rceil$ be the number of bits in the input. Purdy [22] gave an algorithm with average running time $O(n)$ but worst case of order $n^2$. This was improved by Bojanczyk and Brent [2], whose algorithm has worst case running time $O(n)$.

Let $g = \mathrm{GCD}(u, v)$, $u' = u/g$, $v' = v/g$. In [2, §4] an algorithm is given for reducing the fraction $u/v$ to $u'/v'$ without performing any divisions (except by powers of two).

## 2.5 Parallel Variants and the Class NC

There is a systolic array variant of the binary GCD algorithm (Brent and Kung [7]). This takes time $O(\log N)$ using $O(\log N)$ 1-bit processors. The overall bit-complexity is $O((\log N)^2)$.

For $n$-bit numbers the systolic algorithm gives time $O(n)$ using $O(n)$ processors. This is close to the best known parallel time bound (Borodin *et al* [3]).

It is not known if GCD is in the class **NC**.[2] This is an interesting open problem because the basic arithmetic operations of addition, multiplication, and division are in **NC** (see Cook [8, 9]). Thus, the operation of computing GCDs is perhaps the simplest arithmetic operation which is *not* known to be in **NC**.

It is conceivable that testing coprimality, i.e. answering the question of whether $\mathrm{GCD}(u, v) = 1$, is "easier" than computing $\mathrm{GCD}(u, v)$ in general. There is evidence that testing coprimality *is* in **NC** (Litow [21]).

[2] **NC** is the class of problems which can be solved in parallel in time bounded by a polynomial in $\log L$, where $L$ is the length of the input, using a number of processors bounded by a polynomial in $L$. Note that for the GCD problem $L = \Theta(n) = \Theta(\log N)$, so we are asking for a time polynomial in $\log\log N$, not in $\log N$.

# 3 A Heuristic Continuous Model

To analyse the expected behaviour of Algorithm B, we can follow what Gauss [15] did for the classical algorithm. This was first attempted in [4]. There is a summary in Knuth [20, §4.5.2].

Assume that the initial inputs $u_0$, $v_0$ to Algorithm B are uniformly and independently distributed in $(0, N)$, apart from the restriction that they are odd. Let $(u_n, v_n)$ be the value of $(u, v)$ after $n$ iterations of step B3.

Let

$$x_n = \frac{\min(u_n, v_n)}{\max(u_n, v_n)}$$

and let $F_n(x)$ be the probability distribution function of $x_n$ (in the limit as $N \to \infty$). Thus $F_0(x) = x$ for $x \in [0, 1]$.

## 3.1 A Plausible Assumption

We make the assumption[3] that $\mathrm{Val}_2(t)$ takes the value $k$ with probability $2^{-k}$ at step B2. The assumption is plausible because $\mathrm{Val}_2(t)$ at step B2 depends on the least significant bits of $u$ and $v$, whereas the comparison at step B3 depends on the most significant bits, so one would expect the steps to be (almost) independent when $N$ is large. In fact, this independence is exploited in the systolic algorithms [2, 6, 7] where processing elements perform operations on the least significant bits without waiting for information about the most significant bits.

## 3.2 The Recurrence for $F_n$

Consider the effect of steps B2 and B3. We can assume that initially $u > v$, so $t = u - v$. If $\mathrm{Val}_2(t) = k$ then $X = v/u$ is transformed to

$$X' = \min\left(\frac{u - v}{2^k v}, \frac{2^k v}{u - v}\right) = \min\left(\frac{1 - X}{2^k X}, \frac{2^k X}{1 - X}\right) .$$

It follows that $X' < x$ iff

$$X < \frac{1}{1 + 2^k/x} \text{ or } X > \frac{1}{1 + 2^k x} .$$

Thus, the recurrence for $F_n(x)$ is

$$F_{n+1}(x) = 1 + \sum_{k \geq 1} 2^{-k} \left(F_n\left(\frac{1}{1 + 2^k/x}\right) - F_n\left(\frac{1}{1 + 2^k x}\right)\right) \tag{1}$$

with initial condition $F_0(x) = x$ for $x \in [0, 1]$.

It is convenient to define

$$\widetilde{F}_n(x) = 1 - F_n(x) .$$

The recurrence for $\widetilde{F}_n(x)$ is

$$\widetilde{F}_{n+1}(x) = \sum_{k \geq 1} 2^{-k} \left(\widetilde{F}_n\left(\frac{1}{1 + 2^k/x}\right) - \widetilde{F}_n\left(\frac{1}{1 + 2^k x}\right)\right) \tag{2}$$

and $\widetilde{F}_0(x) = 1 - x$ for $x \in [0, 1]$.

[3] Vallée does not make this assumption. Her results are mentioned in §§5–6. They show that the assumption is correct in the limit as $N \to \infty$.

### 3.3 The Recurrence for $f_n$

Differentiating the recurrence (1) for $F_n$ we obtain (formally) a recurrence for the probability density $f_n(x) = F_n'(x)$:

$$f_{n+1}(x) = \sum_{k \ge 1} \left( \left( \frac{1}{x+2^k} \right)^2 f_n\left( \frac{x}{x+2^k} \right) + \left( \frac{1}{1+2^k x} \right)^2 f_n\left( \frac{1}{1+2^k x} \right) \right) . \tag{3}$$

It was noted in [4, §5] that the coefficients in this recurrence are positive, and that the recurrence preserves the $L_1$ norm of nonnegative functions (this is to be expected, since the recurrence maps one probability density to another).

### 3.4 Operator Notation

The recurrence for $f_n$ may be written as

$$f_{n+1} = \mathcal{B}_2 f_n,$$

where the operator $\mathcal{B}_2$ is the case $s = 2$ of a more general operator $\mathcal{B}_s$ which is defined in (14) of §5.4.

## 4 Conjectured and Empirical Results

In the 1976 paper [4] we gave numerical and analytic evidence (but no proof) that $F_n(x)$ converges to a limiting distribution $F(x)$ as $n \to \infty$, and that $f_n(x)$ converges to the corresponding probability density $f(x) = F'(x)$ (note that $f = \mathcal{B}_2 f$ so $f$ is a "fixed point" of the operator $\mathcal{B}_2$).

Assuming the existence of $F$, it is shown in [4] that the expected number of iterations of Algorithm B is $\sim K \lg N$ as $N \to \infty$, where $K = 0.705\ldots$ is a constant given by

$$K = \ln 2 / E_\infty \ , \tag{4}$$

and[4]

$$E_\infty = \ln 2 + \int_0^1 \left( \sum_{k=2}^{\infty} \left( \frac{1-2^{-k}}{1+(2^k-1)x} \right) - \frac{1}{2(1+x)} \right) F(x)\, dx \ . \tag{5}$$

### 4.1 A Simplification

We can simplify the expressions (4)–(5) for $K$ to obtain

$$K = 2/b \ , \tag{6}$$

where

$$b = 2 - \int_0^1 \lg(1-x) f(x)\, dx \ . \tag{7}$$

Using integration by parts we obtain an equivalent expression

$$b = 2 + \frac{1}{\ln 2} \int_0^1 \frac{1-F(x)}{1-x}\, dx \ . \tag{8}$$

For my direct proof of (7)–(8), see Knuth [20, §4.5.2]. The idea is to consider the expected change in $\lg(uv)$ with each iteration of Algorithm B (to obtain the equivalent but more complicated expression (5) we considered the expected change in $\lg(u+v)$).

[4]We have corrected a typo in [4, eqn. (6.3)].

## 5 Another Formulation – Algorithm V

It will be useful to rewrite Algorithm B in the following equivalent form (using pseudo-Pascal):

**Algorithm V** { Assume $u \le v$ }

```
while u ≠ v do
  begin
  while u < v do
    begin
    j ← Val₂(v − u);
    v ← (v − u)/2^j;
    end;
  u ↔ v;
  end;
return u.
```

### 5.1 Continued Fractions

Vallée [30] shows a connection between Algorithm V and continued fractions of a certain form

$$\frac{u}{v} = \cfrac{1}{a_1 + \cfrac{2^{k_1}}{a_2 + \cfrac{2^{k_2}}{\ddots + \cfrac{2^{k_{r-1}}}{a_r + 2^{k_r}}}}}$$

which by convention we write as

$$\frac{u}{v} = 1/a_1 + 2^{k_1}/a_2 + 2^{k_2}/\ldots/(a_r + 2^{k_r}) \,. \tag{9}$$

Here $a_j$ is odd, $k_j > 0$, and $0 < a_j < 2^{k_j}$ (excluding the trivial case $u = v = 1$).

### 5.2 Some Details of Vallée's Results

Algorithm V has two nested loops. The outer loop exchanges $u$ and $v$. Between two exchanges, the inner loop performs a sequence of subtractions and shifts which can be written as

$$\begin{array}{rcl} v & \to & u + 2^{b_1} v_1; \\ v_1 & \to & u + 2^{b_2} v_2; \\ & \cdots & \\ v_{m-1} & \to & u + 2^{b_m} v_m \end{array}$$

with $v_m \le u$.

If $x_0 = u/v$ at the beginning of an inner loop, the effect of the inner loop followed by an exchange is the rational $x_1 = v_m/u$ defined by

$$x_0 = \frac{1}{a + 2^k x_1} \,,$$

where $a$ is an odd integer given by

$$a = 1 + 2^{b_1} + 2^{b_1+b_2} + \cdots + 2^{b_1+\cdots+b_{m-1}} \,,$$

and the exponent $k$ is given by

$$k = b_1 + \cdots + b_m \, .$$

Thus, the rational $u/v$, for $1 \le u < v$, has a unique *binary continued fraction expansion* of the form (9). Vallée studies three parameters related to this continued fraction:

1. The height or the depth (i.e. the number of exchanges) $r$.

2. The total number of operations necessary to obtain the expansion (equivalently, the number of times step B2 of Algorithm B is performed): if $p(a)$ denotes the number of "1"s in the binary expansion of the integer $a$, it is equal to $p(a_1) + p(a_2) + \cdots + p(a_r)$.

3. The total number of one-bit shifts, i.e. the sum of exponents of 2 in the numerators of the binary continued fraction, $k_1 + \cdots + k_r$.

### 5.3 Vallée's Theorems

Vallée's main results give the average values of the three parameters above: the average values are asymptotically $A_i \ln N$ for certain computable constants $A_1, A_2, A_3$ related to the spectral properties of an operator $\mathcal{V}_2$ which is defined in (11) of §5.4. Clearly the constant $K$ of §4 is $A_2 \ln 2$.

### 5.4 Some Useful Operators

Operators $\mathcal{B}_s$, $\mathcal{V}_s$, $\mathcal{U}_s$, $\widetilde{\mathcal{U}}_s$, useful in the analysis of the binary Euclidean algorithm, are defined by

$$\mathcal{B}_s[f](x) = \sum_{k\ge1} \left( \left(\frac{1}{x+2^k}\right)^2 f\left(\frac{x}{x+2^k}\right) + \left(\frac{1}{1+2^k x}\right)^2 f\left(\frac{1}{1+2^k x}\right) \right) , \tag{10}$$

$$\mathcal{V}_s[f](x) = \sum_{k\ge1} \sum_{\substack{a \text{ odd}, \\ 0<a<2^k}} \left(\frac{1}{a+2^k x}\right)^s f\left(\frac{1}{a+2^k x}\right) , \tag{11}$$

$$\mathcal{U}_s[f](x) = \sum_{k\ge1} \left(\frac{1}{1+2^k x}\right)^s f\left(\frac{1}{1+2^k x}\right) , \tag{12}$$

$$\widetilde{\mathcal{U}}_s[f](x) = \left(\frac{1}{x}\right)^s \mathcal{U}_s[f]\left(\frac{1}{x}\right) . \tag{13}$$

In these definitions $s$ is a complex variable, and the operators are called Ruelle operators [24]. They are linear operators acting on certain function spaces. It is immediate from the definitions that

$$\mathcal{B}_s = \mathcal{U}_s + \widetilde{\mathcal{U}}_s, \tag{14}$$

The case $s = 2$ is of particular interest. $\mathcal{B}_2$ encodes the effect of one iteration of the inner "while" loop of Algorithm V, and $\mathcal{V}_2$ encodes the effect of one iteration of the outer "while" loop. See Vallée [29, 30] for further details.

### 5.5 History and Notation

$\mathcal{B}_2$ (denoted $T$) was introduced in [4], and was generalised to $\mathcal{B}_s$ by Vallée. $\mathcal{V}_s$ was introduced by Vallée [29, 30]. We shall call

- $\mathcal{B}_s$ (or sometimes just $\mathcal{B}_2$) the *binary Euclidean operator* and
- $\mathcal{V}_s$ (or sometimes just $\mathcal{V}_2$) *Vallée's operator.*

### 5.6 Relation Between the Operators

The binary Euclidean operator and Vallée's operator are closely related, as Lemma 1 and Theorem 1 show.

**Lemma 1**

$$\mathcal{V}_s = \mathcal{V}_s\widetilde{\mathcal{U}}_s + \mathcal{U}_s.$$

**Proof.** From (11),

$$\mathcal{V}_s[\widetilde{\mathcal{U}}_s[f]](x) = \sum_{k\geq 1} \sum_{\substack{a \text{ odd}, \\ 0<a<2^k}} \left(\frac{1}{a+2^k x}\right)^s \widetilde{\mathcal{U}}_s[f]\left(\frac{1}{a+2^k x}\right)$$

but, from (12) and (13),

$$\widetilde{\mathcal{U}}_s[f](y) = \sum_{m\geq 1} \left(\frac{1}{2^m+y}\right)^s f\left(\frac{1}{1+2^m/y}\right).$$

On substituting $y = 1/(a+2^k x)$ we obtain

$$\mathcal{V}_s[\widetilde{\mathcal{U}}_s[f]](x) = \sum_{k\geq 1} \sum_{\substack{a \text{ odd}, \\ 0<a<2^k}} \sum_{m\geq 1} \left(\frac{1}{1+2^m a+2^{k+m}x}\right)^s f\left(\frac{1}{1+2^m a+2^{k+m}x}\right).$$

Thus, to show that

$$\mathcal{V}_s[f](x) = \mathcal{U}_s[f](x) + \mathcal{V}_s\widetilde{\mathcal{U}}_s[f](x)$$

it suffices to observe that the set of polynomials

$$\{a' + 2^{k'}x \mid k' \geq 1,\ a' \text{ odd},\ 0 < a' < 2^{k'}\}$$

is the disjoint union of the two sets

$$\{1 + 2^k x \mid k \geq 1\}$$

and

$$\{1 + 2^m a + 2^{k+m}x \mid k \geq 1,\ m \geq 1,\ a \text{ odd},\ 0 < a < 2^k\}.$$

To see this, consider the two cases $a' = 1$ and $a' > 1$. If $2^{k'} > a' > 1$ we can write $a' = 1 + 2^m a$, $k' = k + m$, for some (unique) odd $a$ and positive $k$, $m$. □

### 5.7 Algorithmic Interpretation

Algorithm V gives an interpretation of Lemma 1 in the case $s = 2$. If the input density of $x = u/v$ is $f(x)$ then execution of the inner "while" loop followed by the exchange of $u$ and $v$ transforms this density to $\mathcal{V}_2[f](x)$. However, by considering the first iteration of this loop (followed by the exchange if the loop terminates) we see that the transformed density is given by

$$\mathcal{V}_2\widetilde{\mathcal{U}}_2[f](x) + \mathcal{U}_2[f](x),$$

where the first term arises if there is no exchange, and the second arises if an exchange occurs.

### 5.8 Consequence of Lemma 1

The following Theorem gives a simple relationship between $\mathcal{B}_s$, $\mathcal{V}_s$ and $\mathcal{U}_s$.

**Theorem 1**

$$(\mathcal{V}_s - \mathcal{I})\mathcal{U}_s = \mathcal{V}_s(\mathcal{B}_s - \mathcal{I}) \ .$$

**Proof.** This is immediate from Lemma 1 and the definitions of the operators. □

### 5.9 Fixed Points

It follows immediately from Theorem 1 that, if

$$g = \mathcal{U}_2 f, \tag{15}$$

then

$$(\mathcal{V}_2 - \mathcal{I})g = \mathcal{V}_2(\mathcal{B}_2 - \mathcal{I})f.$$

Thus, if $f$ is a fixed point of the operator $\mathcal{B}_2$, then $g = \mathcal{U}_2 f$ is a fixed point of the operator $\mathcal{V}_2$. (We can not assert the converse without knowing something about the null space of $\mathcal{V}_2$.) From a result of Vallée [30, Prop. 4] we know that $\mathcal{V}_2$, acting on a certain Hardy space $\mathcal{H}^2(\mathcal{D})$, has a unique positive dominant simple eigenvalue 1, so $g$ must be (a constant multiple of) the corresponding eigenfunction (provided $g \in \mathcal{H}^2(\mathcal{D})$).

**Lemma 2** *If $f$ is a fixed point of $\mathcal{B}_2$ and $g$ is given by* (15), *then*

$$f(1) = 2g(1) = 2\sum_{k\geq 1}\left(\frac{1}{1+2^k}\right)^2 f\left(\frac{1}{1+2^k}\right) \ .$$

**Proof.** This is immediate from the definitions of $\mathcal{B}_2$ and $\mathcal{U}_2$. □

## 6 A Result of Vallée

Using her operator $\mathcal{V}_s$, Vallée [30] proved that

$$K = \frac{2\ln 2}{\pi^2 g(1)} \sum_{\substack{a \text{ odd},\\ a>0}} 2^{-\lfloor \lg a \rfloor} G\left(\frac{1}{a}\right) \tag{16}$$

where $g$ is a nonzero fixed point of $\mathcal{V}_2$ (i.e. $g = \mathcal{V}_2 g \neq 0$) and $G(x) = \int_0^x g(t)\,dt$ . This is the only expression for $K$ which has been proved rigorously.

Because $\mathcal{V}_s$ has nice spectral properties, the existence and uniqueness (up to scaling) of $g$ can be established.

### 6.1 A Conjecture of Vallée

Let

$$\lambda = f(1) \ , \tag{17}$$

where $f$ is the limiting probability density (conjectured to exist) as in §4. $\lambda$ and $K$ are fundamental constants which are not known to have simple closed form expressions – to evaluate them numerically we seem to have to approximate a probability density $f(x)$ (or $g(x)$) or the corresponding distribution $F(x)$ (or $G(x)$). Vallée (see Knuth [20, §4.5.2(61)]) conjectured that

$$\frac{\lambda}{b} = \frac{2\ln 2}{\pi^2} \ ,$$

where $b$ is given by (7) or (8). Equivalently, from (6), her conjecture is that

$$K\lambda = \frac{4\ln 2}{\pi^2} \ . \tag{18}$$

Vallée proved the conjecture under the assumption that the operator $\mathcal{B}_s$ satisfies a "spectral gap" condition which has not been proved, but which is plausible because it is known to be satisfied by $\mathcal{V}_s$. Specifically, a sufficient condition is that the operator, acting on a suitable function space, has a simple positive dominant eigenvalue $\lambda_1$, and there is a positive $\epsilon$ such that all other eigenvalues $\lambda_j$ satisfy $|\lambda_j| \le \lambda_1 - \epsilon$.

# 7 Some Relations Between Fixed Points

In this section we *assume* that $f$ is a fixed point of the operator $\mathcal{B}_2$, $g = \mathcal{U}_2 f$ as in §5.9 is a fixed point of the operator $\mathcal{V}_2$, and both $f$ and $g$ are analytic functions (not necessarily regular at $x = 0$). Using analyticity we extend the domains of $f$, $g$ etc to include the positive real axis $(0, +\infty)$. Let

$$F(x) = \int_0^x f(t)\ dt$$

and

$$G(x) = \int_0^x g(t)\ dt$$

be the corresponding integrals. By scaling, we can assume that

$$F(1) = 1$$

but, in view of (15), we are not free to scale $g$. (See (20) below.)

From the definition (12) of $\mathcal{U}_s$ and (15), we have

$$g(x) = \sum_{k=1}^{\infty} \left(\frac{1}{1+2^k x}\right)^2 f\left(\frac{1}{1+2^k x}\right) \ ,$$

so, integrating with respect to $x$,

$$G(x) = \sum_{k=1}^{\infty} 2^{-k} \left(F(1) - F\left(\frac{1}{1+2^k x}\right)\right) \ .$$

This simplifies to

$$G(x) = \sum_{k=1}^{\infty} 2^{-k} \widetilde{F}\left(\frac{1}{1+2^k x}\right) \ . \tag{19}$$

Although our derivation of (19) assumes $x \in [0,1]$, we can use (19) to give an analytic continuation of $G(x)$. Allowing $x$ to approach $+\infty$, we see that there exists

$$\lim_{x\to+\infty} G(x) = G(+\infty)$$

say, and

$$G(+\infty) = 1 \ . \tag{20}$$

We can use the functional equation (2) to extend the domain of definition of $\widetilde{F}(x)$ to the nonnegative real axis $[0, +\infty)$. It is convenient to work with $\widetilde{F}(x) = 1 - F(x)$ rather than with $F(x)$ because of the following result.

**Lemma 3**

$$\widetilde{F}(x) = G(1/x) - G(x)$$

*and consequently*

$$\widetilde{F}(1/x) = -\widetilde{F}(x) \ .$$

**Proof.** This is immediate from (2) and (19). □

**Lemma 4**

$$G(x) = \sum_{k=1}^{\infty} 2^{-k} \sum_{\substack{a \text{ odd}, \\ 0<a<2^k}} \left( G\left(\frac{1}{a}\right) - G\left(\frac{1}{a+2^k x}\right)\right) \ .$$

**Proof.** Since $g(x)$ is a fixed point of $\mathcal{V}_2$, we have

$$\begin{aligned} g(x) &= \mathcal{V}_2[g](x) \\ &= \sum_{k=1}^{\infty} \sum_{\substack{a \text{ odd}, \\ 0<a<2^k}} \left(\frac{1}{a+2^k x}\right)^2 g\left(\frac{1}{a+2^k x}\right) \ . \end{aligned}$$

Integrating with respect to $x$ and making the change of variable $u = 1/(a + 2^k x)$ gives

$$G(x) = \sum_{k=1}^{\infty} 2^{-k} \sum_{\substack{a \text{ odd}, \\ 0<a<2^k}} \int_{1/(a+2^k x)}^{1/a} g(u) du \ ,$$

and the result follows. □

**Lemma 5**

$$\sum_{k=1}^{\infty} 2^{-k} \sum_{\substack{a \text{ odd}, \\ 0<a<2^k}} G\left(\frac{1}{a}\right) = 1 \ .$$

**Proof.** Let $x \to +\infty$ in Lemma 4 and use (20). □

The sum occurring in the following Lemma is the same as the sum in (16). To avoid confusion we repeat that our normalisation of $G$ is different from Vallée's, but note that the right side of (16) is independent of the normalisation of $G$ because $g(1)$ appears in the denominator.

**Lemma 6**

$$\sum_{\substack{a \text{ odd}, \\ a>0}} 2^{-\lfloor \lg a \rfloor} G\left(\frac{1}{a}\right) = 1 \ .$$

**Proof.** We can write Lemma 5 as

$$\sum_{\substack{a \text{ odd}, \\ a>0}} c_a \, G\left(\frac{1}{a}\right) = 1 \; ,$$

where

$$c_a = \sum_{\substack{k \ge 1, \\ 2^k > a}} 2^{-k} = 2^{-\lfloor \lg a \rfloor} \; .$$

Thus, the sums occurring in Lemma 5 and Lemma 6 are identical. □

**Theorem 2** *Under the assumptions stated at the beginning of this section, the expressions* (16) *and* (18) *are equivalent.*

**Proof.** This follows immediately from Lemmas 2 and 6. □

**Remarks.** As noted above, Vallée proved (18) under an assumption about the spectrum of $\mathcal{B}_s$. Our proof of Theorem 2 is more direct. We are not able to prove the equivalence of (6) and (18), but (as described in §8) it has been verified numerically to high precision.

# 8 Numerical Results

Using an improvement of the "discretization method" of [4], with Richardson extrapolation (see §8.1) and the equivalent of more than 50 decimal places (50D) working precision, we computed the limiting probability distribution $F$, then $K$ (using (6) and (8)), $\lambda = f(1)$, and $K\lambda$. The results were

$$\begin{aligned} K &= 0.7059712461\ 0191639152\ 9314135852\ 8817666677 \\ \lambda &= 0.3979226811\ 8831664407\ 6707161142\ 6549823098 \\ K\lambda &= 0.2809219710\ 9073150563\ 5754397987\ 9880385315 \end{aligned}$$

These are believed to be correctly rounded values.

The computed value of $K\lambda$ agrees with $4 \ln 2/\pi^2$ to 40 decimals[5], in complete agreement with Vallée's conjecture (18).

## 8.1 Some Details of the Numerical Computation

A consequence of Lemma 3 is that $\widetilde{F}(e^{-y})$ is an *odd* function of $y$. This fact was exploited in the numerical computations. By discretising with uniform stepsize $h$ in the variable $y$ we can obtain $K$ with error $O(h^{2r+2})$ after $r$ Richardson/Romberg [16, 18] extrapolations, because the error has an asymptotic expansion containing only even powers of $h$.

In fact, we found it better to take a uniform stepsize $h$ in the variable $z = \sqrt{y}$, i.e. make the change of variables

$$x = \exp(-z^2)$$

because this puts more points near $x = 0$ and less points in the "tail". We truncated at a point $z_{max}$ sufficiently large that $\exp(-z_{max}^2)$ was negligible.

To obtain 40D results it was sufficient to take $z_{max} = 11$, $h = z_{max}/2^{15}$, $r = 7$, iterate the recurrence for $\widetilde{F}_n$ 81 times (using interpolation by polynomials of degree $2r + 1$ where necessary) to obtain $\widetilde{F} \approx \widetilde{F}_{81}$ to $O(h^{16})$ accuracy. Using the trapezoidal rule, we obtained $K$

[5] In fact the agreement is to 44 decimals.

by numerical quadrature to $O(h^2)$ accuracy, and then applied seven Richardson extrapolations (using the results for stepsize $h, 2h, 2^2h, \cdots, 2^7h$) to obtain $K$ with error $O(h^{16})$. Similarly, we approximated $\lambda = F'(1)$ by

$$\lambda \approx \widetilde{F}(\exp(-h^2))/h^2$$

and then used extrapolation. Only three Richardson extrapolations were needed to obtain $\lambda$ with error $O(h^{16})$ because the relevant asymptotic expansion includes only powers of $h^4$.

### 8.2 Subdominant eigenvalues

In order to estimate the speed of convergence of $f_n$ to $f$ (assuming $f$ exists), we need more information on the spectrum of $\mathcal{B}_2$. What can be proved ?

Preliminary numerical results indicate that the sub-dominant eigenvalue(s) are a complex conjugate pair:

$$\lambda_2 = \overline{\lambda}_3 = 0.1735 \pm 0.0884i \ ,$$

with $|\lambda_2| = |\lambda_3| = 0.1948$ to 4D.

The appearance of a complex conjugate pair is interesting because in the classical case it is known that the eigenvalues are all real, and conjectured that (when ordered in decreasing absolute value) they alternate in sign [13].

### 8.3 Complexity of approximating $K$

We have several expressions for $K$ which are conjectured to be equivalent. Which is best for numerical computation of $K$ ? Suppose we want to estimate $K$ to $n$-bit accuracy, i.e. with error $O(2^{-n})$.

We could iterate the recurrence

$$g_{k+1}(x) = \mathcal{V}_2[g_k](x)$$

to obtain the principal fixed point $g(x)$ of Vallée's operator $\mathcal{V}_2$. However, the sum over odd $a$ in the definition of $\mathcal{V}_2$ appears to require the summation of exponentially many terms. Similarly, the sum in (16) appears to require exponentially many terms (unless we can assume that $g$ is scaled so that Lemma 6 applies).

Thus, it seems more efficient numerically to approximate the principal fixed point $f(x)$ of the binary Euclidean operator $\mathcal{B}_2$ or the corresponding integral $F(x)$ (or $\widetilde{F}(x) = 1 - F(x)$), even though the existence of $f$ or $F$ has not been proved.

It seems likely that $f(x)$ is unbounded in a neighbourhood of $x = 0$, so it is easier numerically to work with $\widetilde{F}(x)$. In the sum (2), the terms are bounded by $2^{-k}$, so we need take only $O(n)$ terms to get $n$-bit accuracy. (If we used the recurrence (3) it would not be so clear how many terms were required.)

Assuming that $\mathcal{B}_2$ has a positive dominant simple eigenvalue 1 (as seems very likely), convergence of $\widetilde{F}_k(x)$ to $\widetilde{F}(x)$ is linear, so $O(n)$ iterations are required. We have to tabulate $F(x)$ at a sufficiently dense set of points that the value at any point can be obtained to sufficient accuracy by interpolation. If the scheme of §8.1 is used, it may be sufficient to take $h = O(1/n)$ and use polynomial interpolation of degree $O(n/\log n)$. (Here [20, ex. 4.5.2.25] may be relevant.)

The final step, of estimating the integral (8) and $\lambda = f(1)$, can be done as in §8.1 with a relatively small amount of work. Alternatively, we can avoid the computation of an integral by using (18). However, the independent computation of $K$ and $\lambda$ provides a good check on the numerical results, since it is unlikely that any errors in the computed values of $K$ and/or $\lambda$ would be correlated in such a way as to leave the product $K\lambda$ unchanged.

Overall, the work required to obtain an $n$-bit approximation to $K$ appears to be bounded by a low-degree polynomial in $n$. Probably $O(n^4)$ bit operations are sufficient. It would be interesting to know if significantly faster algorithms exist. For example, is it possible to avoid the computation of $\widetilde{F}(x)$ or a similar function at a large number of points ?

# 9 Correcting an Error

In [4] it was claimed that, for all $n \geq 0$ and $x \in (0, 1]$,

$$F_n(x) = \alpha_n(x) \lg(x) + \beta_n(x) \ , \tag{21}$$

where $\alpha_n(x)$ and $\beta_n(x)$ are analytic and regular in the disk $|x| < 1$. However, *this is incorrect*, even in the case $n = 1$.

The error appeared to go unnoticed until 1997, when Knuth was revising Volume 2 in preparation for publication of the third edition. Knuth computed the constant $K$ using recurrences for the analytic functions $\alpha_n(x)$ and $\beta_n(x)$, and I computed $K$ directly using the defining integral and recurrences for $F_n(x)$. Our computations disagreed in the 14th decimal place ! Knuth found

$$K = 0.70597\ 12461\ 019\underline{45\ 99986} \cdots$$

but I found

$$K = 0.70597\ 12461\ 019\underline{16\ 39152} \cdots$$

We soon discovered the source of the error. It was found independently, and at the same time, by Flajolet and Vallée.

The source of the error is illustrated by [4, Lemma 3.1], which is incorrect, and corrected in [20, solution to ex. 4.5.2.29]. In order to explain the error, we need to consider Mellin transforms (a very useful tool in average-case analysis [12]).

## 9.1 Mellin Transforms and Mellin Inversion

The *Mellin transform* of a function[6] $g(x)$ is defined by

$$g^*(s) = \int_0^\infty g(x) x^{s-1} dx \ .$$

It is easy to see that, if

$$f(x) = \sum_{k \geq 1} 2^{-k} g(2^k x) \ ,$$

then the Mellin transform of $f$ is

$$f^*(s) = \sum_{k \geq 1} 2^{-k(s+1)} g^*(s) = \frac{g^*(s)}{2^{s+1} - 1} \ .$$

Under suitable conditions we can apply the Mellin inversion formula to obtain

$$f(x) = \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} f^*(s) x^{-s} ds \ .$$

Applying these results to

$$g(x) = 1/(1+x) \ ,$$

[6] The functions $f$ and $g$ here are not necessarily related to those occurring in other sections.

whose Mellin transform is

$$g^*(s) = \pi/\sin\pi s \ \text{ when } \ 0 < \mathcal{R}s < 1 \ ,$$

we find

$$f(x) = \sum_{k\ge 1} \frac{2^{-k}}{1+2^k x} \tag{22}$$

as a sum of residues of

$$\left(\frac{\pi}{\sin\pi s}\right) \frac{x^{-s}}{2^{s+1}-1} \tag{23}$$

for $\mathcal{R}s \le 0$. This gives

$$f(x) = 1 + x\lg x + \frac{x}{2} + xP(\lg x) - \frac{2}{1}x^2 + \frac{4}{3}x^3 - \cdots \ , \tag{24}$$

where

$$P(t) = \frac{2\pi}{\ln 2}\sum_{n=1}^{\infty} \frac{\sin 2n\pi t}{\sinh(2n\pi^2/\ln 2)} \ . \tag{25}$$

### 9.2 The "Wobbles" Caused by $P(t)$

$P(t)$ is a very small periodic function:

$$|P(t)| < 7.8 \times 10^{-12}$$

for real $t$. In [4, Lemma 3.1], the term $xP(\lg x)$ in (24) was omitted. Essentially, the poles of (23) off the real axis at

$$s = -1 \pm \frac{2\pi i n}{\ln 2} \ , \quad n = 1, 2, \ldots$$

were ignored.[7]

Because of the sinh term in the denominator of (25), the residues at the non-real poles are tiny, and numerical computations performed using single-precision floating-point arithmetic did not reveal the error.

### 9.3 Details of Corrections

The function $f(x)$ of (22) is called $D_1(x)$ in [4]. In (3.29) of [4, Lemma 3.1], the expression for $D_1(x)$ is missing the term $xP(\lg x)$.

Equation (3.8) of [4] is (correctly)

$$F_n(x) = 1 + D_n(1/x) - D_n(x)$$

so in Corollary 3.2 the expression for $F_1(x)$ is missing a term $-xP(\lg x)$.

The statement following Corollary 3.2 of [4], that "In principle we could obtain $F_2(x), F_3(x)$, etc in the same way as $F_1(x)$" is dubious because it is not clear how to handle the terms involving $P(\lg x)$.

To quote Gauss [notebook, 1800], who was referring to $F_2(x)$ etc for the *classical* algorithm:

> *Tam complicatæ evadunt, ut nulla spes superesse videatur.*[8]

Corollary 3.3 of [4], that $F_{n+1} \ne F_n$, is probably correct, but the proof given is incorrect because it assumes the incorrect form (21) for $F_n(x)$.

[7] In fact, the incorrect result was obtained without using Mellin transforms. If I had used them I probably would have obtained the correct result!

[8] They come out so complicated that no hope appears to be left.

### 9.4 An Analogy

Ramanujan made a similar error when he gave a formula for $\pi(x)$ (the number of primes $\leq x$) which essentially ignored the residues of $x^s\zeta'(s)/\zeta(s)$ arising from zeros of $\zeta(s)$ off the real axis. For further details we refer to Berndt [1], Hardy [17], Riesel [23, Ch. 1–3] and the references given there.

## 10 Conclusion and Open Problems

Since Vallée's recent work [29, 30], analysis of the average behaviour of the binary Euclidean algorithm has a rigorous foundation. However, some interesting open questions remain.

For example, does the binary Euclidean operator $\mathcal{B}_2$ have a unique positive dominant simple eigenvalue 1? Vallée [30, Prop. 4] has proved the corresponding result for her operator $\mathcal{V}_2$. Are the various expressions for $K$ given above all provably correct ? (Only (16) has been proved.) Is there an algorithm for the numerical computation of $K$ which is asymptotically faster than the one described in §8.1 ? How can we give rigorous error bounds on numerical approximations to $K$ ?

In order to estimate the speed of convergence of $f_n$ to $f$ (assuming $f$ exists), we need more information on the spectrum of $\mathcal{B}_2$. What can be proved ? As mentioned in §8.2, numerical results indicate that the sub-dominant eigenvalue(s) are a complex conjugate pair with absolute value about 0.1948.

It would be interesting to compute the spectra of $\mathcal{B}_2$ and $\mathcal{V}_2$ numerically, and compare with the classical case, where the spectrum is real and the eigenvalues appear to alternate in sign.

In order to give rigorous numerical bounds on the spectra of $\mathcal{B}_2$ and $\mathcal{V}_2$, we need to bound the error caused by making finite-dimensional approximations to these operators. This may be easier for $\mathcal{V}_2$ than for $\mathcal{B}_2$.


### Acknowledgements

Thanks to:

- Don Knuth for encouraging me to correct and extend my 1976 results for the third edition [20] of *Seminumerical Algorithms.* Some of the results given here are described in [20, §4.5.2].
- Brigitte Vallée for correspondence and discussions on her conjectures and results.
- Philippe Flajolet for his notes [12] on Mellin transforms.
- The British Council for support via an *Alliance* grant.

An abbreviated version of this report appeared as [5].

## References


[1] Bruce C. Berndt, *Ramanujan's Notebooks*, Parts I-III, Springer-Verlag, New York, 1985, 1989, 1991.

[2] Adam W. Bojanczyk and Richard P. Brent, A systolic algorithm for extended GCD computation, *Comput. Math. Applic.* **14** (1987), 233–238.

[3] A. Borodin, J. von zur Gathen and J. Hopcroft, Fast parallel matrix and GCD computations, *Proc. 23rd Annual IEEE Symposium on Foundations of Computer Science*, IEEE, New York, 1982, 65–77.

[4] Richard P. Brent, Analysis of the binary Euclidean algorithm, *New Directions and Recent Results in Algorithms and Complexity* (J. F. Traub, editor), Academic Press, New York, 1976, 321–355.

[5] Richard P. Brent, Twenty years' analysis of the binary Euclidean algorithm, *Proc. Oxford–Microsoft Symposium in Celebration of the work of C. A. R. Hoare*, Oxford, 13–15 September, 1999. `ftp://ftp.comlab.ox.ac.uk/pub/Documents/techpapers/Richard.Brent/rpb183.dvi.gz`. Final version to be published in the first volume of a series *Cornerstones in Computing* (series editor Richard Bird), MacMillan, 2000.

[6] R. P. Brent and H. T. Kung, Systolic VLSI arrays for linear-time GCD computation, in *VLSI 83* (F. Anceau and E. J. Aas, editors), North-Holland, Amsterdam, 1983, 145–154.

[7] R. P. Brent and H. T. Kung, A systolic VLSI array for integer GCD computation, in *ARITH-7, Proc. Seventh Symposium on Computer Arithmetic* (K. Hwang, editor), IEEE/CS Press, 1985.

[8] Stephen A. Cook, An overview of computational complexity, *Comm. ACM* **26** (1983), 401–408.

[9] Stephen A. Cook, A taxonomy of problems with fast parallel algorithms, *Information and Control* **64** (1985), 2–22.

[10] Hervé Daudé, Philippe Flajolet and Brigitte Vallée, An analysis of the Gaussian algorithm for lattice reduction, *Proc. ANTS'94, Lecture Notes in Computer Science* **877**, Springer-Verlag, 1994, 144–158. Extended version in *Combinatorics, Probability and Computing* **6** (1997), 397–433.

[11] Euclid, Book VII. For an English rendition, see Knuth [20, §4.5.2]. For some relevant pages of a Latin translation see `http://www.univ-tln.fr/~langevin/NOTES/EUCLIDE/Euclide.html`

[12] Philippe Flajolet and Robert Sedgewick, *The Average Case Analysis of Algorithms: Mellin Transform Asymptotics*, Report 2956, INRIA Rocquencourt, August 1996. `http://pauillac.inria.fr/algo/flajolet/Publications/anacombi4.ps.gz`

[13] Philippe Flajolet and Brigitte Vallée, On the Gauss-Kuzmin-Wirsing constant, manuscript, 29 October 1995. `http://pauillac.inria.fr/algo/flajolet/Publications/gauss-kuzmin.ps.gz`

[14] Philippe Flajolet and Brigitte Vallée, Continued fraction algorithms, functional operators and structure constants, *Theoretical Computer Science* **194** (1998), 1–34. See also `http://www-rocq.inria.fr/algo/flajolet/Publications/RR2931.ps.gz`

[15] Carl F. Gauss, Brief an Laplace vom 30 Jan. 1812, *Carl Friedrich Gauss Werke*, Bd. $\mathbf{X}_1$, Göttingen, 371–374.

[16] Gene H. Golub and James M. Ortega, *Scientific Computing and Differential Equations: An Introduction to Numerical Methods*, Academic Press, New York, 1992.

[17] G. H. Hardy, *Ramanujan: Twelve Lectures on Subjects Suggested by his Life and Work*, Cambridge University Press, Cambridge, 1940.

[18] Peter Henrici, *Elements of Numerical Analysis*, John Wiley and Sons, New York, 1964.

[19] Donald E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms* (first edition). Addison-Wesley, Menlo Park, 1969.

[20] Donald E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms* (third edition). Addison-Wesley, Menlo Park, 1997.

[21] Bruce Litow, Parallel complexity of integer coprimality, manuscript, 11 March 1999. `http://www.cs.jcu.edu.au/~bruce/papers/coprime99.ps`

[22] G. B. Purdy, A carry-free algorithm for finding the greatest common divisor of two integers, *Comput. Math. Applic.* **9** (1983), 311–316.

[23] Hans Riesel, *Prime Numbers and Computer Methods for Factorization*, second edition, Birkhäuser, Boston, 1994.

[24] David Ruelle, *Thermodynamic Formalism*, Addison Wesley, Menlo Park, 1978.

[25] Roland Silver and John Terzian, unpublished, 1962. See [19, §4.5.2].

[26] J. Stein, Computational problems associated with Racah algebra, *J. Comput. Phys.* **1** (1967), 397–405.

[27] Brigitte Vallée, Opérateurs de Ruelle–Mayer généralisés et analyse des algorithmes de Gauss et d'Euclide, *Acta Arithmetica* **81** (1997), 101–144.

[28] Brigitte Vallée, Dynamique des fractions continues à contraintes périodiques, *J. of Number Theory* **72** (1998), 183–235. `http://www.info.unicaen.fr/~brigitte/Publications/cfperi.ps`

[29] Brigitte Vallée, The complete analysis of the binary Euclidean algorithm, *Proc. ANTS'98*, *Lecture Notes in Computer Science* **1423**, Springer-Verlag, 1998, 77–94.

[30] Brigitte Vallée, Dynamics of the binary Euclidean algorithm: functional analysis and operators, *Algorithmica* **22** (1998), 660–685. `http://www.info.unicaen.fr/~brigitte/Publications/bin-gcd.ps`

[31] Brigitte Vallée and Charly Lemée, Average-case analyses of three algorithms for computing the Jacobi symbol, preprint, Oct. 1998. `http://www.info.unicaen.fr/~brigitte/Publications/jacobi.ps`