**МАТЕМАТИЧЕСКИЕ ВОПРОСЫ КРИПТОГРАФИИ**
**2018 Т. 9 № 2 С. 87–98**

УДК 519.719.2 DOI https://doi.org/10.4213/mvk257

# Approximate common divisor problem and lattice sieving

**K. D. Zhukov**

*TVP Laboratories, Moscow*

*Получено 02.II.2017*

**Abstract.** A heuristic algorithm for computing common divisors of two integers (one of which is known only approximately) is described. We reduce this computational problem to the solution of a system of integer linear inequalities. This system with two unknowns is solved by the method suggested by J. Franke and T. Kleinjung for lattice sieving. In some cases our algorithm is faster than other methods.

**Key words:** approximate common divisor problem, lattice sieving, system of integer linear inequalities, integer linear programming, Gaussian volume heuristics

## Задача о приближенном делителе и решеточное просеивание

**К. Д. Жуков**

*Лаборатории ТВП, Москва*

**Аннотация.** Описан эвристический алгоритм вычисления общего делителя двух целых чисел, одно из которых известно приближенно. Эта задача сводится к задаче решения системы целочисленных линейных неравенств. Такая система с двумя неизвестными может быть решена с помощью метода решеточного просеивания, предложенного Й. Франке и Т. Клейнюнгом. Разработанный алгоритм в некоторых случаях оказывается быстрее других известных методов.

**Ключевые слова:** задача о приближенном делителе, решеточное просеивание, система целочисленных линейных неравенств, целочисленное линейное программирование, объемная эвристика Гаусса

Citation: *Mathematical Aspects of Cryptography*, 2018, v. 9, № 2, pp. 87–98 (Russian)

# 1. Introduction

We begin with the description of the general approximate common divisor problem. Let $N_1, N_2, \ldots, N_k$ be integers; we denote by $A$ a nontrivial common divisor of $N_1 - \delta_1, N_2 - \delta_2, \ldots, N_k - \delta_k$ for some small $\delta_1, \delta_2, \ldots, \delta_k$. The goal is to find $A$ provided that it is large. The problem was originally introduced by N. Howgrave-Graham [1] with $k = 2$.

The special case of the general approximate common divisor problem with the first input to be exact (i.e. $\delta_1 = 0$) is called *partial approximate common divisor problem* (PACDP). S. Sarkar and S. Maitra [5] showed that the PACDP relates to implicit factoring problem. The last one was introduced by A. May and M. Ritzenhofen [3] and in special case consists in factoring RSA moduli $N_1 = p_1 q_1$ and $N_2 = p_2 q_2$ provided $|p_1 - p_2|$ is small.

In this paper we focus on two-dimensional PACDP. We show how to weaken the standard restrictions on inputs of PACDP and obtain the method of exponential complexity. We introduce also another exponential method for PACDP related to lattice sieving algorithm of J. Franke and T. Kleinjung. In some special cases our method is asymptotically the fastest.

# 2. Methods for solving PACDP

We describe a partial approximate common divisor problem (PACDP) as follows: *for every input* $N_1$, $N_2$, $\Delta$, $D \in \mathbb{N}$ $(\Delta < D < N_1, N_2)$ *find all* $B_1, B_2 \in \mathbb{N}$ *such that*

$$N_1 = AB_1 \quad \textit{and} \quad N_2 = AB_2 + \delta,$$

*where* $A \in \mathbb{N}$, $A > D$ *and* $\delta \in \mathbb{Z}$, $|\delta| < \Delta$. The PACDP is known to be hard in the case where parameters $\Delta$ and $D$ are close (there are too many solutions) or in the case where $D$ is much smaller than $\sqrt{N_1}$ (too large complexity).

Here we describe known methods for solving PACDP with some restrictions on inputs. The complexity of these methods depends polynomially on $n = \max\{[\ln N_1], [\ln N_2]\}$.

The first method was proposed by N. Howgrave-Graham [1]. The method is based on the observation that if

$$\delta < \frac{A^2}{2N_1},$$

then

$$\left| \frac{N_2}{N_1} - \frac{B_2}{B_1} \right| < \frac{1}{2B_1^2}.$$

Hence $B_2/B_1$ is one of the convergents of $N_2/N_1$. One may find all the convergents and check their numerators and denominators for being a solution of PACDP by means of $O(n^2 \ln n \ln\ln n)$ binary operations.

Another method due to N. Howgrave-Graham may be applied to PACDP if the inequality $\Delta < D^{\log_{N_1} D}$ holds. This method is based on the Coppersmith technique, its complexity is $O(n^2 \ln n \ln\ln n)$ binary operations.

There is also a heuristic approach to PACDP [4] related to the Shortest Vector Problem (SVP). Every solution $(B_1, B_2) \in \mathbb{N}^2$ of PACDP with inputs $N_1, N_2, \Delta, D \in \mathbb{N}$ satisfies the inequalities

$$-\frac{N_1\Delta}{D} < N_2B_1 - N_1B_2 < \frac{N_1\Delta}{D}, \quad 0 < B_1 < \frac{N_1}{D}. \tag{1}$$

Let us consider matrix $\mathbf{B} = \begin{pmatrix} N_2 & \Delta \\ -N_1 & 0 \end{pmatrix}$ and vector $\mathbf{b} = (B_1, B_2)\,\mathbf{B}$. The determinant of $\mathbf{B}$ equals $\det(\mathbf{B}) = N_1\Delta$. It follows from inequalities (1) that the Euclidean norm of $\mathbf{b}$ has the estimate

$$\|\mathbf{b}\| < \frac{\sqrt{2}N_1\Delta}{D}.$$

If $\Delta < D^2/N_1$, then

$$\frac{1}{D} < \frac{1}{\sqrt{N_1\Delta}}$$

and

$$\|\mathbf{b}\| < \frac{\sqrt{2}N_1\Delta}{\sqrt{N_1\Delta}} = \sqrt{2\,N_1\Delta} = \sqrt{2\,\bigl|\det(\mathbf{B})\bigr|},$$

i. e. $\mathbf{b}$ is a short vector of the lattice $\mathcal{L}(\mathbf{B})$. If a heuristic assumption that $\mathbf{b}$ is the single vector with the norm less than $\sqrt{2\,|\det(\mathbf{B})|}$ holds we may find it by solving SVP for $\mathcal{L}(\mathbf{B})$. The complexity of the method is $O(n \ln^2 n \ln\ln n)$ binary operations.

Notice that any PACDP with inputs $N_1, N_2, m\Delta, D$ for some $m \in \mathbb{N}$ may be reduced to $2m-1$ cases of PACDP with inputs $N_1, N_2 - \Delta\delta_1, \Delta, D$, where $\delta_1 \in (-m, m)$. For this aim we divide $\delta$ by $\Delta$ forming a quotient $\delta_1$ (with rounding towards zero) and remainder $\delta_0$ ($\delta_0$ has the same sign as $\delta$), i. e. get representation $\delta = \Delta\delta_1 + \delta_0$, where $|\delta_0| < \Delta$ and $|\delta_1| < m$. Therefore the bounds for $\Delta$ in the three methods described above may be increased by $m$ times with proportional growth of complexity.

Another method of solving PACDP under condition $\Delta < mD^2/N_1$ was introduced in [6]. This method is based on continued fractions technique and has complexity $O(mn \ln m \ln n \ln\ln n)$ binary operations provided $m = O(n)$.

Here we describe a new method which works under the same condition $\Delta < mD^2/N_1$, but is $\ln m$ times faster than one of [6]. It follows from (1) that the vector $(B_2, B_1)$ is a solution of a system of integer linear inequalities

$$\mathbf{v}_1 \leqslant \mathbf{x}\mathbf{A} \leqslant \mathbf{v}_2, \tag{2}$$

where

$$\mathbf{v}_1 = \Big(-\frac{N_1\Delta}{D}, 0\Big), \quad \mathbf{v}_2 = \Big(\frac{N_1\Delta}{D}, \frac{N_1}{D}\Big), \quad \mathbf{A} = \begin{pmatrix} -N_1 & 0 \\ N_2 & 1 \end{pmatrix}$$

and $\mathbf{x} = (x, y)$ is a vector of unknown variables.

The solution of the system (2) in geometry of numbers terms is equivalent to finding all points of a lattice

$$\mathcal{L} = \big\{(-N_1, 0)\,x + (N_2, 1)\,y \mid x, y \in \mathbb{Z}\big\}$$

inside a rectangle

$$\mathcal{R} = \Big\{(a, b) \in \mathbb{R}^2 |\ -\frac{N_1\Delta}{D} \leqslant a \leqslant \frac{N_1\Delta}{D},\ 0 \leqslant b \leqslant \frac{N_1}{D}\Big\}.$$

Each point $(a, b) \in \mathcal{L} \cap \mathcal{R}$ corresponds to a solution of (2) with

$$x = \frac{N_2 b - a}{N_1}, \quad y = b.$$

In the next section we describe the algorithm suggested by J. Franke and T. Kleinjung that finds all points of two dimensional lattice inside a rectangle. Using this algorithm one could easily formulate a practical procedure for solving PACDP provided the number of solutions of the system (2) is small.

Using the Gaussian volume heuristic we may estimate the number of solutions of system (2) by the number $\mathrm{Vol}(\mathcal{R})/\det(\mathcal{L})$, where

$$\mathrm{Vol}(\mathcal{R}) = \frac{2N_1^2\Delta}{D^2}$$

is the volume of rectangle $\mathcal{R}$ and $\det(\mathcal{L}) = N_1$ is the determinant of the lattice $\mathcal{L}$. Hence,

$$\frac{\mathrm{Vol}(\mathcal{R})}{\det(\mathcal{L})} < 2m$$

provided $\Delta < mD^2/N_1$.

In what follows we will use the following assumption.

The rectangle $\mathcal{R}$ contains no more than $2m$ points of the lattice $\mathcal{L}$.

Notice that for some special lattices and rectangles we cannot use the Gaussian volume heuristic to estimate the cardinality of their intersection. But in our experiments all random PACDP fulfill heuristic assumption.

The summary of the methods described above is presented in the table below.

Exponential methods for solving PACDP

| Technique | Origin | Restriction | Complexity |
|---|---|---|---|
| CF | [1] | $\Delta < mD^2/N_1$ | $O(mn^2 \ln n \ln\ln n)$ |
| SVP | [4] | $\Delta < mD^2/N_1$ | $O(mn\ln^2 n \ln\ln n)$ |
| CF | [6] | $\Delta < mD^2/N_1$ | $O(mn\ln m \ln n \ln\ln n)$ if $n = O(m\ln m)$ |
| Copp. | [1], [5] | $\Delta < mD^{\log_{N_1} D}$ | $O(mn^2 \ln n \ln\ln n)$ |
| FK | this paper | $\Delta < mD^2/N_1$ | $O(mn\ln n \ln\ln n)$ if $n = O(m)$ |

The method described in this paper is asymptotically faster than methods based on continued fractions technique and the method based on SVP provided $n = O(m)$. Here we specify the parameters of PACDP for which the method of this paper is asymptotically faster than the method based on the Coppersmith technique. Suppose

$$\log_2 D > \log_2 N_1 - \sqrt{\log_2 N_1} \quad \text{and} \quad \Delta = \left[mD^{\log_{N_1} D}\right] - 1.$$

Then $\Delta < mD^{\log_{N_1} D}$. On the one hand we may solve PACDP using Coppersmith technique and $O(mn^2 \ln n \ln\ln n)$ binary operations. However it follows from proposition 1 that

$$\Delta < \frac{2mD^2}{N_1}.$$

So on the other hand we may solve PACDP using lattice sieving and

$$O(2mn\ln n \ln\ln n) = O(mn\ln n \ln\ln n)$$

binary operations.

**Proposition 1.** *Let* $N, D \in \mathbb{N}$ *and* $c \in \mathbb{R}$ *be such that* $c > 0$ *and*

$$\log_c D > \log_c N - \sqrt{\log_c N}.$$

*Then* $D^{\log_N D} < cD^2/N$.

## 3. Algorithm suggested by J. Franke and T. Kleinjung

For $k \in \mathbb{N}$ and vector $\mathbf{v} \in \mathfrak{R}^k$, where $\mathfrak{R}$ is a ring, we denote the $i$-th coordinate as $v^{(i)}$, $i \in \overline{1,k}$ (i. e. $\mathbf{v} = (v^{(1)}, v^{(2)}, \ldots, v^{(k)})$ ). For matrix

$$\mathbf{A} = \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{pmatrix} \in \mathbb{Z}_{2,2},$$

vectors $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{Z}^2$ and integers $I_1, I_2 \in \mathbb{Z}$ we define subsets of $\mathbb{R}^2$:

$$\begin{aligned} \mathcal{L}(\mathbf{A}) &= \{\mathbf{a}_1 x + \mathbf{a}_2 y \mid x, y \in \mathbb{Z}\}, \\ \mathcal{R}(\mathbf{v}_1, \mathbf{v}_2) &= \{\mathbf{v} \in \mathbb{R}^2 \colon v_1^{(i)} < v^{(i)} < v_2^{(i)} \mid i = 1, 2\}, \\ \mathcal{S}(I_1, I_2) &= \{\mathbf{v} \in \mathbb{R}^2 \mid I_1 < v^{(1)} < I_2\}, \\ \mathcal{LS}(\mathbf{A}, I_1, I_2) &= \mathcal{L}(\mathbf{A}) \cap \mathcal{S}(I_1, I_2). \end{aligned}$$

**Definition.** We say that matrix

$$\mathbf{B} = \begin{pmatrix} \mathbf{b_1} \\ \mathbf{b_2} \end{pmatrix} \in \mathbb{Z}_{2,2}$$

is *FK-reduced with parameter* $I \in \mathbb{Z}_{>0}$ if:

1) $-I < b_1^{(1)} \leqslant 0$ and $0 \leqslant b_2^{(1)} < I$,
2) $b_1^{(2)} > 0$ and $b_2^{(2)} > 0$,
3) $b_2^{(1)} - b_1^{(1)} > I$.

We denote the set of FK-reduced with parameter $I$ matrices as $\mathcal{FK}_I(\mathbb{Z}_{2,2})$.

**Proposition 2.** *If matrix* $\mathbf{A} = \left(\begin{smallmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{smallmatrix}\right) \in \mathbb{Z}_{2,2}$ *satisfies the conditions*

1) $a_1^{(1)} < 0$ *and* $a_2^{(1)} > 0$,
2) $a_1^{(2)} \geqslant 0$ *and* $a_2^{(2)} > 0$,
3) $a_1^{(1)} + a_2^{(1)} < 0$,

*then for any integer* $I \leqslant |a_1^{(1)}|$ *there exists the unique decomposition* $\mathbf{A} = \mathbf{UB}$ *such that* $\mathbf{U} \in \mathbb{Z}_{2,2}$ *is invertible and* $\mathbf{B} \in \mathcal{FK}_I(\mathbb{Z}_{2,2})$.

**Proposition 3.** *If* $\mathbf{B} = \left(\begin{smallmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{smallmatrix}\right) \in \mathcal{FK}_I(\mathbb{Z}_{2,2})$, *then for any* $\mathbf{b} \in \mathcal{L}(\mathbf{B})$ *such that* $-I < b^{(1)} < I$ *and* $b^{(2)} > 0$ *the representation* $\mathbf{b} = x\mathbf{b}_1 + y\mathbf{b}_2$ *has non-negative coefficients* $x \geqslant 0$, $y \geqslant 0$.

Propositions 2 and 3 are proved in [2]. The algorithm 1 is a reformulation of a reduction procedure from [2].

**ALGORITHM 1** (FK REDUCTION)

**Input**: $I \in \mathbb{N},\ \mathbf{A} = \begin{pmatrix}\mathbf{a}_1\\ \mathbf{a}_2\end{pmatrix} : \mathbf{a}_1^{(1)} < 0,\ \mathbf{a}_2^{(1)} > 0,\ \mathbf{a}_1^{(2)} \geqslant 0,\ \mathbf{a}_2^{(2)} > 0,\ \mathbf{a}_1^{(1)} + \mathbf{a}_2^{(1)} < 0.$
**Output**: $\mathbf{B} \in \mathcal{FK}_I(\mathbb{Z}_{2,2}) : \mathcal{L}(\mathbf{A}) = \mathcal{L}(\mathbf{B})$

```
1  PROCEDURE Reduce(A, I)
2    b_1 ← a_1, b_2 ← a_2
3    WHILE 1 DO
4      b_1 ← b_1 + ⌊|b_1^(1) / b_2^(1)|⌋ · b_2
5      IF b_1^(1) > −I THEN
6        b_2 ← b_2 + ⌈(I − b_2^(1)) / b_1^(1)⌉ · b_1
7        RETURN (b_1, b_2)
8      b_2 ← b_2 + ⌊|b_2^(1) / b_1^(1)|⌋ · b_1
9      IF b_2^(1) < I THEN
10       b_1 ← b_1 + ⌈(b_1^(1) − I) / b_2^(1)⌉ · b_2
11       RETURN (b_1, b_2)
```

If a lattice basis is FK-reduced with parameter $I$, then we can successively enumerate all the lattice points inside a strip of a width $I$. The proof of correctness of the algorithm 2 is given in [2].

**ALGORITHM 2** (NEXT LATTICE POINT IN A STRIP)

**Input**: $I_1, I_2 \in \mathbb{Z} : I_1 < I_2, \mathbf{B} \in \mathcal{FK}_{I_2 - I_1}(\mathbb{Z}_{2,2}), \mathbf{a} \in \mathcal{LS}(\mathbf{B}, I_1, I_2)$
**Output**: $\mathbf{b} \in \mathcal{LS}(\mathbf{B}, I_1, I_2) : b^{(2)} = \min\{c^{(2)} \mid \mathbf{c} \in \mathcal{LS}(\mathbf{B}, I_1, I_2), c^{(2)} > a^{(2)}\}$

```
1  PROCEDURE StripNextPoint (B, a, I_1, I_2)
2    IF a^(1) ⩾ I_1 − b_1^(1) THEN
3      b ← a + b_1
4    ELSE IF a^(1) < I_2 − b_2^(1) THEN
5      b ← a + b_2
6    ELSE
7      b ← a + b_1 + b_2
8      RETURN b
```

**ALGORITHM 3** (NEXT LATTICE POINT IN A STRIP WITH STEP $D$)

**Input**: $I_1, I_2 \in \mathbb{Z} : I_1 < I_2,\ \mathbf{B} \in \mathcal{FK}_{I_2-I_1}(\mathbb{Z}_{2,2}),\ \mathbf{a} \in \mathcal{LS}(\mathbf{B}, I_1, I_2),\ D \in \mathbb{Z}$
**Output**: $\mathbf{b} \in \mathcal{LS}(\mathbf{B}, I_1, I_2) : b^{(2)} = \min\{c^{(2)} \mid \mathbf{c} \in \mathcal{LS}(\mathbf{B}, I_1, I_2), c^{(2)} > a^{(2)} + D\}$

1 PROCEDURE StripJumpPoint $(\mathbf{B}, \mathbf{a}, I_1, I_2, D)$
2 $J \leftarrow a^{(2)} + D$
3 $s \leftarrow \lfloor -b_2^{(1)} D / \det(\mathbf{B}) \rfloor,\ t \leftarrow \lfloor b_1^{(1)} D / \det(\mathbf{B}) \rfloor,\ \mathbf{b} \leftarrow \mathbf{a} + s\mathbf{b}_1 + t\mathbf{b}_2$
4 IF $b^{(1)} \geqslant I_1 - b_1^{(1)}$ THEN
5 $\mathbf{b} \leftarrow \mathbf{b} + \mathbf{b}_1$
6 IF $b^{(2)} > J$ THEN
7 $l \leftarrow \min\{\lfloor (b^{(1)} - I_1)/b_2^{(1)} \rfloor, \lfloor (b^{(2)} - J)/b_2^{(2)} \rfloor\}, \mathbf{b} \leftarrow \mathbf{b} - l\mathbf{b}_2$
8 ELSE
9 $l \leftarrow \min\left\{\lfloor (I_1 - b^{(1)})/b_1^{(1)} \rfloor, \lfloor (J - b^{(2)})/b_1^{(2)} \rfloor\right\},\ \mathbf{b} \leftarrow \mathbf{b} + l\,\mathbf{b}_1$
10 $\mathbf{b} \leftarrow$ StripNextPoint $(\mathbf{B}, \mathbf{b}, I_1, I_2, )$
11 ELSE IF $b^{(1)} < I_2 - b_2^{(1)}$ THEN
12 $\mathbf{b} \leftarrow \mathbf{b} + \mathbf{b}_2$
13 IF $b^{(2)} > J$ THEN
14 $l \leftarrow \min\{\lfloor (b^{(1)} - I_2)/b_1^{(1)} \rfloor, \lfloor (b^{(2)} - J)/b_1^{(2)} \rfloor\}, \mathbf{b} \leftarrow \mathbf{b} - l\mathbf{b}_1$
15 ELSE
16 $l \leftarrow \min\left\{\lfloor (I_2 - b^{(1)})/b_2^{(1)} \rfloor, \lfloor (J - b^{(2)})/b_2^{(2)} \rfloor\right\},\ \mathbf{b} \leftarrow \mathbf{b} + l\,\mathbf{b}_2$
17 $\mathbf{b} \leftarrow$ StripNextPoint $(\mathbf{B}, \mathbf{b}, I_1, I_2, )$
18 ELSE
19 $\mathbf{b} \leftarrow \mathbf{b} + \mathbf{b}_1 + \mathbf{b}_2$
20 RETURN $\mathbf{b}$

In the original description [2] of procedure StripJumpPoint steps 7 and 14 are omitted. In that case the procedure may not work. We prove the correctness of our version of procedure in the appendix.

**ALGORITHM 4** (LATTICE POINTS INSIDE A RECTANGLE)

**Input**: $\mathbf{A} = \begin{pmatrix} \mathbf{a_1} \\ \mathbf{a_2} \end{pmatrix} : \mathbf{a_1}^{(1)} < 0, \mathbf{a_2}^{(1)} > 0, a_1^{(2)} \geqslant 0, a_2^{(2)} > 0, a_1^{(1)} + a_2^{(1)} < 0,$
$\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{Z}^2 \colon \mathcal{R}(\mathbf{v}_1, \mathbf{v}_2) \neq \varnothing$

**Output**: $\mathcal{L}(\mathbf{A}) \cap \mathcal{R}(\mathbf{v}_1, \mathbf{v}_2)$

```
1 PROCEDURE RectAllPoints (A, v_1, v_2)
2   T ← ∅
3   B ← Reduce(A, v_2^(1) − v_1^(1))
4   b ← ⌊v_2^(1) / b_2^(1)⌋ · b_2
5   b ← StripJumpPoint(B, b, v_1^(1), v_2^(1), v_1^(2))
6   WHILE b^(2) < v_2^(2) DO
7     T ← T ∪ {b}
8     b ← StripNextPoint(B, b, v_1^(1), v_2^(1))
9   RETURN T
```

It is easy to see that the complexity of Algorithm 4 is the sum of complexities of Reduce procedure, StripJumpPoint procedure, and also the complexity of StripNextPoint procedure multiplied by the number of iterations of loop 6–8.

## 4. New algorithm for solving PACDP

Suppose that the natural $N_1$ and $N_2$ are the inputs of PACDP and $N_1 > N_2$. As mentioned in [1] the restriction $N_1 > N_2$ is not a limitation at all. If $N_1 < N_2$ we solve PACDP with the inputs $N_1$ and $N_2 - N_1 \cdot \lfloor N_2/N_1 \rfloor$.

If $N_1 > N_2$, then the matrix

$$\begin{pmatrix} -N_1 & 0 \\ N_2 & 1 \end{pmatrix}$$

satisfies the restrictions from proposition 2 hence we can apply algorithm 4 to find all solutions of system of integer inequalities (2). Then we have to check each solution of (2) for being a solution for PACDP. In practical implementation we would like to check each solution individually, rather than to collect all the solutions in memory. This leads to the following algorithm for solving PACDP.

**ALGORITHM 5** (PACDP)

**Input**: $N_1, N_2, \Delta, D, m$ — natural numbers such that $N_1 > N_2,\ \Delta < mD^2/N_1$

**Output**: $\mathcal{W} \subset \mathbb{N}^2$ — a set of pairs $(B_1, B_2)$ such that $N_1 = AB_1,\ N_2 = AB_2 + \delta$ with $A \geqslant D,\ |\delta| < \Delta$

1 PROCEDURE SolvePACDP $(N_1, N_2, \Delta, D, m)$
2 $\quad \mathbf{A} \leftarrow \left(\begin{smallmatrix} -N_1 & 0 \\ N_2 & 1 \end{smallmatrix}\right),\ \mathbf{v_1} \leftarrow \left(\lfloor -\frac{N_1\Delta}{D} \rfloor, 0\right),\ \mathbf{v_2} \leftarrow \left(\lceil \frac{N_1\Delta}{D} \rceil, \lceil \frac{N_1}{D} \rceil\right)$
3 $\quad \mathcal{W} \leftarrow \varnothing$
4 $\quad \mathbf{B} \leftarrow \mathsf{Reduce}\,(\mathbf{A}, v_2^{(1)} - v_1^{(1)})$
5 $\quad \mathbf{b} \leftarrow \lfloor v_2^{(1)} / b_2^{(1)} \rfloor \cdot \mathbf{b}_2$
6 $\quad \mathbf{b} \leftarrow \mathsf{StripJumpPoint}\,(\mathbf{B}, \mathbf{b}, v_1^{(1)}, v_2^{(1)}, v_1^{(2)})$
7 $\quad$ FOR $i = 1, 2, \ldots, 2m$ DO
8 $\quad\quad$ IF $b^{(2)} \mid N_1$ THEN
9 $\quad\quad\quad B_1 \leftarrow b^{(2)},\ B_2 \leftarrow \frac{N_2 B_1 - b^{(1)}}{N_1}$
10 $\quad\quad\quad \mathcal{W} \leftarrow \mathcal{W} \cup \{(B_1, B_2)\}$
11 $\quad\quad \mathbf{b} \leftarrow \mathsf{StripJumpPoint}\,(\mathbf{B}, \mathbf{b}, v_1^{(1)}, v_2^{(1)})$
12 $\quad$ RETURN $\mathcal{W}$

Algorithm 5 is correct for inputs satisfying the Assumption. Let $n = \max\{[\ln N_1],\ [\ln N_2]\}$. The complexity of step 4 is $O(n^2)$ binary operations (the Euclidean algorithm). The complexity of loop 7–11 is $O(mn \ln n \ln\ln n)$ binary operations. Therefore we obtain the following total complexity estimate.

**Proposition 4.** *If $n{=}O(m)$, then the complexity of Algorithm* 5 *is $O(mn \ln n \ln\ln n)$ binary operations, where $n = \max\{[\ln N_1],\ [\ln N_2]\}$.*

# References


[1] Howgrave-Graham N., "Approximate integer common divisors", *Lect. Notes Comput. Sci.*, **2146** (2001), 51–66

[2] Franke J. and Kleinjung T., "Continued fractions and lattice sieving". In: "*SHARCS 2005*", http://www.hyperelliptic.org/tanja/SHARCS/talks/FrankeKleinjung.pdf.

[3] May A., Ritzenhofen M., "Implicit factoring: on polynomial time factoring given only an implicit hint", *Lect. Notes Comput Sci.*, **5443** (2009), 1–14.

[4] Van Dijk M., Gentry C., Halevi S., Vaikuntanathan V., "Fully homomorphic encryption over the integers", *Lect. Notes Comput Sci.*, **6110** (2010), 24–43.

[5] Sarkar S., Maitra S., "Approximate integer common divisor problem relates to implicit factorization", *IEEE Trans. Inf. Theory*, **57** (2011), 4002–4013.

[6] Zhukov K. D., "Approximate common divisor problem and continued fractions", *Mathematical Aspects of Cryptography*, **7**:2 (2016), 61–70.

# A. Proof of proposition 1

*Proof.* It follows from the condition of proposition that

$$\log_c^2 \frac{N}{D} < \log_c N.$$

Therefore

$$\left(\frac{\log_N \frac{N}{D}}{\log_N c}\right)^2 < \frac{1}{\log_N c}.$$

Hence,

$$(1 - \log_N D)^2 < \log_N c$$

or

$$1 + \log_N^2 D < 2\log_N D + \log_N D \log_D c.$$

Divide both sides of the last inequality by $\log_N D$, then exponentiate:

$$D^{\log_D N + \log_N D} < D^{2+\log_D c}.$$

The statement of proposition follows immediately from the last inequality. □

# B. Proof of correctness of Algorithm 3

*Proof.* Denote $(\mu, \nu) = (0, D)\mathbf{A}^{-1}$. After step 3 the equalities $s = \lfloor \mu \rfloor$, $t = \lfloor \nu \rfloor$ hold. Hence, after step 3 the estimates $I_1 - b_2^{(1)} \leqslant b^{(1)} < I_2 - b_1^{(1)}$ and $b^{(2)} < J$ hold.

Suppose $b^{(1)} \geqslant I_1 - b_1^{(1)}$. We are going to show that the algorithm is correct in this case (the correctness in the case $b^{(1)} < I_2 - b_2^{(1)}$ may be proved in the same way). After step 5 the following inclusion holds $\mathbf{b} \in \mathcal{L}(\mathbf{B}) \cap \mathcal{S}(I_1, I_2)$. Consider the cases $b^{(2)} > J$ and $b^{(2)} \leqslant J$ separately.

If after step 5 the inequality $b^{(2)} > J$ holds, then for

$$l = \min\left\{\lfloor (b^{(1)} - I_1)/b_2^{(1)} \rfloor, \lfloor (b^{(2)} - J)/b_2^{(2)} \rfloor\right\}$$

the second coordinate of the vector $\mathbf{b} - l\mathbf{b}_2$ is minimal such that $b^{(2)} - lb_2^{(2)} \leqslant J$. To see this let us consider the vector $\mathbf{b}'$ such that its second coordinate is maximal possible satisfying the condition $b'^{(2)} < b^{(2)} - lb_2^{(2)}$. According to Algorithm 2 the vector $\mathbf{b}'$ is the difference of $\mathbf{b} - l\mathbf{b}_2$ and either $\mathbf{b}_1$, either $\mathbf{b}_2$ or $\mathbf{b}_1 + \mathbf{b}_2$. If $\mathbf{b}' = \mathbf{b} - (l+1)\mathbf{b}_2$, then $b^{(2)} \leqslant J$ by the definition of $l$. Else

$$\mathbf{b}' = \mathbf{a} + s\mathbf{b}_1 + (t - l')\mathbf{b}_2 \quad (l' = l \text{ or } l' = l + 1),$$

hence $b^{(2)} \leqslant J$.

If after step 5 the inequality $b^{(2)} \leqslant J$ holds, then for

$$l \leftarrow \min\left\{\lfloor (I_1 - b^{(1)})/b_1^{(1)} \rfloor, \lfloor (J - b^{(2)})/b_1^{(2)} \rfloor\right\}$$

the second coordinate of the vector $\mathbf{b} + l\mathbf{b}_1$ is maximal such that $b^{(2)} + lb_1^{(1)} \leqslant J$. To see this let us consider the vector $\mathbf{b}''$ such that its second coordinate is minimal possible satisfying the condition $b''^{(2)} > b^{(2)} + lb_1^{(2)}$. According to Algorithm 2 the vector $\mathbf{b}''$ is the sum of $\mathbf{b} + l\mathbf{b}_1$ and $\mathbf{b}_1$ or $\mathbf{b}_2$ (or both). If $\mathbf{b}'' = \mathbf{b} + (l+1)\mathbf{b}_1$, then $b^{(2)} > J$ by the definition of $l$. Else $\mathbf{b}'' = \mathbf{a} + (s + 1 + l')\mathbf{b}_1 + (t+1)\mathbf{b}_2$ ($l' = l$ or $l' = l + 1$), hence $b''^{(2)} > J$. Notice that in this case algorithm outputs $\mathbf{b}''$.

Consider step 18 By the transition condition we have $b^{(1)} + b_1^{(1)} < I_1$. Since the matrix $\mathbf{B}$ is FK-reduced with parameter $(I_2 - I_1)$, then $b_1^{(1)} > I_1 - I_2$. Hence, $b^{(1)} < I_2$. In the same manner we get an estimate $b^{(1)} > I_1$. Therefore $b \in \mathcal{L}(\mathbf{B}) \cap \mathcal{S}(I_1, I_2)$. Using Algorithm 2 we calculate the next point as on step 19. After step 19 the estimate $b^{(2)} > a^{(2)} + D$ holds. □