Master’s Thesis

The Approximate Common Divisor Problem: Two Algorithmic Solutions

Helen Riedtmann

Supervisor: Prof. Dr. Joachim Rosenthal

October 7, 2013
Acknowledgments

I would like to thank Prof. Dr. Joachim Rosenthal for introducing me to the subject of cryptography and for supervising my thesis.

I would also like to express my gratitude to my boyfriend, Christoph, for his infinite patience and understanding during the last year, even though he has retained a healthy suspicion of the notion of the infinite.

Last but not least, I would like to take this opportunity to thank my parents for supporting me financially during my studies.
Contents

1 Introduction 2

2 Approximate Common Divisor Algorithms 2

3 The Continued Fraction-Based Algorithm 4
   3.1 Continued Fractions ................................. 4
   3.2 The Partially Approximate Common Divisor Problem .... 14
   3.3 The General Approximate Common Divisor Problem .... 20

4 The Lattice-Based Algorithm 27
   4.1 The Partially Approximate Common Divisor Problem ... 27
   4.2 Finding Roots of a Polynomial over the Integers ....... 34
      4.2.1 Finding Roots Modulo a Prime .................. 35
      4.2.2 Hensel Lifting ................................. 38
      4.2.3 The Integer Root Finding Algorithm .............. 43
   4.3 The Lattice-Based PACDP Algorithm ................... 46
   4.4 A Generalization of the Partially Approximate Common Di-
      visor Problem ........................................... 49
   4.5 The General Approximate Common Divisor Problem .... 54

5 Conclusion 58

A Appendix: The LLL-Algorithm 60
1 Introduction

We are all familiar with the notion of the greatest common divisor of two integers, say $a$ and $b$. However, can we still compute their greatest common divisor if we are only given approximate values of $a$ and $b$?

All questions of this type are subsumed under the umbrella term approximate common divisor problem. More precisely, given bounds $M$, $X$ and $Y$, an approximate common divisor algorithm with input $a$ and $b$ should compute all natural numbers $d$ greater than $M$ with the property that there exist two integers $x_0$ and $y_0$ whose absolute value is less than $X$ and $Y$, respectively, such that $d$ is the greatest common divisor of $a + x_0$ and $b + y_0$. In addition, the bounds $X$ and $Y$ may depend on $d$. In this context, the natural number $d$ is called an approximate common divisor of $a$ and $b$ with noise $(x_0, y_0)$ in $\mathbb{Z}^2$ ([1]). A solution to an approximate common divisor problem thus consists of the concatenation of an approximate common divisor and its noise. From a cryptographic perspective, the solution set might be regarded as the private information of the problem, while the parameters $a$, $b$, $M$, $X$ and $Y$ on which it depends constitute its public information.

If the exact value of the parameter $b$ is known, or equivalently, if the bound $Y$ is set to be zero, the corresponding problem is referred to as a partially approximate common divisor problem (PACDP) since only one of the input values is approximate. Accordingly, a general approximate common divisor problem (GACDP) possesses strictly positive bounds $X$ and $Y$.

In the next section, we will present three algorithms that compute solutions to specific approximate common divisor problems, which we will then justify in Sections 3 and 4. While Section 3 uses properties of the continued fraction approximation of a rational number, Section 4 is based on the LLL-reduction of lattices.

2 Approximate Common Divisor Algorithms

In this section, we discuss a few simplifications of the approximate common divisor problem in order to arrive at reasonable conditions on its parameters, before stating the PACDP and GACDP algorithms.

Let us first focus on the partially approximate case. Then we may assume without loss of generality that the integers $a$ and $b$, whose approximate common divisors are to be computed, are non-negative. Indeed, if a natural number $d$ divides both $(a + x_0)$ and $b$, it is also a common factor of $(a + k b + x_0)$ and $-b$ for any $k$ in $\mathbb{Z}$. Hence, we may even assume that $0 \leq a < b$. The Continued Fraction-Based PACDP Algorithm, which will be
Algorithm (Continued Fraction-Based PACDP).
Input: Two natural numbers \( a \) and \( b \) satisfying \( b/8 \leq a \leq 7b/8 \).
Output: The set \( S \) of all \( (d, x_0) \) in \( \mathbb{N} \times \mathbb{Z} \) so that \( x_0 \) is bounded by \( X(d) = d^2/2b \) and \( d \) is the greatest common divisor of \( a + x_0 \) and \( b \).

The algorithm produces the solution set of the PACDP with public information \( a, b, M = 1, X(d) = d^2/2b \) and \( Y = 0 \). Notice that the bounds are determined by the values of \( a \) and \( b \).

One advantage of the lattice-based approach to approximate common divisor problems is that the bounds \( X \) and \( M \) are not specified by the following algorithm, which will be established in Section 4.3.

Algorithm (Lattice-Based PACDP).
Input: Natural numbers \( a, b, M \) and \( X \) so that \( \log_b X < (\log_b M)^2 \) and \( \max\{a, M, X\} < b \).
Output: The solution set \( S \) of all \( (d, x_0) \) in \( \mathbb{Z}^2 \) so that \( d \) is the greatest common divisor of \( a + x_0 \) and \( b \), \( d \) is greater than \( M \) and \( x_0 \) is less than \( X \) in absolute value, or "failure".

In fact, the stipulation that \( b \) be strictly greater than the other parameters represents no substantial additional restriction. Having already seen that this inequality holds for the parameter \( a \), it is sufficient to consider the cases \( M \geq b \) and \( X \geq b \). Suppose that \( M \) is greater than \( b \), in particular, \( M \) is strictly greater than any proper factor of \( b \). Hence, \( a \) and \( b \) have at most one approximate common divisor, namely \( b \) itself. If, on the other hand, the bound \( X \) is greater than \( b \), any factor \( d \) of \( b \) with \( d \geq M \) is an approximate common divisor of \( a \) and \( b \), regardless of the value of \( a \). Therefore, one might regard approximate common divisor problems whose bounds exceed the value of the input \( b \) as the degenerate case.

In general, the partially approximate common divisor problem solved by the lattice-based algorithm can be reduced to factoring \( b \). Indeed, whenever \( d \) is an approximate common divisor of \( a \) and \( b \), it is a factor of \( b \) that is greater than \( M \) with the property that there exists an integer \( x_0 \) bounded by \( X \) such that \(-x_0\) is equivalent to \( a \) modulo \( d \). We thus conclude that the problem is only interesting if the input \( b \) is hard to factor.

Remark 2.1. For later reference, we also mention that a brute-force attack to the PACDP with public information \( a, b, M \) and \( X \) would be to verify whether the greatest common divisor of \( a + x_0 \) and \( b \) is greater than \( M \) for all integers \( x_0 \) lying in the interval \([-X, X]\). This involves computing \( 2X + 1 \) greatest common divisors of integers of size \( O(\log b) \), and thus has a running time of \( O(X(\log b)^2) \) bit operations.

Let us now turn our attention the the general approximate common divisor problem. Like in the partially approximate case, we may assume the
parameters $a$ and $b$ to be non-negative. However, unlike above, considering the problem with input $-b$ instead of $b$ also affects the noise. In fact, the three tuple $(d, x_0, y_0)$ is a solution to the GACDP with parameter $b$ if and only if $(d, x_0, -y_0)$ solves the problem with parameter $-b$. Given that the general problem is symmetric with respect to $a$ and $b$, we may restrict ourselves to the case $0 \leq a \leq b$ without loss of generality. Whenever $a$ is sufficiently far away from 0 as well as $b$, the following continued fraction-based algorithm, which will be discussed in Section 4.5, returns the solution set to the GACDP with parameters $a$, $b$, $M = 2\sqrt{b}$ and $X(d)$.

**Algorithm (Continued Fraction-Based GACDP).**

**Input:** Two natural numbers $a$ and $b$ with the property that $(\sqrt{b} - 1)/4 \leq a \leq b - (\sqrt{b} - 1)/2$.

**Output:** The set $S$ of all three tuples $(d, x_0, y_0)$ in $\mathbb{Z}^3$ so that $d \geq 2\sqrt{b}$, $(x_0, y_0)$ are both bounded by $X(d) = \min\{d^2/4b, b/2d - 1/4\}$ and $d$ is the greatest common divisor of $a + x_0$ and $b + y_0$.

### 3 The Continued Fraction-Based Algorithm

Given that the Euclidean algorithm provides an efficient standard method for computing the greatest common divisor of two integers, the most intuitive approach to determining approximate common divisors of two integers probably lies in somehow adapting the Euclidean algorithm to meet the new requirements. The theory on continued fractions presented in the next section will allow us to formalize this intuition, resulting in the partially and general approximate common divisor algorithms presented in Section 3.2 and 3.3, respectively.

#### 3.1 Continued Fractions

This section is dedicated to providing sufficient theoretical background on continued fractions to prove Proposition 3.6, on which the ACDP-algorithms presented in Sections 3.2 and 3.3 are based. We attempt to give nothing but the information necessary for this paper to be self-contained; therefore, what follows is not a discussion of the general theory on continued fractions but a direct route to the statements needed for the application to the approximate common divisor problem. To this end, we will prove some properties of the continued fraction approximation of a rational number, after having established a connection between continued fractions and the Euclidean semigroup.

We will restrict ourselves to finite continued fractions since we are primarily interested in the continued fraction approximation of rational numbers. Letting $[q_1, \ldots, q_n]$ be a sequence of strictly positive integers, the continued
approach is given by

\[ \frac{1}{q_1 + \frac{1}{q_2 + \frac{1}{\ddots + \frac{1}{q_n}}}}. \]

For ease of notation, we will sometimes denote the continued fraction associated to a sequence of positive integers by the sequence itself. As expressions of this form are not well suited to theoretical considerations, we introduce a new concept which will prove to be equivalent.

**Definition 3.1.** The Euclidean semigroup $E$ is defined by

\[ E = \left\{ \begin{pmatrix} s & u \\ t & v \end{pmatrix} \in GL_2(\mathbb{Z}) : 0 \leq s \leq u \leq v \text{ and } s \leq t \leq v \right\}. \]

Note that $E$ is closed under matrix multiplication, that is to say it is indeed a semigroup. Moreover, the identity matrix does not lie in $E$ and thus $E$ cannot be a group. The Euclidean semigroup is closely related to the Euclidean algorithm, which will become apparent while proving the following

**Lemma 3.2.** The Euclidean semigroup is generated by the matrices of the form $E(q) = \begin{pmatrix} 0 & 1 \\ 1 & q \end{pmatrix}$ where $q$ is some strictly positive integer. Moreover, this factorization is unique.

**Proof.** Letting the matrix $A = \begin{pmatrix} s & u \\ t & v \end{pmatrix}$ be an element of the Euclidean semigroup, we will prove that there exists a unique positive integer $q$ so that $E(q)^{-1}A$ also lies in $E$, unless $s$ is equal to 0. As $E(q)^{-1}A$ has the additional property that its upper left entry is strictly less than $s$, the proposition follows by induction on the upper left entry of $A$, given that the statement holds in the base case $s = 0$. Indeed, if $s$ is equal to 0, $A$ must be of the form $A = \begin{pmatrix} 0 & 1 \\ 1 & v \end{pmatrix} = E(v)$ as it is unimodular by definition, which shows the existence of such a factorization. By observing that any product consisting of two or more factors has a strictly positive left upper entry, uniqueness follows immediately.

If $s$ is greater than 0, we have that

\[ E(q)^{-1}A = \begin{pmatrix} -q & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} s & u \\ t & v \end{pmatrix} = \begin{pmatrix} t - qs & v - qu \\ s & u \end{pmatrix}. \]

We consider the cases $s = 1$ and $s > 1$ separately. Assuming $s$ to be equal to 1, $E(q)^{-1}A = \begin{pmatrix} t - q & v - qu \\ 1 & u \end{pmatrix}$ is an element of $E$ if and only if $q$ satisfies
$0 \leq t - q \leq 1$ and $t - q \leq v - qu \leq u$. By the first inequality, $q$ is equal to either $t$ or $t - 1$, and in consequence the second inequality simplifies to either $0 \leq \det A \leq u$ or $1 \leq \det A + u \leq u$. Hence, whenever the determinant of $A$ is equal to 1, $E(q)^{-1}A$ lies in $E$ if and only if $q$ equals $t$, in which case the upper left entry of $E(t)^{-1}A$ is strictly less than 1. On the other hand, if the determinant of $A$ is negative, the inequalities can only be satisfied by setting $q = t - 1$, which is, in fact, a valid option since both $t$ and $u$ are necessarily strictly greater than 1. Notice, however, that the upper left entry is not decreased. This presents no obstacle since the determinant of $E(t)^{-1}E(t)^{-1}A$ vanishes.

In order to prove the claim in case $s$ is strictly greater than 1, we first establish that any unimodular matrix $A = \left( \begin{smallmatrix} s & u \\ t & v \end{smallmatrix} \right)$ satisfying $t > 1$ as well as $0 \leq s \leq t \leq v$ is an element of the Euclidean semigroup. Indeed, under these conditions, we have that

$$s = \frac{tu + \det A}{v} \leq \frac{vu + 1}{v} = u + \frac{1}{v},$$

and

$$u = \frac{sv - \det A}{t} \leq \frac{tv + 1}{t} = v + \frac{1}{t},$$

which implies that $s \leq u \leq v$ since both $1/v$ and $1/t$ are strictly less than 1. Given that $s > 1$, $E(q)^{-1}A$ thus lies in the Euclidean semigroup if and only if $0 \leq t - qs \leq s$. As $t$ is not equivalent to 0 modulo $s$, there exists exactly one integer $q$ that satisfies this inequality, which is strictly positive since $s \leq t$. In addition, the upper left entry of $E(q)^{-1}A$ satisfies $t - q s < s$ as desired.

The reductive argument used in this proof provides a connection between the Euclidean semigroup and the Euclidean algorithm, motivating the name of the former. By iterating the reduction step outlined while justifying the claim in the case that $s$ is strictly greater than 1, any matrix $A = \left( \begin{smallmatrix} s & u \\ t & v \end{smallmatrix} \right) \in E$ can be written as a product of the form

$$E(q_1) \cdots E(q_n)A' = E(q_1) \cdots E(q_n) \left( \begin{smallmatrix} s' & u' \\ t' & v' \end{smallmatrix} \right)$$

where $A'$ is an element of the Euclidean semigroup such that $s' \leq 1$. In fact, setting $r_{-1} = t$ and $r_0 = s$, the $q_i$ are determined by the algorithmic prescription

$$r_{i+1} \equiv r_{i-1} \mod r_i \text{ (so that } 0 \leq r_{i+1} < r_i); \quad q_{i+1} = \frac{r_{i-1} - r_{i+1}}{r_i},$$

while $r_i > 1$. This is the Euclidean algorithm with the integers $s$ and $t$ as input, except that it terminates as soon as $r_i \leq 1$ instead of $r_i = 0$. Given
that $s' \leq 1$, there are three possible factorizations of $A'$, which completes the factorization algorithm.

Notice for later use that initializing the Euclidean algorithm by $r'_{-1} = v$ and $r'_0 = u$ instead of $r_{-1} = t$ and $r_0 = s$ results in the same quotients even though the remainders might be different. Let us first consider the case $u > 1$. By construction, the quotient $q_1$ obtained through the factorization algorithm satisfies

$$E(q_1)^{-1} \begin{pmatrix} s & u \\ t & v \end{pmatrix} = \begin{pmatrix} t - q_1s & v - q_1u \\ s & u \end{pmatrix} \in E,$$

and hence $0 \leq v - q_1u \leq u$. Given that $u$ and $v$ are coprime, there exists exactly one integer with this property, namely the quotient $q'_1$ associated to the input values $r'_{-1} = v$ and $r'_0 = u$ into the Euclidean algorithm, entailing that $q_1 = q'_1$. By an analogous argument, the subsequent quotients are identical as long as the remainder corresponding to $u$ is strictly greater than 1. In case $u = 1$, the quotient $q'_1$ is equal to $v$, which agrees with the factorization algorithm if $s = 0$; however, if $s = 1$, the factorization algorithm outputs $q_1 = v - 1$ and $q_2 = 1$. We will see later on, that whether the factorization finishes with $E(q-1)E(1)$ or just $E(q)$ for some natural number $q \geq 2$ is of no great importance for our purposes.

In order to state how continued fractions and elements of the Euclidean semigroup are related, we observe that any invertible $2 \times 2$-matrix over the rationals acts on the set of all one-dimensional subspaces of the rational plane $\mathbb{Q}^2$, and thus on the rational projective line. Equating $x \in \mathbb{Q}$ with the subspace generated by $\begin{pmatrix} x \\ 1 \end{pmatrix}$ and $\infty$ with the horizontal line, any matrix $A = \begin{pmatrix} a & c \\ b & d \end{pmatrix} \in GL_2(\mathbb{Q})$ corresponds to a map from to projective line $\mathbb{Q} \cup \{\infty\}$ onto itself by

$$Ax = \begin{cases} (ax + c)/(bx + d), & \text{if } bx + d \neq 0 \\ \infty, & \text{if } bx + d = 0 \end{cases} \text{ for } x \in \mathbb{Q},$$

and

$$A\infty = \begin{cases} a/b, & \text{if } b \neq 0 \\ \infty, & \text{if } b = 0. \end{cases}$$

This map, called the Möbius transformation of the rational projective line induced by $A$, allows us to formalize the connection between continued fractions and the Euclidean semigroup.
Lemma 3.3. Letting \([q_1, \ldots, q_n]\) be a finite sequence of strictly positive integers, the following equation is satisfied:

\[
E(q_1)E(q_2) \cdots E(q_n)0 = \frac{1}{q_1 + \frac{1}{q_2 + \frac{1}{\ddots + \frac{1}{q_n}}}}.
\]

Proof. We prove a slightly stronger claim that lends itself to induction on \(n\):

Let \(x\) be a nonnegative rational number, then

\[
E(q_1) \cdots E(q_{n-1})E(q_n)x = \frac{1}{q_1 + \frac{1}{\ddots + \frac{1}{q_n + x}}}.
\]

The claim clearly holds if \(n\) is equal to 1. Assuming it to hold for some natural number \(n\), we have that

\[
E(q_1) \cdots E(q_{n-1})E(q_n)E(q_{n+1})x
= E(q_1) \cdots E(q_{n-1})E(q_n) \frac{1}{q_{n+1} + x}
= \frac{1}{q_1 + \frac{1}{\ddots + \frac{1}{q_{n-1} + x}}}
= \frac{1}{q_1 + \frac{1}{\ddots + \frac{1}{q_n + \frac{1}{q_{n+1} + x}}}}
\]

as desired. \(\square\)

According to Lemma 3.2, Lemma 3.3 provides a surjective map from the Euclidean semigroup onto the set of all continued fractions. Viewing continued fractions form this perspective will allow us to prove the facts Howgrave-Graham uses in his algorithm for solving approximate common divisor problems based on the continued fraction approximation of rational numbers.

Proposition 3.4. Let \(x\) be a rational number bounded by \(0 < x \leq 1\). Then \(x\) can be written as a continued fraction. Moreover, there are at most two finite sequences of positive integers whose continued fractions are equal to \(x\).
Notice that any rational number \( x \) which can be expressed as a continued fraction necessarily satisfies \( 0 < x \leq 1 \). Proposition 3.4 states that this condition is also sufficient.

**Proof.** Let \( u/v \) be a fraction in lowest terms with the property that \( 0 < u \leq v \). By the two preceding lemmas, the existence of a continued fraction whose value is equal to \( u/v \) can be shown by constructing natural numbers \( s \) and \( t \) such that \( A = \begin{pmatrix} s & u \\ t & v \end{pmatrix} \) is an element of the Euclidean semigroup.

If \( u \) is equal to \( v \) and hence \( u = v = 1 \), the matrix \( A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \) satisfies \( A0 = u/v \). We therefore restrict ourselves to the case \( u < v \). Given that \( u \) and \( v \) are coprime, there exist \( s, t \in \mathbb{Z} \) such that \( sv - tu = 1 \). In addition, we may choose \( t \) so that \( 0 \leq t < v \). As \( u \) is less than \( v \), the matrix \( A \) lies in the Euclidean semigroup if and only if \( s \) is nonnegative and satisfies \( s \leq \min\{t, u\} \). Clearly, \( s \) is greater than \( 1 \); otherwise it would follow that \( 1 = sv - tu \leq -tu \leq 0 \). We claim that \( s \) also satisfies the condition \( s \leq \min\{t, u\} \). Indeed, if \( s \) were greater than \( \min\{t, u\} + 1 \), we would have

\[
1 = sv - tu \geq (\min\{t, u\} + 1)v - tu > (\min\{t, u\} + 1)v - \min\{t, u\}v = v
\]

since both \( t \) and \( u \) are strictly less than \( v \). Hence, it would hold that \( 0 < u < v < 1 \), which is a contradiction. Therefore, the matrix \( A = \begin{pmatrix} s & u \\ t & v \end{pmatrix} \) is an element of the Euclidean semigroup, which establishes the first statement of the proposition.

It remains to show that the matrix \( A \) constructed above is one of at most two possible choices. We first observe that two unimodular matrices with positive entries map \( 0 \) to the same rational number if and only if their second columns are identical. Since the determinant of any element of the Euclidean semigroup is either \( 1 \) or \( -1 \), it is sufficient to prove that any two elements of the Euclidean semigroup with the same determinant and the same second column agree: Assuming \( A = \begin{pmatrix} s & u \\ t & v \end{pmatrix} \in E \) and \( A' = \begin{pmatrix} s' & u \\ t' & v \end{pmatrix} \in E \) to satisfy \( \det A = \det A' \), we have that

\[
\det AA'^{-1} = \begin{pmatrix} v & -u \\ -t & s \end{pmatrix} \begin{pmatrix} s' & u \\ t' & v \end{pmatrix} = \begin{pmatrix} \det A' & 0 \\ st' - s't & \det A \end{pmatrix} = \det A \begin{pmatrix} 1 & 0 \\ c & 1 \end{pmatrix}
\]

for some integer \( c \), and thus

\[
A' = A \begin{pmatrix} 1 & 0 \\ c & 1 \end{pmatrix} = \begin{pmatrix} s + cu & u \\ t + cv & v \end{pmatrix}.
\]

Given that \( A \) and \( A' \) lie in the Euclidean semigroup, both \( 1 \leq t \leq v \) and \( 1 \leq t + cv \leq v \) are satisfied, which implies that \( c = 0 \) and hence \( A = A' \).  

It is not surprising that most rational numbers do not have a unique representation as a continued fraction. In fact, any rational number \( x \) bounded
by $0 < x < 1$ has exactly two continued fraction representations. Indeed,
whenever the last coefficient $q_n$ is not equal to 1, the continued fractions
associated to $[q_1, \ldots, q_{n-1}, q_n]$ and $[q_1, \ldots, q_{n-1}, q_n - 1, 1]$ have the same value. Notice that the respective lengths of the two sequences do not have the same parity, which accounts for the fact that the two elements of the Euclidean semigroup corresponding to them have distinct determinants. In order to obtain a unique continued fraction which equals a given rational number lying in the open unit interval, one might require either that the last term be strictly greater than 1 or that the determinant of the associated matrix be positive (i.e. that that the number of terms be even). Neither requirement works for the only exception to the rule that continued fractions come in pairs which have the same value. From now on, we exclude the exception $x = 1$ since statements about it are trivial in any case.

Therefore, any rational number $x$ satisfying $0 < x < 1$ corresponds to exactly one continued fraction whose last term is strictly greater than 1, which is called the canonical continued fraction representation of $x$. The continued fraction approximation of $x$ is obtained by truncating the sequence of its terms; more formally,

**Definition 3.5.** Let $[q_1, \ldots, q_n]$ be the canonical continued fraction representation of a rational number $x \in (0, 1)$. For $i = 1, \ldots, n$, the $i$-th convergent to $x$ is given by the fraction $s_i/t_i = [q_1, \ldots, q_i]$. The sequence consisting of all convergents is called the continued fraction approximation of $x$.

What we have done so far enables us to explicitly compute the continued fraction approximation of any given rational number $x = u/v$ lying in the open unit interval. In a first step, we initialize the Euclidean algorithm by setting $r_{-1} = v$ and $r_0 = u$ to obtain the finite sequence $[q_1, \ldots, q_n]$ of quotients, which has the property that

$$E(q_1) \cdots E(q_n) = \begin{pmatrix} s & u \\ t & v \end{pmatrix}$$

for some integers $s$ and $t$ and satisfies the condition that $q_i$ be strictly positive for all $i \leq n - 1$ and $q_n \geq 2$. We then determine the matrices $A_i = E(a_1) \cdots E(a_i)$ for $i = 1, \ldots, n$ as the fraction $s_i/t_i = A_i 0$ is the $i$-th convergent to $x$. As an illustration, we implement this procedure on a concrete example: Letting $x = 3/20$, we have that $q_1 = 6$, $q_2 = 1$ and $q_3 = 2$, and hence

$$A_1 = E(q_1) = \begin{pmatrix} 0 & 1 \\ 1 & 6 \end{pmatrix},$$

$$A_2 = A_1 E(q_2) = \begin{pmatrix} 1 & 1 \\ 6 & 7 \end{pmatrix}$$

and

$$A_3 = A_2 E(q_3) = \begin{pmatrix} 1 & 3 \\ 7 & 20 \end{pmatrix}.$$
The continued fraction approximation to $3/20$ is thus given by

$$\frac{1}{6}, \frac{1}{7} \text{ and } \frac{3}{20}.$$ 

The following proposition enumerates some properties of the continued fraction approximation of rational numbers, which will become relevant for the application to approximate common divisor problems.

**Proposition 3.6.** Let $s_1/t_1, \ldots, s_n/t_n$ be the convergents in the continued fraction approximation of some rational number $x$ with canonical continued fraction representation $[q_1, \ldots, q_n]$. Then the following statements hold:

(i) Setting $s_{-1} = 1$, $s_0 = 0$ and $t_{-1} = 0$, $t_0 = 1$, the convergents conform to the recursive formula $s_{i+1} = s_{i-1} + q_i s_i$ and $t_{i+1} = t_{i-1} + q_i t_i$ for $1 \leq i \leq n - 1$.

(ii) The denominators $t_1, \ldots, t_n$ form a strictly increasing sequence. Moreover, if $(f_i)_{i \geq 0}$ denotes the Fibonacci sequence with $f_0 = 1 = f_1$, it holds that $t_i \geq f_i$ for all $i \in \{1, \ldots, n\}$.

(iii) For all $1 \leq i \leq n - 1$, $\frac{1}{t_i(t_i + t_{i+1})} \leq (-1)^i \left(x - \frac{s_i}{t_i}\right) \leq \frac{1}{t_i t_{i+1}}$.

(iv) For all $1 \leq i \leq n$, we have that $\left|x - \frac{s_i}{t_i}\right| < \frac{1}{t_i^2}$.

(v) At least one in two adjacent convergents to $x$ satisfies the estimate $\left|\frac{x - s_i}{t_i}\right| < \frac{1}{2t_i^2}$.

(vi) If two integers $s$ and $t$ with $0 < s < t$ satisfy $\left|\frac{x - s}{t}\right| < \frac{1}{2t^2}$, then the rational number $s/t$ is an element of the continued fraction approximation of $x$.

**Proof.** Define the matrix $A_i$ by setting $A_i = E(q_1) \cdots E(q_i)$ for $i = 1, \ldots, n$. By Lemma 3.3, the $i$-th convergent satisfies $s_i/t_i = A_i0$ and hence the second column of $A_i$ consists of the vector $(t_i^T)$ given that the integers $s_i$ and $t_i$ may be assumed to be coprime and strictly positive. It now follows by induction on $i$ that $A_i$ is equal to $\begin{pmatrix} s_{i-1} & s_i \\ t_{i-1} & t_i \end{pmatrix}$ since multiplying form the right by any matrix of the form $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ replaces the first column by the second.

The first statement now follows immediately. Indeed,

$$\begin{pmatrix} s_i & s_{i+1} \\ t_i & t_{i+1} \end{pmatrix} = A_{i+1} = A_i E(q_{i+1}) = \begin{pmatrix} s_i & s_{i-1} + q_{i+1} s_i \\ t_i & t_{i-1} + q_{i+1} t_i \end{pmatrix}$$
for \( i = 1, \ldots, n - 1 \).

The second claim is a direct consequence of the first. Given that \( t_{i-1} \) is strictly positive for \( i \geq 1 \), we have that

\[
t_{i+1} = t_{i-1} + q_{i+1}t_i \geq t_{i-1} + t_i > t_i.
\]

In addition, assuming that both \( f_{i-1} \leq t_{i-1} \) and \( f_i \leq t_i \) implies that

\[
t_{i+1} = t_{i-1} + q_{i+1}t_i \geq f_{i-1} + q_{i+1}f_i \geq f_{i-1} + f_i = f_{i+1}.
\]

As \( t_0 \geq f_0 \) and \( t_1 \geq f_0 \), the claim now follows by induction on \( i \).

In order to establish the third claim, recall that the coefficients \( q_1, \ldots, q_n \) of the canonical continued fraction representation of \( x \) are equal to the quotients obtained through initializing the Euclidean algorithm by \( r_{-1} = t_n \) and \( r_0 = s_n \). Denoting the remainders by \( r_i \), we thus have that

\[
E(q_{i+1}) \frac{r_{i+1}}{r_i} = \frac{r_i}{r_{i+1} + q_{i+1}r_i} = \frac{r_i}{r_{i-1}}
\]

for all \( i = 0, \ldots, n - 1 \), and in consequence \( A_i(r_i/r_{i-1}) = s_{n+1}/t_{n+1} = x \) for all \( i = 1, \ldots, n \). Setting \( x_{i+1} = r_{i+1}/r_i \), we deduce that

\[
(-1)^i \left( x - \frac{s_i}{t_i} \right) = (-1)^i \left( A_{i+1}x_{i+1} - \frac{s_i}{t_i} \right) = (x_{i+1} - s_i/t_i) = \frac{1}{(x_{i+1}t_i + t_{i+1})t_i}.
\]

As \( 0 \leq x_{i+1} \leq 1 \), statement (iii) follows. Taking the absolute value, we obtain

\[
\left| x - \frac{s_i}{t_i} \right| \leq \frac{1}{t_{i+1}t_i} < \frac{1}{t_i^2}
\]

since \( t_{i+1} > t_i \), which proves statement (iv).

For the fifth claim, assume that

\[
\left| x - \frac{s_i}{t_i} \right| \geq \frac{1}{2t_i^2} \quad \text{and} \quad \left| x - \frac{s_{i+1}}{t_{i+1}} \right| \geq \frac{1}{2t_{i+1}^2}
\]

for some \( 1 \leq i \leq n - 1 \). Then we deduce that

\[
\frac{1}{t_it_{i+1}} = \frac{(-1)^{i+1} \det A_{i+1}}{t_{i+1}t_i} = (-1)^{i+1} \left( \frac{s_i}{t_i} - \frac{s_{i+1}}{t_{i+1}} \right) = (-1)^{i+1} \left( x - \frac{s_{i+1}}{t_{i+1}} \right) + (-1)^i \left( x - \frac{s_i}{t_i} \right) = \left| x - \frac{s_{i+1}}{t_{i+1}} \right| + \left| x - \frac{s_i}{t_i} \right| \geq \frac{1}{2t_{i+1}^2} + \frac{1}{2t_i^2},
\]

and hence \((t_{i+1} - t_i)^2 \leq 0\), which is contradictory to the fact that \( t_{i+1} > t_i \) established in (ii).
It remains to show that if a fraction $s/t$ satisfies the conditions specified in (vi) there exists a natural number $i$ with $1 \leq i \leq n$ such that $s/t = A_i0$. We may assume without loss of generality that $s$ and $t$ be coprime given that $1/2(kt)^2 \leq 1/2t^2$ for any integer $k$. We first prove an auxiliary statement:

**Lemma 3.7.** Any matrix $A = \begin{pmatrix} s & u \\ t & v \end{pmatrix}$ in the Euclidean semigroup induces a continuous bijection from the unit interval $[0, 1]$ to the interval with endpoints $u/v$ and $(s + u)/(t + v)$ which agrees with the Möbius transformation induced by $A$ where their domains overlap. Moreover, the preimage under $A$ of any rational point is also a rational point.

**Proof.** Consider the map

$$A : [0, 1] \to \mathbb{R}; \ y \mapsto \frac{sy + u}{ty + v},$$

which is well defined since the denominator is strictly greater than zero for all points in the domain given that $1 \leq t \leq v$. Clearly, this map is continuous and agrees with the Möbius transform induced by $A$. As the derivative of $A$

$$A' y = \frac{\det A}{(ty + v)^2}$$

is nowhere vanishing, the map $A$ is strictly monotone and hence a bijection onto its image, which is equal to the interval with endpoints $A0 = u/v$ and $A1 = (s + u)/(t + v)$ by the intermediate value theorem. To prove the last claim, assume $x$ to be some rational number in the image of $A$, then $A^{-1}(x) = (vx - u)/(-tx + s) \in \mathbb{Q}$ since the entries of $A$ are integers.

If $x$ is equal to the fraction $s/t$, it is the last element of the continued fraction approximation of $x$, that is to say the claim holds. Otherwise, consider the matrix $A' = \begin{pmatrix} s' & s \\ t' & t \end{pmatrix} \in E$ having determinant so that $|x - s/t| = \det A(x - s/t)$, whose existence is shown in the proof of Proposition 3.4. It then follows that

$$0 < \det A'(x - \frac{s}{t}) \leq \frac{1}{2t^2} \leq \frac{1}{t(t' + t)} = \det A' \left( \frac{s' + s}{t' + t} - \frac{s}{t} \right)$$

by assumption on the difference of $x$ and $s/t$, which implies that $x$ lies in the open interval with endpoints $s/t$ and $(s' + s)/(t' + t)$. By the lemma above, there exists a rational number $y \in (0, 1)$ having the property that $A'y = x$. By Proposition 3.4, $y$ possesses a canonical continued fraction representation; or equivalently, $y$ satisfies $y = B0$ for some element $B$ of the Euclidean semigroup whose last factor is not $E(1)$. As $A'B0 = A'y = x$, we conclude that $A'B$ is equal to $A_n$, and in consequence $s/t = A'0$ equals $A_i0$ for some $i = 1, \ldots, n$, that is to say the fraction $s/t$ is a convergent to $x$.  \qed
3.2 The Partially Approximate Common Divisor Problem

In this section we apply the theory on the continued fraction approximation of rational numbers to obtain an algorithm for solving partially approximate common divisor problems by adapting some of the ideas underlying Howgrave-Graham’s ”(continued fraction based) partially approximate common divisor algorithm” ([3] 53). In the notation adopted in this paper, his description of the algorithm is as follows:

The (continued fraction based) partially approximate common divisor algorithm, is defined thus: Its input is two integers $a$, $b$ such that $a < b$. The algorithm should output all integers $d > \sqrt{b}$, such that there exists an $x_0$ with $|x_0| < X(d) = d^2/b$, and $d$ divides both $a + x_0$ and $b$, or report that no such $d$ exists (under the condition on $X(d)$ we are assured that there are only polynomially many solutions for $d$). ([3] 53, emphasis mine)

However, the algorithm described in his paper might miss some approximate common divisors $d$, depending on the parameters $a$ and $b$. Consider for instance $a = 49007$ and $b = 100,000$, then Howgrave-Graham’s algorithm does not find the solution $d = 1000$ and $x_0 = -7$, outputting that $d = 50,000$ with noise $x_0 = 993$ is the only approximate common divisor of $a = 49007$ and $b = 100,000$. This problem is due to the fact that Howgrave-Graham chooses to neglect a factor 2 during the justification of his algorithm without stating the consequences. Fortunately, his ideas can still be applied when retaining this factor, resulting in the slight modification of the algorithm which will be explained in this section. The price of obtaining an algorithm which is guaranteed to work is that the upper bound $X(d)$ on the noise $x_0$ is reduced by half. In brief, the algorithm presented in this section takes two natural numbers $a$ and $b$ - satisfying $b/8 \leq a \leq 7b/8$ - as input and outputs all natural numbers $d$ having the property that there exists an integer $x_0$ bounded by $X(d) = d^2/2b$ such that $d$ is the greatest common divisor of $a + x_0$ and $b$.

Applying the theory presented in Section 3.1, we will show that solving the PACDP with parameters $a$, $b$ and $X(d) = d^2/2b$ can be reduced to testing the convergents to $a/b$ in its continued fraction approximation for certain properties (provided that Conditions 3.1 and 3.3 are satisfied). The following corollary to Proposition 3.6 will allow us to deduce the conditions under which any approximate common divisor of $a$ and $b$ corresponds to some convergent of the rational number $a/b$.

**Corollary 3.8.** Let three integers $a$, $b$ and $x_0$ satisfy $0 < a < b$ as well as $0 < a + x_0 < b$. If there exists a natural number $d$ such that $|x_0| < d^2/2b$ and
d divides both \(a + x_0\) and \(b\), then the ratio \((a + x_0)/b\) appears in the continued fraction approximation of \(a/b\).

**Proof.** Setting

\[
s = \frac{a + x_0}{d} \in \mathbb{Z} \quad \text{and} \quad t = \frac{b}{d} \in \mathbb{Z},
\]

we obtain

\[
\left| \frac{a}{b} - \frac{s}{t} \right| = \left| \frac{a}{b} - \frac{a + x_0}{b} \right| = \frac{|x_0|}{b} < \frac{d^2}{2b^2} = \frac{1}{2t^2}.
\]

As \(0 < s < t\), it follows that \(s/t = (a + x_0)/b\) is a convergent in the continued fraction approximation of \(a/b\) by statement (vi) of Proposition 3.6.

Let us now assume that two integers \(a, b\) and a positive function \(X : \mathbb{N} \rightarrow \mathbb{R}\) satisfy the following two conditions for some natural number \(d\):

\[
X(d) \leq \frac{d^2}{2b} \quad \text{(3.1)}
\]

\[
X(d) \leq a \leq b - X(d) \quad \text{(3.2)}
\]

An immediate consequence of Corollary 3.8 is that the ratio \((a + x_0)/b\) appears in the continued fraction approximation of the rational number \(a/b\) whenever the pair \((d, x_0)\) is a solution to the PACDP with parameters \(a, b\) and \(X\).

Notice that Condition 3.1 entails that a continued fraction based algorithm will only find approximate common divisors which are greater than the square root of \(2b\). Indeed, if an approximate common divisor \(d\) with noise \(x_0\) bounded by \(X(d) \leq d^2/2b\) satisfies \(d < \sqrt{2b}\), \(x_0\) vanishes and hence \(d\) is an *exact* divisor of \(a\) as well as \(b\). Further notice that Condition 3.2 may be replaced by

\[
\frac{b}{8} \leq a \leq \frac{7b}{8} \quad \text{(3.3)}
\]

under the assumption that Condition 3.1 holds, given that any approximate common divisor \(d\) is less than \(b/2\) (being a proper factor of \(b\)). As we may assume without loss of generality that \(0 \leq a < b\), Condition 3.3 is likely to be satisfied automatically. Hence, any solution to the PACDP corresponds to some convergent in the continued fraction approximation of \(a/b\).

Howgrave-Graham "chooses to ignore constant terms like 2" in Condition 3.1 reasoning that he is "primarily concerned with large \(a\) and \(b\" ([3] 55). He thereby implicitly asserts that for large natural numbers \(a < b\), a fraction \(s/t\) - in lowest terms and lying in the open unit interval - is a convergent to the rational \(a/b\) if and only if \(\left| \frac{a}{b} - \frac{s}{t} \right| < \frac{1}{t^2}\). (Notice that one implication
holds by statement (iv) of Proposition 3.6.) However, the following lemma provides counterexamples to this assertion for arbitrarily large denominators $b$. Therefore, when neglecting this factor 2, there is no guarantee that every approximate common divisor of two large integers $a$ and $b$ corresponds to a convergent to $a/b$, which will ultimately result in the fact that the algorithm he proposes misses some approximate common divisors.

**Lemma 3.9.** Letting $s$ and $t$ be coprime integers such that $0 < s < t$, determine $s'$, $s''$, $t'$ and $t''$ by stipulating that $A' = \left( \begin{array}{cc} s' & s \\ t' & t \end{array} \right)$ and $A'' = \left( \begin{array}{cc} s'' & s \\ t'' & t \end{array} \right)$ be the elements of the Euclidean semigroup which map 0 to $s/t$ having determinant 1 and $-1$, respectively. Define $\mathcal{I}(s,t)$ by setting

$$\mathcal{I}(s,t) = \left( \frac{s}{t} - \frac{1}{t^2}, \frac{s'' + s}{t'' + t} \right) \cup \left[ \frac{s' + s}{t' + t}, \frac{1}{t^2} \right].$$

For any rational number $x$ lying in the open unit interval, the following two statements are equivalent:

(i) The ratio $s/t$ does not appear in the continued fraction approximation of $x$ and $|x - \frac{s}{t}| < \frac{1}{t^2}$.

(ii) The rational number $x$ lies in $\mathcal{I}(s,t)$.

Moreover, the (Lebesgue) measure of the union of intervals $\mathcal{I}(s,t)$ is greater than $\frac{1}{(t + t')(t + t'')}$. In particular, $\mathcal{I}(s,t)$ is not empty.

In other words, the above lemma states that whenever a rational number $a/b$ lies in $\mathcal{I}(s,t)$ for some suitable integers $s$ and $t$, it is a counterexample to Howgrave-Graham’s implicit assertion. In addition, contrary to Howgrave-Graham’s intuition, the larger the denominator $b$ is, the greater the number of numerators $a$ such that the fraction $a/b$ lies in $\mathcal{I}(s,t)$.

**Proof.** Letting $x \in (0, 1)$ be a rational number, we first show that the fraction $s/t$ is a convergent to $x$ if and only if $x$ lies in the image of the interval $[0, 1)$ under the M"obius transformation induced by $A'$ or $A''$. To this end, recall that if $[q_1, \ldots, q_n]$ is the canonical continued fraction representation of $x$, the $i$-th convergent to $x$ is given by $E(q_1) \cdots E(q_i)0 = A_i0$. Assuming $s/t$ to be the $i$-th convergent to $x$, that is to say $s/t = A_i0$, we deduce that

$$x = A_i0 = A_iE(q_{i+1})\cdots E(q_n)0 \in A_i[0, 1)$$

since we chose $q_n$ to be strictly greater than 1. Given that $s/t = A_i0$, the matrix $A_i$ is equal to either $A'$ or $A''$ by Proposition 3.4, and thus $x$ lies in the image of $[0, 1)$ under $A'$ or $A''$. 

Conversely, assume that \( x = A'y \) for some \( y \in [0,1) \), the argumentation being analogous for \( x \in A''[0,1) \). In case \( y = 0 \), \( s/t = x = A_00 \). It is therefore sufficient to consider rational numbers \( y \in (0,1) \). By Proposition 3.4, such a number \( y \) possesses a canonical continued fraction representation \([\tilde{q}_1, \ldots, \tilde{q}_m]\); or equivalently, \( y = E(\tilde{q}_1) \cdots E(\tilde{q}_m)0 \).

In consequence,

\[
x = A'y = A'E(\tilde{q}_1) \cdots E(\tilde{q}_m)0,
\]

which implies that \( A' = A_i \) for some \( i < n \) as desired.

According to Lemma 3.7, the image of the half-open interval \([0,1)\) under \( A' \) and \( A'' \) is equal to \( \left[ \frac{s}{t} + \frac{s'}{t'} + \frac{t}{t'} \right] \) and \( \left( \frac{s''}{t''} + \frac{s'}{t'} + \frac{t}{t'} \right) \), respectively. In conjunction with what we have proved so far, this implies that the fraction \( s/t \) does not appear in the continued fraction approximation of the rational number \( x \) if and only if \( x \notin \left( \frac{s''}{t''} + \frac{s'}{t'} + \frac{t}{t'} \right) \). Therefore, it suffices to verify that

\[
\mathcal{I}(s,t) = \left( \frac{s}{t} - \frac{1}{t^2}, \frac{s}{t} + \frac{1}{t^2} \right) \cap \left( \frac{s''}{t''} + \frac{s'}{t'} + \frac{t}{t'} \right) \cap \left( \frac{s''}{t''} + \frac{s'}{t'} + \frac{t}{t'} \right)
\]

to prove the equivalence.

It remains to determine the measure of \( \mathcal{I}(s,t) \): The length of the first interval is given by

\[
\frac{s'' + s}{t'' + t} - \left( \frac{s}{t} - \frac{1}{t^2} \right) = \frac{(s'' + s)t^2 - s(t'' + t)t + (t'' + t)}{(t'' + t)t^2}
\]

\[
= \frac{(s''t - st'')t + t'' + t}{(t'' + t)t^2}
\]

\[
= \frac{\det A''t + t'' + t}{(t'' + t)t^2}
\]

\[
= \frac{t''}{(t'' + t)t^2} > 0,
\]

the determinant of \( A'' \) being equal to \(-1\) by definition. By a similar calculation, the second interval has length \( \frac{t'}{(t' + t)t^2} > 0 \). Using the fact that
18

H. RIEDTMANN

\[ t'' = t - t', \]
we deduce that \( \mathcal{I}(s, t) \) has measure

\[ \frac{t''}{(t'' + t)t^2} + \frac{t'}{(t' + t)t^2} = \frac{(t - t')(t + t') + t'(2t - t')}{(t'' + t)(t' + t)t^2} \]

\[ = \frac{t^2 + 2t't - 2(t')^2}{(t'' + t)(t' + t)t^2} \]

\[ \geq \frac{(t'' + t)(t' + t)t^2}{t^2} \]

\[ = \frac{1}{(t'' + t)(t' + t)} \]

since \( t \) is greater than \( t' \). \( \square \)

We therefore choose to retain the constant factor 2 in Condition 3.1, which will necessitate some further modifications of the algorithm.

To sum up, if the parameters \( a, b \) and \( X \) of a PACDP satisfy Conditions 3.1 and 3.2 for some natural number \( d \) and the pair \( (d, x_0) \) is a solution to the PACDP, then the ratio \((a + x_0)/b\) appears in the continued fraction approximation of \( a/b \). Hence, the first step in solving the PACDP will be to explicitly determine the continued fraction approximation of \( a/b \). In a next step, we check each convergent \( s_i/t_i \) to see whether it corresponds to an approximate common divisor \( d \). A necessary condition is that \( t_i \) be a factor of \( b \). Indeed, \( s_i/t_i \) is equal to \((a + x_0)/b\) for some integer \( x_0 \) if and only if \( s_i b = t_i(a + x_0) \), which implies that \( t_i \) divides \( b \) since \( s_i \) and \( t_i \) may be assumed to be coprime. We thus restrict our attention to convergents to \( x \) having the property that the denominator is a factor of \( b \). For any denominator \( t_i \) which divides \( b \), we set \( d = b/t_i \) and \( x_0 = s_i d - a \), which yields a solution whenever \( x_0 \) is sufficiently small. This procedure results in the following algorithm for solving partially approximate common divisor problems having parameters \( a, b \) and \( X(d) = d^2/2b \):

Algorithm 3.10 (Continued Fraction-Based PACDP).

**Input:** Two natural numbers \( a \) and \( b \) satisfying \( b/8 \leq a \leq 7b/8 \).

**Output:** The set \( S \) of all \((d, x_0)\) in \( \mathbb{N} \times \mathbb{Z} \) so that \( x_0 \) is bounded by \( X(d) = d^2/2b \) and \( d \) is the greatest common divisor of \( a + x_0 \) and \( b \).

**Step 1:** \( S = \emptyset \)

\[ d = \gcd(a, b) \]

\[ S = S \cup \{(d, 0)\} \]

**Step 2:** \{Continued fraction approximation of \( a/b \)\}

\[ A = \emptyset \]

\[ r_{-1} = b; r_0 = a; s_{-1} = 1; s_0 = 0; t_{-1} = 0; t_0 = 1; i = 0 \]

While \( t_i < \sqrt{b/2} \)
\[ r_{i+1} \equiv r_{i-1} \mod r_i \text{ (with the convention that } 0 \leq r_{i+1} < r_i) \]
\[ q_{i+1} = \frac{r_{i-1} - r_{i+1}}{r_i} \]
\[ s_{i+1} = s_{i-1} + q_{i+1}s_i \]
\[ t_{i+1} = t_{i-1} + q_{i+1}t_i \]
\[ i = i + 1 \]
\[ A = A \cup \{(s_i, t_i)\} \]
\[ n = |A| \]

**Step 3:** For \( i = 1, \ldots, n \)

If \( t_i \) divides \( b \)

\[ d = b/t_i \]
\[ x_0 = s_id - a \]

If \( |x_0| < \frac{d^2}{2b} \)

\[ S = S \cup \{(d, x_0)\} \]

**Step 4:** Return \( S \)

**Remark 3.11.** Step 2 of the algorithm only outputs the convergents to \( a/b \) whose denominator is less than the square root of \( b/2 \). This is sufficient, since the square of any approximate common divisor \( d = b/t_i \) must be greater than \( 2b \); otherwise, \( X(d) \) is strictly less than one, which means that \( d \) was outputted in Step 1, being an exact divisor of \( a \) and \( b \).

**Proposition 3.12.** Algorithm 3.10 is correct.

**Proof.** Let \( a \) and \( b \) be natural numbers so that \( a \) lies in the interval \([b/8, 7b/8]\). Clearly, the algorithm only outputs approximate common divisors of \( a \) and \( b \). Conversely, assuming a natural number \( d \) to have the property that there exists an integer \( x_0 \) bounded by \( X(d) = d^2/2b \) so that \( d \) is the greatest common divisor of \( a + x_0 \) and \( b \), we will show that the algorithm outputs \( d \), thus proving that it finds all approximate common divisors. If \( x_0 \) vanishes, \( d \) is the greatest common divisor of \( a \) and \( b \).

By Corollary 3.8, the ratio \((a + x_0)/b\) is a equal to a convergent \( s_i/t_i \) to the rational number \( a/b \) in its continued fraction approximation. Since \( d \) is the greatest common divisor of \( a + x_0 \) and \( b \) and the fraction \( s_i/t_i \) is in lowest terms, we have that \( b = dt_i \) and \( a + x_0 = ds_i \), justifying Step 3. In addition, if \( x_0 \) does not vanish, \( t_i \) is less than the square root of \( b/2 \) as discussed in Remark 3.11.

At the very end of the algorithm, it is verified whether a potential solution \( d \) to the PACDP is an actual solution by testing whether the absolute value of the corresponding noise \( x_0 \) is strictly less than \( X(d) \). There is a good chance that the majority of the integers \( x_0 \) constructed by the algorithm will be sufficiently small to satisfy this bound, for the following two reasons:
By construction, \( x_0 = b s_i / t_i - a \) where \( s_i / t_i \) is the \( i \)-th convergent to \( a / b \). According to estimate (iv) in Proposition 3.6, we therefore have that

\[
|x_0| = \left| a - b \frac{s_i}{t_i} \right| = b \left| \frac{a}{b} - \frac{s_i}{t_i} \right| < \frac{b}{t_i^2} = 2X(b/t_i) = 2X(d),
\]

which entails that \( x_0 \) is at worst double the size of \( X(d) \). Here, the factor 2 which we retain and Howgrave-Grahem chooses to ignore rears its head again. In fact, the algorithm he proposes is more elegant primarily because no such verification is needed.

Furthermore, statement (v) of Proposition 3.6 entails that at least half the convergents to \( a / b \) satisfy the estimate

\[
\left| \frac{a}{b} - \frac{s_i}{t_i} \right| < \frac{1}{2t_i^2},
\]

in which case \( |x_0| \) is strictly less than \( X(d) \).

**Remark 3.13.** There are two options for adapting the above algorithm so that it may be applied to natural numbers \( a \) and \( b \) which do not meet Condition 3.3. Assuming that two natural numbers \( a \) and \( b \) satisfy \( 0 \leq a < b \), the algorithmic procedure outlined above is guaranteed to find all common divisors of \( a \) and \( b \) whose square is less than \( 2b \min\{a, b - a\} \). Indeed, we need Condition 3.3 to insure that the bound function \( X(d) = d^2/2b \) satisfies Condition 3.2 for all approximate common divisors \( d \), thus allowing us to apply Corollary 3.8. By only considering potential approximate common divisors \( d \) which satisfy \( d^2 \leq 2b \min\{a, b - a\} \), we also achieve that

\[
\frac{d^2}{2b} = X(d) \leq a \text{ and } X(d) \leq b - a.
\]

Alternatively, one might decrease the value of the bound function \( X(d) \) by setting \( X(d) = \min\{d^2/2b, a, b - a\} \) to avoid putting a constraint on the size of approximate common divisors.

**Remark 3.14.** Let us note for later reference that Algorithm 3.10 uses

\[
O((\log b)^3)
\]

bit operations.

Indeed, the first two steps of the algorithm both take \( O((\log b)^2) \) bit operations, as the extended Euclidean algorithm takes \( O((\log n)^2) \) bit operations to compute the greatest common divisor of two numbers, each less than \( n \) ([5] 66). For a fixed \( i \), Step 3 also requires \( O((\log b)^2) \) bit operations, which entails that the entire step takes \( O((\log b)^3) \) since \( n \) is less than \( \log_2 b \).

### 3.3 The General Approximate Common Divisor Problem

We will establish an algorithm for solving general approximate common divisor problems, by generalizing the continued fraction-based approach to
partially approximate common divisor problems presented in the previous section. To be more explicit, the algorithm takes two natural numbers \(a\) and \(b\) - satisfying \(\sqrt{b} - 1 \leq 4a \leq 4b - 2(\sqrt{b} - 1)\) - as input and outputs all natural numbers \(d \geq 2\sqrt{b}\) with the property that there exists a pair of integers \((x_0, y_0)\) both bounded by \(X(d) = \min \{d^2/4b, b/2d - 1/4\}\) so that \(d\) is the greatest common divisor of \(a + x_0\) and \(b + y_0\). In his paper "Approximate Integer Common Divisors", Howgrave-Graham also asserts that there exists an analogous algorithm for solving general approximate common divisor problems, but without really justifying his claim ([3] 54).

In analogy to the partially approximate case, we derive a corollary to Proposition 3.6 stating that every solution to the GACDP with parameters \(a, b\) and \(X\) (satisfying Conditions 3.4 and 3.5) corresponds to a convergent to the rational number \(a/b\) in its continued fraction approximation.

**Corollary 3.15.** Let \(a\) and \(b\) be integers and \(X : \mathbb{N} \to \mathbb{R}\) a positive function, satisfying

\[
X(d) \leq \frac{d^2}{4b} \tag{3.4}
\]

for all natural numbers \(d\). If a natural number \(d\) has the property that

\[
X(d) \leq a \leq b - 2X(d) \tag{3.5}
\]

and if there exist integers \(x_0\) and \(y_0\) bounded by \(X(d)\) so that \(d\) divides both \(a + x_0\) and \(b + y_0\), then the ratio \((a + x_0)/(b + y_0)\) appears in the continued fraction approximation of \(a/b\).

**Proof.** Setting

\[
s = \frac{a + x_0}{d} \in \mathbb{Z} \quad \text{and} \quad t = \frac{b + y_0}{d} \in \mathbb{Z},
\]

Condition 3.5 entails that \(0 < s < t\) and \(0 < a < b\); therefore, provided that \(\left|\frac{a}{b} - \frac{s}{t}\right| < \frac{1}{2t^2}\), the fraction \(s/t = (a + x_0)/(b + y_0)\) is a convergent to the rational number \(a/b\), according to statement (vi) of Proposition 3.6. As \(x_0\)
and $y_0$ are strictly less than $d^2/4b$ in absolute value, we have that

$$
\left| \frac{a}{b} - \frac{s}{t} \right| = \left| \frac{a}{b} - \frac{a + x_0}{b + y_0} \right|
= \frac{|ay_0 - bx_0|}{b(b + y_0)}
\leq \frac{a|y_0| + b|x_0|}{b(b + y_0)}
< \frac{(a + b)X(d)}{b(b + y_0)}
\leq \frac{(a + b)d^2}{4b^2(b + y_0)}
= \frac{a + b}{2b^2} \cdot \frac{d^2}{2(b + y_0)}.
$$

In addition, the ratio $\frac{a + b}{2b^2}$ is less than $\frac{1}{b + y_0}$. Indeed,

$$(a + b)(b + y_0) < (2b - 2X(d))(b + X(d)) = 2b^2 - 2X(d)^2 \leq 2b^2.$$

Hence,

$$
\left| \frac{a}{b} - \frac{s}{t} \right| < \frac{d^2}{2(b + y_0)^2} = \frac{1}{2t^2}
$$

as desired.

Having established conditions under which any solution of the GACDP with parameters $a$, $b$ and $X$ corresponds to a convergent to $a/b$ in the preceding corollary, we now impose a further restriction on the bound function $X$ to ensure that any convergent to $a/b$ corresponds to at most one solution. This entails that the number of elements in the continued fraction approximation of $a/b$ provides an upper bound on the number of approximate common divisors of $a$ and $b$.

**Lemma 3.16.** Let $0 < a < b$ be integers and $X : \mathbb{N} \to \mathbb{R}$ a positive function which satisfies

$$
X(d) \leq \min \left\{ \frac{d^2}{4b}, \frac{b}{2d} - \frac{1}{4} \right\}
$$

for all natural numbers $d$. For any natural number $t$ there exists at most one natural number $d$ having the property that

$$
|td - b| < X(d) \text{ and } d \leq 2b.
$$
Notice that if a solution \((d, x_0, y_0)\) to the GACDP with parameters \(a, b\) and \(X\) corresponds to a convergent \(s/t\), i.e. if it satisfies \((a + x_0)/(b + y_0) = s/t\), then \(y_0\) is equal to \(td - b\). Hence, the above lemma does indeed state that at most one solution corresponds to any given convergent to \(a/b\).

Further notice that the assumption that the approximate common divisor \(d\) be less than \(2b\) is not an additional constraint, being a consequence of Condition 3.5. Indeed, suppose that \(d\) divides \(b + y_0\) for some integer \(y_0\) bounded by \(X(d)\), then Condition 3.5 implies that

\[d \leq b + y_0 < b + X(d) \leq b + a \leq 2b.\]

**Proof.** First, we will derive form Condition 3.6 that if a natural number \(d\) satisfies \(d \leq 2b\) as well as \(|td - b| < X(d)|\) for some fixed natural number \(t\), the value of the bound function \(X\) at \(d\) is less than \(t/2\). As \(d\) is bounded by \(2b\), we have that

\[X(d) \leq \frac{d^2}{4b} \leq \frac{d \cdot 2b}{4b} = \frac{d}{2},\]

and in consequence \(|td - b| < X(d) \leq d/2\), which implies that \(|t - b/d| < 1/2\).

Therefore,

\[X(d) \leq \frac{b}{2d} - \frac{1}{4} = \frac{1}{2} \left( \frac{b}{d} - \frac{1}{2} \right) < \frac{t}{2}. \quad (*)\]

Setting \(y(d) = td - b\), let us now assume that two natural numbers \(d\) and \(d'\) are bounded by \(2b\) and satisfy \(|y(d)| < X(d)|\) and \(|y(d')| < X(d')\), respectively. In order to infer that \(d\) is equal to \(d'\), we observe that

\[|y(d) - y(d')| < X(d) + X(d') < t,\]

according to statement \(*\). Since, in addition, \(y(d) \equiv -b \equiv y(d') \mod t\), \(y(d)\) is in fact equal to \(y(d')\) and hence \(d\) and \(d'\) are identical, which proves the claim.

**Remark 3.17.** We note for later reference that the bound function \(X\) of a GACDP with parameters \(a\) and \(b\) which satisfies Condition 3.5 and 3.6 for some natural number \(d\) has the following property: For any natural number \(t\) so that \((td - b)\) is bounded by \(X(d)\), the value of the function \(X\) at \(d\) is less than \(t/2\).

We set \(X(d) = \min\{\frac{d^2}{4b}, \frac{b}{2d} - \frac{1}{4}\}\). Given that \(X(d)\) is an upper bound on the noise \((x_0, y_0)\), we are primarily interested in the case \(X(d) \geq 1\); otherwise, the corresponding approximate common divisor \(d\) of \(a\) and \(b\) is, in fact, an exact common divisor of \(a\) and \(b\), which are comparatively easy to determine. Assuming \(X(d) \geq 1\), we have that \(1 \leq X(d) \leq d^2/4b\) and thus \(d \geq 2\sqrt{b}\). Since \(X(d)\) is also less than \(b/2d - 1/4\), it then follows that \(X(d) \leq \sqrt{b}/4 - 1/4\).
Proposition 3.18. Letting the natural numbers \( a \) and \( b \) satisfy
\[
\frac{\sqrt{b} - 1}{4} \leq a \leq b - \frac{\sqrt{b} - 1}{2},
\] (\( \star \star \))
define the positive function \( X : \mathbb{N} \to \mathbb{R} \) by setting
\[
X(d) = \min \{ \frac{d^2}{4b}, \frac{b}{2d} - \frac{1}{4} \}.
\]
Consider the set \( S(a, b) = \{(d, x_0, y_0) \in \mathbb{N} \times \mathbb{Z}^2 : \gcd(a + x_0, b + y_0) = d \geq 2\sqrt{b} \text{ and } |x_0|, |y_0| < X(d) \} \).
Then the map \( f \) from \( S(a, b) \) to the set of all convergents to the rational number \( a/b \) given by
\[
f(d, x_0, y_0) = \frac{a + x_0}{b + y_0}
\]
is injective. Moreover, the image of \( S(a, b) \) under \( f \) consists of all convergents \( s/t \) having the property that there exists a natural number \( d \geq 2\sqrt{b} \) such that the absolute values of \((dt - b)\) and \((ds - a)\) are strictly less than \( X(d) \).

In other words, the map \( f \) provides a bijection between the set \( S(a, b) \) of all solutions to the GACDP with parameters \( a \), \( b \) and \( X \) such that the approximate common divisor \( d \) is greater than twice the square root of \( b \) and an explicitly given subset of the continued fraction approximation of \( a/b \).

Proof. In order to verify that the map \( f \) is well defined, we show that \( f(d, x_0, y_0) \) is a convergent to \( a/b \) for an arbitrary vector \((d, x_0, y_0) \in S(a, b) \) by applying Corollary 3.15. Indeed, given that the greatest common divisor \( d \) of \( a + x_0 \) and \( b + y_0 \) is greater than twice the square root of \( b \), \( X(d) \) is less than \((\sqrt{b} - 1)/4\), which means that Condition \( \star \star \) is stronger than Condition 3.5. Therefore, the assumptions under which Corollary 3.15 may be applied are all met, which allows us to deduce that \( f(d, x_0, y_0) = (a + x_0)/(b + y_0) \) is indeed a convergent to \( a/b \).

In order to show that \( f \) is injective, assume that two elements in \( S(a, b) \), say \((d, x_0, y_0)\) and \((d', x_0', y_0')\), have the same image under \( f \), that is to say, assume that there exist two coprime integers \( s \) and \( t \) such that
\[
\frac{a + x_0}{b + y_0} = \frac{s}{t} = \frac{a + x_0'}{b + y_0'}.
\]
Since \( d = \gcd(a + x_0, b + y_0) \) and \( d' = \gcd(a + x_0', b + y_0') \), we have that \( y_0 = dt - b \) and \( y_0' = d't - b \). Given that \( y_0 \) and \( y_0' \) are bounded by \( X(d) \) and \( X(d') \), respectively, Lemma 3.16 allows us to infer that \( d \) is equal to \( d' \), which entails that \((d, x_0, y_0)\) and \((d', x_0', y_0')\) are identical.
It remains to determine necessary and sufficient conditions under which a convergent \( s/t \) to \( a/b \) lies in the image of \( f \): Whenever there exists a three-tuple \( (d, x_0, y_0) \in S(a, b) \) so that \( s/t = f(d, x_0, y_0) = \frac{(a + x_0)}{(b + y_0)} \), the natural number \( d = \gcd(a + x_0, b + y_0) \geq 2\sqrt{b} \) satisfies \( ds = a + x_0 \) and \( dt = b + y_0 \) and hence \( |ds - a| = |x_0| \) and \( |dt - b| = |y_0| \) are strictly less than \( X(d) \). Conversely, if there exists a natural number \( d \geq 2\sqrt{b} \) with the property that \( |ds - a| \) and \( |dt - b| \) are strictly less than \( X(d) \), the vector \((d, ds - a, dt - b)\) lies in \( S(a, b) \) and it is the preimage of \( s/t \) under \( f \). \( \square \)

Proposition 3.18 suggests an algorithmic procedure for solving general approximate common divisor problems with parameters \( a, b \) and \( X(d) = \min\{d^2/4b, b/2d - 1/4\} \): After having computed the continued fraction approximation of the rational number \( a/b \), consider each convergent \( s/t \) singly to determine whether it lies in the image of \( S(a, b) \) under \( f \). If this is the case, the projection of \( f^{-1}(s/t) \) onto its first component is an approximate common divisor of \( a \) and \( b \). Formalizing this procedure results in the following algorithm:

**Algorithm 3.19 (Continued Fraction-Based GACDP).**

**Input:** Two natural numbers \( a \) and \( b \) with the property that \( \frac{\sqrt{b} - 1}{4} \leq a \leq b - \frac{\sqrt{b} - 1}{2} \).

**Output:** The set \( S \) of all three tuples \((d, x_0, y_0)\) in \( \mathbb{Z}^3 \) so that \( d \geq 2\sqrt{b} \), \((x_0, y_0)\) are both bounded by \( X(d) = \min\{d^2/4b, b/2d - 1/4\} \) and \( d \) is the greatest common divisor of \( a + x_0 \) and \( b + y_0 \).

**Step 1:** \{Continued fraction approximation of \( a/b \)\}

\[
A = \emptyset \\
r_{-1} = b; r_0 = a; s_{-1} = 1; s_0 = 0; t_{-1} = 0; t_0 = 1; i = 0
\]

**While** \( r_i \neq 0 \)

\[
r_{i+1} \equiv r_{i-1} \mod r_i \quad \text{(with the convention that} \ 0 \leq r_{i+1} < r_i) \\
q_{i+1} = \frac{r_{i-1} - r_{i+1}}{r_i} \\
s_{i+1} = s_{i-1} + q_{i+1}s_i \\
t_{i+1} = t_{i-1} + q_{i+1}t_i \\
i = i + 1 \\
A = A \cup \{(s_i, t_i)\}
\]

\[n = |A|\]

**Step 2:** \( S = \emptyset \)

**For** \( i = 1, \ldots, n \)

\[
y_0 \equiv -b \mod t_i \quad \text{(with the convention that} \ -\frac{t_i}{2} < y_0 < \frac{t_i + 1}{2}) \\
d = (b + y_0)/t_i
\]
\[ x_0 = ds_i - a \]

**If** \( d \geq 2\sqrt{b} \), \(|y_0| < X(d)\) and \(|x_0| < X(d)\)

\[ S = S \cup \{ (d, x_0, y_0) \} \]

**Step 3: Return** \( S \)

**Corollary 3.20.** Algorithm 3.19 is correct.

**Proof.** Letting the fraction \( \frac{s}{t} \) be a convergent to the rational number \( \frac{a}{b} \) in its continued fraction approximation, define the integers \( y_0 \), \( d \) and \( x_0 \) as prescribed in the algorithm. By Proposition 3.18, it is sufficient to show that the three tuple \( (d, x_0, y_0) \) is equal to \( f^{-1}(s/t) \) whenever the convergent \( s/t \) lies in the image of \( S(a, b) \) under \( f \); and that otherwise, \( d \) is strictly less than \( 2\sqrt{b} \) or at least one of \( x_0 \) and \( y_0 \) does not satisfy the bound \( X(d) \).

We first assume the convergent \( s/t \) to lie in the image of \( S(a, b) \) under \( f \), that is to say, we presuppose the existence of an element \( (d', x_0', y_0') \in S(a, b) \) which is mapped to \( s/t \) under \( f \). Using the definitions of \( f \) and \( S(a, b) \), we see that \( y_0' \) satisfies \( y_0' = td' - b \) and \(|y_0'| < X(d')\), which implies that \( y_0' \) is strictly less than \( t/2 \) in absolute value, according to Remark 3.17. Since \( y_0 \) is equivalent to \( y_0' \) modulo \( t \) and is also bounded by \( t/2 \), we deduce that \( y_0 \) is in fact equal to \( y_0' \), and thus \( (d, x_0, y_0) = (d', x_0', y_0') \) as desired.

If, on the other hand, the convergent \( s/t \) is not in the image of \( S(a, b) \) under \( f \), Proposition 3.18 states that the condition that \( d \geq 2\sqrt{b} \) and both \( (dt - b) \) and \( (ds - a) \) are strictly less than \( X(d) \) in absolute value cannot be satisfied for any natural number \( d \). In particular, it is not satisfied for the specific three tuple given in the algorithm. \( \square \)

**Remark 3.21.** Analogously to the partially approximate common divisor algorithm, the Continued Fraction-Based GACDP-Algorithm may be adapted so that it can be applied to natural numbers \( a < b \) which do not satisfy

\[ \frac{\sqrt{b} - 1}{4} \leq a \leq b - \frac{\sqrt{b} - 1}{2} \]

by imposing additional restrictions on either the approximate common divisors or the bound function to achieve that Condition 3.5 is still satisfied.

**Remark 3.22.** Let us note for later reference that Algorithm 3.19 uses

\[ O((\log b)^3) \]

bit operations.

Indeed, being the extended Euclidean algorithm, Step 1 takes \( O((\log b)^2) \) bit operations ([5] 66). For a fixed \( i \), Step 2 requires \( O((\log b)^2) \) bit operations, which implies that the cost of Step 2 is \( O((\log b)^3) \) bit operations, given that \( n \) is less than \( \log_2 b \).
4 The Lattice-Based Algorithm

In this section, we take a fundamentally different approach to approximate common divisor problems. In fact, the lattice-based algorithm even solves a slightly different problem: In the problem considered in Section 3, the upper bound on the noise is a predetermined function of the approximate common divisor, whereas the algorithm presented in Section 4.3 computes a solution to the ACDP for an arbitrary - but fixed - bound on the noise.

In Section 4.1, finding all solutions to this version of the partially approximate common divisor problem is reduced to determining the distinct bounded roots of an explicitly constructed polynomial over the integers, which leads to the next section, in which we present an efficient integer root finding algorithm. Finally, Section 4.3 contains the Lattice-Based PACDP Algorithm as well as a numerical example to which it is applied. In Sections 4.4 and 4.5, the lattice-based approach is extended to two possible generalizations of the PACDP.

4.1 The Partially Approximate Common Divisor Problem

Throughout this section, the letters $a$, $b$, $M$ and $X$ denote the parameters of a partially approximate common divisor problem. To be exact, $a < b$ are the natural numbers whose approximate common divisors are to be determined, where the bounds on the common divisors and their noise are given by $M$ and $X$, respectively. Hence, a solution to the above PACDP consist of a pair of integers $(d, x_0)$ - satisfying $|x_0| \leq X$ as well as $d \geq M$ - with the property that $d$ is the greatest common divisor of $a + x_0$ and $b$. By a slight abuse of notation, we will frequently call the noise $x_0$ a solution to the PACDP as this information suffices to uniquely determine the approximate common divisor $d$.

Let us now turn our attention to solving the problem outlined above. To this end, consider the polynomials $p_a(x) = x + a$ and $p_b(x) = b$ in $\mathbb{Z}[x]$. Any solution to the PACDP is a root $x_0 \in \mathbb{Z}$ of both $p_a(x)$ and $p_b(x)$ modulo some integer $d \geq M$ so that $x_0$ is bounded by $X$. Based on this connection to roots of polynomials, we will first reduce the approximate common divisor problem to the problem of finding bounded integer roots of another polynomial, say $r(x)$ in $\mathbb{Z}[x]$, the construction of which uses public information only. In a second step, we will apply the LLL-Algorithm A.10 to explicitly construct such a polynomial $r(x)$.

For the moment, we consider a fixed integer $d$ which is greater than $M$. Then our goal is to determine the distinct bounded simultaneous roots of $p_a(x)$ and $p_b(x)$ over $\mathbb{Z} / d \mathbb{Z}$. 
Lemma 4.1. For $n \in \mathbb{N}$ and $i \in \{0, \ldots, n\}$, define the polynomial $q_i(x)$ by setting $q_i(x) = p_a(x)^i p_b(x)^{n-i}$ in $\mathbb{Z}[x]$. If an integer $x_0$ is a simultaneous root of $p_a(x)$ and $p_b(x)$ over $\mathbb{Z}/d\mathbb{Z}$, then $x_0$ is a root over $\mathbb{Z}/d^n\mathbb{Z}$ of any polynomial $r(x)$ in $\mathbb{Z}[x]$ of the form

$$r(x) = \sum_{i=0}^{n} h_i(x)q_i(x),$$

where $h_i(x)$ is some polynomial with integer coefficients.

Proof. Let $d$ divide $p_a(x_0)$ as well as $p_b(x_0)$. Then $d^n$ divides $q_i(x_0)$ for all indices $i$, and thus any $\mathbb{Z}$-linear combination of the $q_i(x_0)$.

The advantage of considering polynomials $r(x)$ of the specified form is that, under certain circumstances, one can choose the polynomials $h_i(x)$ so that the absolute value of $r(k)$ is less than $d^n$ for all integers $k$ bounded by $X$, allowing us to apply the following lemma:

Lemma 4.2. Let $K$ be a natural number and $x_0$ an integer bounded by $X$. Let a polynomial $r(x)$ in $\mathbb{Z}[x]$ have the property that $|r(k)|$ is strictly less than $K$ for any integer $k$ of absolute value less than $X$. Then

$$r(x_0) \equiv 0 \mod K \text{ if and only if } r(x_0) = 0 \text{ in } \mathbb{Z}.$$ 

Proof. One implication is immediate. For the other, assume $x_0$ to be a root of $r(x) \mod K$, then $r(x_0)$ is equal to $nK$ for some integer $n$. In consequence, $|nK| = |r(x_0)| < K$, which implies that $n$ vanishes and hence that $x_0$ is a root of $r(x)$ over the integers.

Combing the two preceding lemmas, will allow us to infer conditions on a polynomial $r(x)$ under which any solution to the PACDP is an integer root of $r(x)$. However, in order to state them, we need the following definition:

Definition 4.3. We define the norm of a polynomial $r(x) = r_0 + r_1x + \cdots + r_mx^m$ in $\mathbb{R}[x]$ to be the standard Euclidean norm of its coefficient vector, i.e.

$$||r(x)|| = ||(r_0, r_1, \ldots, r_m)|| = \sqrt{r_0^2 + r_1^2 + \cdots + r_m^2}.$$ 

This brings us to the proposition underlying the lattice-based PACDP Algorithm:

Proposition 4.4. Let a nonzero polynomial $r(x)$ in $\mathbb{Z}[x]$ satisfy the following two conditions for some natural number $n$:

(i) $r(x) = \sum_{i=0}^{n} h_i(x)q_i(x)$ for some $h_i(x) \in \mathbb{Z}[x]$,

(ii) $(\deg r(x) + 1)||r(xX)|| < M^n$. 


Then any solution to the partially approximate common divisor problem with parameters \(a, b, M\) and \(X\) is a root of \(r(x)\) over the integers.

Notice that the conditions stated above are formulated in terms of public information only.

**Proof.** Let a polynomial \(r(x)\) satisfy both conditions and suppose that the pair \((d, x_0)\) forms a solution to the PACDP. As \(p_a(x_0)\) and \(p_b(x_0)\) vanish modulo \(d\), condition (i) entails that \(x_0\) is a root of the polynomial \(r(x)\) over \(\mathbb{Z}/d^m\mathbb{Z}\), according to Lemma 4.1.

We claim that if an integer \(k\) is bounded by \(X\), the value of \(|r(x)|\) at \(k\) is strictly less than \(d^n\), which will allow us to apply Lemma 4.2. Indeed, setting \(m = \deg r(x)\), we infer that

\[
|r(k)| \leq |r_0| + |r_1|X + \cdots + |r_m|X^m \\
\leq (m + 1) \max\{|r_iX^i| : i = 0, \ldots, m\} \\
\leq (m + 1)||r(xX)||,
\]

when viewing \(r(xX)\) as a polynomial with indeterminate \(x\). By condition (ii), we thus have that

\[
|r(k)| \leq (\deg r(x) + 1)||r(xX)|| < M^n \leq d^n
\]
since \(M\) is a lower bound on the approximate common divisors. By Lemma 4.2, we conclude that \(x_0\) is in fact a root of \(r(x)\) in \(\mathbb{Z}\).

In the remainder of this section, we present an explicit construction of a polynomial which satisfies the conditions specified in Proposition 4.4. In fact, provided that

\[
\log_b X < (\log_b M)^2 \tag{4.1}
\]

such a polynomial can be constructed by applying the LLL-Algorithm to a suitable lattice. A short introduction to lattices as well as a description of the LLL-Algorithm is appended at the end of this paper. For the moment, it is sufficient to know that given generators of a lattice \(\Lambda\) in \(\mathbb{Z}^a\) of full rank, the LLL-Algorithm returns a nonzero vector \(v\) in \(\Lambda\) whose norm satisfies

\[
||v|| \leq 2^{(a-1)/2} \sqrt{\theta} (\det \Lambda)^{1/e},
\]

according to Formula A.2.

We will establish that applying the LLL-Algorithm to the lattice \(\Lambda(n,l)\) generated by the row vectors of the \((l+n+1) \times (l+n+1)\)-matrix \(A(n,l)\)
given below yields a vector \( r \) which corresponds to a polynomial \( r(x) \) with the desired properties. Let us define the matrix \( A(n, l) \) to be

\[
\begin{pmatrix}
q_{0,0} & 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\
q_{1,0} & q_{1,1}X & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\
\vdots & & & & & & & & \\
q_{n,0} & q_{n,1}X & q_{n,2}X^2 & \ldots & q_{n,n}X^n & 0 & 0 & \ldots & 0 \\
0 & q_{n,0}X & q_{n,1}X^2 & \ldots & q_{n,n-1}X^n & q_{n,n}X^{n+1} & 0 & \ldots & 0 \\
\vdots & & & & & & & & \\
0 & 0 & \ldots & q_{n,0}X^l & \ldots & \ldots & q_{n,n}X^{n+l}
\end{pmatrix}
\]

where \( q_{i,j} \) is the \( j \)-th coefficient of polynomial \( q_i(x) = (x + a)^i b^{n-i} \) in \( \mathbb{Z}[x] \).

(4.2)

Under the canonical identification of vectors having \( m \) entries and polynomials of degree strictly less than \( m \), the first \( n + 1 \) rows of \( A(n, l) \) correspond to \( q_0(xX) \) up to \( q_n(xX) \), respectively, while the last \( l \) rows are identified with to \((xX)^j q_n(xX)\) for \( j = 1, \ldots, l \).

**Lemma 4.5.** There is a bijection between the set of all \( \mathbb{Z} \)-linear combinations of the rows of \( A(n, l) \) and the set of all \( \mathbb{Z}[xX] \)-linear combinations of the polynomials \( q_0(xX), \ldots, q_n(xX) \) of degree less than \( n + l \).

In other words, the above lemma states that elements of the lattice \( \Lambda(n, l) \) correspond to polynomials of the form \( r(xX) = \sum_{i=0}^{n} h_i(x)q_i(xX) \) where the \( h_i(x) \) in \( \mathbb{Z}[x] \) are polynomials of sufficiently small degree. Moreover, the \( j \)-th coefficient of the corresponding polynomial \( r(x) = \sum_{i=0}^{n} h_i(x)q_i(x) \) may be obtained by dividing the \( j \)-th coefficient of \( r(xX) \) by \( X^j \). More formally, all polynomials in the image of the map \( F : \Lambda(n, l) \rightarrow \mathbb{Z}[x] \) given by

\[
F(v_0, \ldots, v_{n+l}) = \sum_{j=0}^{n+l} v_j X^j
\]

satisfy condition (i) of Proposition 4.4.

**Proof.** As \( \mathbb{Z} \)-linear combinations of the rows of \( A(n, l) \) correspond to \( \mathbb{Z} \)-linear combinations of the polynomials they are identified with, it is equivalent to show that the set of all polynomial linear combinations of \( q_0(xX), \ldots, q_n(xX) \) of degree less than \( n + l \) is equal to to the set of all integer linear combinations of the polynomials \( q_0(xX), \ldots, q_n(xX), (xX)q_0(xX), \ldots, (xX)^l q_n(xX) \).

In a first step, we show that the \( \mathbb{Z}[xX] \)-span of \( \{ q_i(xX) : i = 0, \ldots, n \} \) is a subset of the \( \mathbb{Z} \)-span of \( \{ q_i(xX) : i = 0, \ldots, n \} \cup \{(xX)^j q_n(xX) : j = 1, \ldots, l \} \). To this end, we prove by induction on the degree \( k + m \) that for
any two natural numbers \(k, m\) with \(m \leq n\) the polynomial \((xX)^k q_m(xX)\) lies in the \(\mathbb{Z}\)-span of \(\{q_0(xX), \ldots, q_n(xX), (xX)^m q_0(xX), \ldots, (xX)^k q_m(xX)\}\).

The crucial observation is that one can write \((xX)^k q_m(xX)\) where \(k + m \leq n\) as an integer linear combination of the \(q_i(xX)\). Since \(q_i(xX) = p_a(xX)^i p_b(xX)^{n-i} = (xX + a)^i b^{n-i}\), we have

\[
\sum_{i=0}^{k} \binom{k}{i} b^i (-a)^{k-i} q_{m+i}(xX)
\]

\[
= \sum_{i=0}^{k} \binom{k}{i} b^i (-a)^{k-i} (xX + a)^{m+i} b^{n-(m+i)}
\]

\[
= (xX + a)^m b^{n-m} \sum_{i=0}^{k} \binom{k}{i} (-a)^{k-i} (xX + a)^i
\]

\[
= q_m(xX)(xX)^k,
\]

which proves the statement in the base case \(k + m \leq n\).

For the induction step, assume that the statement holds for a polynomial of the form \((xX)^k q_m(xX)\). Then \((xX)^{k+1} q_m(xX)\) can be expressed as a \(\mathbb{Z}\)-linear combination of \((xX)^m q_0(xX), \ldots, (xX)^m q_{n-1}(xX), (xX)^m q_n(xX), \ldots, (xX)^{k+1} q_m(xX)\), which proves the claim since the first \(n\) polynomials in this enumeration are \(\mathbb{Z}\)-linear combinations of the polynomials \(q_0(xX), \ldots, q_n(xX)\) by the base case.

In particular, any \(\mathbb{Z}[xX]\)-linear combination \(r(xX)\) of \(q_0(xX), \ldots, q_n(xX)\) of degree at most \(n + l\) lies in the \(\mathbb{Z}\)-span of \(\{q_i(xX) : i = 0, \ldots, n\} \cup \{(xX)^j q_n(xX) : j = 1, \ldots, l\}\). Since the degrees of the polynomials in this set are pairwise distinct, it is linearly independent over \(\mathbb{Z}\), and in consequence \(r(xX)\) is equal to exactly one integer linear combination of the polynomials \(q_0(xX), \ldots, q_n(xX), (xX)^m q_0(xX), \ldots, (xX)^m q_n(xX)\). This proves the non-trivial inclusion and hence the lemma.

Suppose that a polynomial \(r(x)\) in \(\mathbb{Z}[x]\) is the image of a vector \(v\) in \(\Lambda(n, l)\) under \(F\), then the norm of \(v\) is equal to the norm of \(r(xX)\) viewed as an element of \(\mathbb{Z}[x]\). In consequence, any nonzero vector \(v\) in the lattice \(\Lambda(n, l)\) whose norm is strictly less than \(M^n/(n + l + 1)\) corresponds to a polynomial which satisfies the conditions of Proposition 4.4.

We therefore determine natural numbers \(n\) and \(l\) so that the LLL-Algorithm is guaranteed to compute a sufficiently short vector in \(\Lambda(n, l)\) to satisfy condition (ii): The lattice \(\Lambda(n, l)\) is of rank \(l + n + 1\) and its determinant is given by

\[
\det \Lambda(n, l) = |\det A(n, l)| = \left| \prod_{i=0}^{n} q_i x^i \prod_{j=1}^{l} q_{n+j} x^{n+j} \right| = \prod_{i=1}^{n} b_i^{n+l} X^j.
\]
By Formula A.2, the LLL-Algorithm with input $\Lambda(n, l)$ outputs a vector $v$ in $\Lambda(n, l)$ whose norm is bounded by

$$||v|| \leq 2^{(l+n)/2} \sqrt{l+n+1} \left( \prod_{i=1}^{n} b^i \prod_{j=1}^{n+l} X^j \right)^{1/(l+n+1)},$$

which means that in order ascertain that the polynomial corresponding to $v$ satisfies condition (ii) of Proposition 4.4, $n$ and $l$ should be chosen so that

$$2^{(l+n)/2} (l+n+1)^{3/2} \left( \prod_{i=1}^{n} b^i \prod_{j=1}^{n+l} X^j \right)^{1/(l+n+1)} < M^n. \quad (4.3)$$

Set $\xi = \log_b X$ and $\mu = \log_b M$. Since in any reasonable setting for the PACDP $b > M > 1$ and $b > X > 1$, we may assume that $\xi, \mu \in (0, 1)$. To further simplify the equation above, we neglect the term $2^{(l+n)/2} (l+n+1)^{3/2}$, which will be comparatively small if $M, X$ and $b$ are large. Taking the logarithm, the problem reduces to finding a pair $(n, l)$ of natural numbers such that

$$\frac{n(n+1) + \xi(l+n+1)(l+n)}{2(l+n+1)} < \mu n;$$

or equivalently,

$$\xi < \frac{2\mu n(l+n+1) - n(n+1)}{(l+n+1)(l+n)}.$$

By setting $k = n + l$, this is equivalent to finding a pair $(n, k)$ of natural numbers which satisfies $k \geq n$ and

$$\xi < \frac{2\mu n(k+1) - n(n+1)}{(k+1)k}.$$

Furthermore, $k$ should be chosen to be as small as possible since the cost of the LLL-Algorithm primarily depends on the rank of the lattice $\Lambda(n, l)$, which is $n + l + 1 = k + 1$.

We consider the function

$$f_{\mu, k}(n) = \frac{2\mu n(k+1) - n(n+1)}{(k+1)k}$$

with parameters $\mu$ and $k$. A short calculation shows that $f_{\mu, k}$ attains its maximal value at $n = \mu k + \mu - \frac{1}{2}$. Given that $|\mu| < 1$ and that the value of $n$ will have to be rounded to the nearest natural number, we set $n = \mu k$ for ease of computation. The value of the function $f_{\mu, k}$ at $\mu k$ is $f_{\mu, k}(\mu k) = \mu^2 - \frac{\mu(1-\mu)}{k+1}$. Hence, if there exists $\varepsilon > 0$ such that $\xi = \mu^2 - \varepsilon$, it is possible to find natural
numbers $n$ and $k$ with the properties that $n \leq k$ and $\xi < f_{\mu,k}(n)$; namely $k = \lfloor \mu(1 - \mu) \rfloor$ and $n = \lfloor \mu k \rfloor$ (which is less than $k$ since $|\mu| < 1$).

In summary, provided that Condition 4.1 is satisfied, we set

$$n = \lfloor \log_b M \left( \frac{\log_b M(1 - \log_b M)}{(\log_b M)^2 - \log_b X} \right) \rfloor$$

and

$$l = \lfloor \log_b M \left( \frac{\log_b M(1 - \log_b M)}{(\log_b M)^2 - \log_b X} \right) - n \rfloor.$$

(4.4)

Then applying the LLL-Algorithm to the lattice $\Lambda(n, l)$ should yield a vector $v$ in $\Lambda(n, l)$ corresponding to a polynomial $r(x)$ which satisfies all conditions of Proposition 4.4. However, owing to numerous simplifications, it is theoretically possible that the norm of the vector $v$ does not satisfy Inequality 4.3, in which case the polynomial $r(x)$ does not satisfy condition (ii). Hence, there is a vanishingly small probability that the Lattice-Based PACDP Algorithm returns "failure" instead of all solutions to the PACDP.

Once a polynomial $r(x)$ in $\mathbb{Z}[x]$ with the desired properties is found, Proposition 4.4 states that all solutions to the PACDP are roots of $r(x)$. Given that the degree of $r(x)$ is at most $n + l$, it has at most $n + l$ roots, which means that the number of solutions to the PACDP is also bounded by $n + l$. In sum, we have reduced the partially approximate common divisor problem to the problem of finding the distinct bounded integer roots of a polynomial over the integers. More formally, the LLL-Algorithm returns a polynomial $r(x)$ in $\mathbb{Z}[x]$ so that any solution to the PACDP is an element of the set

$$\{x_0 \in \mathbb{Z} : r(x_0) = 0 \text{ and } |x_0| \leq X\}.$$

Before presenting an algorithm that efficiently computes the distinct bounded roots of any polynomial over the integers in Section 4.2, we turn our attention to the cost of explicitly computing a suitable polynomial $r(x)$. We have seen that the LLL-Algorithm with input $\Lambda(n, l)$ (where $n$ and $l$ are given by Formula 4.4) returns a vector $v$ that corresponds to a polynomial with the desired properties. According to Proposition A.19, given a basis $v_1, \ldots, v_\varrho$ of a lattice $\Lambda$ in $\mathbb{Z}^\varrho$, the LLL-Algorithm uses $O(\varrho^6(\log C)^3)$ bit operations where $C$ bounds the norm of the basis vectors form above.

The following Lemma provides such an upper bound for the lattice $\Lambda(n, l)$.

**Lemma 4.6.** Let $v_i$ denote the $i$-th row vector of the matrix $\Lambda(n, l)$. Then $C = \sqrt{n + 1} X^{\frac{6}{\log b}}$ is greater than $\max\{||v_i|| : i = 1, \ldots, n + l + 1\}$. In particular, $(n + 1) b^{n+1}$ is greater than $\max\{||v_i|| : i = 1, \ldots, n + l + 1\}$.

**Proof.** By definition of the matrix $\Lambda(n, l)$, the vector $v_{i+1}$ corresponds to the
polynomial \( q_i(xX) \) whenever \( 0 \leq i \leq n \), in which case
\[
||v_{i+1}|| = ||q_i(xX)|| = ||(xX + a)^i b^{n-i}|| = ||b^{n-i} \sum_{j=0}^{i} (xX)^j a^{i-j}||
\]
\[
= b^{n-i} \left( \sum_{j=0}^{i} (X^j a^{i-j})^2 \right)^{1/2} \leq b^{n-i} \left( \sum_{j=0}^{i} b^{2j} \right)^{1/2}
\]
\[
= \sqrt{i+1} b^n \leq \sqrt{n+1} b^n \leq C
\]
where we used that both \( X \) and \( a \) are less than \( b \). On the other hand, if \( 1 \leq i \leq l \), the vector \( v_{i+n+1} \) is identified with the polynomial \( X^i q_n(xX) \), form which we infer that
\[
||v_{i+l+1}|| = ||X^i q_n(xX)|| = X^i \sqrt{n+1} b^n \leq X^l \sqrt{n+1} b^n = C.
\]
The second claim now follows immediately as \((n+1)b^{n+l}\) is greater than \(C\).

Therefore, we can compute a polynomial \( r(x) \) that satisfies the conditions outlined in Proposition 4.4 using
\[
O((n + l)^9 (\log b)^3) \tag{4.5}
\]
bit operations where \( n \) and \( l \) are given by Formula 4.4.

4.2 Finding Roots of a Polynomial over the Integers

In the previous section, we have reduced the partially approximate common divisor problem to the problem of finding the bounded integer roots of a polynomial with integer coefficients, raising the question of how to find bounded roots over the integers efficiently. In consequence, this section is dedicated to explaining a root finding algorithm based on Chapters 14 and 15 of [6]. Concretely, given a polynomial \( r(x) \) with integer coefficients and a natural number \( X \), Algorithm 4.18 outputs the set of all distinct integer roots of \( r(x) \) which are less than \( X \) in absolute value.

Algorithm 4.18 consists of three main conceptual steps. In a first step, the distinct roots of the polynomial \( r(x) \) modulo some suitable, small prime number \( p \) are determined, giving rise to a factorization of the form \( r(x) \equiv r_0(x)(x - x_1) \cdots (x - x_d) \mod p \) where \( x_1, \ldots, x_d \) are the roots of \( r(x) \) over \( \mathbb{Z}/p\mathbb{Z} \). In a next step, this factorization is lifted to a factorization of the same type modulo some power of \( p \) using a procedure called Hensel lifting, which in turn allows us to deduce the set of all roots of \( r(x) \) modulo, say \( p^m \). Assuming the \( m \)-th power of \( p \) to be greater than twice the bound \( X \), the roots of \( r(x) \) over the integers that are bounded by \( X \) can be viewed as
a subset of the roots modulo $p^m$, by choosing representatives with minimal absolute value. This observation leads to the third and last step: For each root over $\mathbb{Z}/p^m\mathbb{Z}$, the algorithm verifies whether its representative is a root over the integers that is bounded by $X$.

4.2.1 Finding Roots Modulo a Prime

Let us first focus on finding the distinct roots of a polynomial over $\mathbb{Z}/p\mathbb{Z}$ for some fixed odd prime number $p$. Algorithm 4.13 - the root finding algorithm over $\mathbb{Z}/p\mathbb{Z}$ used in this paper - is a specialization of the factoring algorithm discussed in Chapter 14 of [6], which consists of two stages, namely distinct-degree factorization and equal-degree factorization. In our case, distinct-degree factorization splits off the product of all distinct linear factors occurring in the polynomial, while equal-degree factorization gives the factorization of any product of distinct linear factors.

Distinct-degree factorization is based on the following application of Fermat’s Little Theorem:

**Lemma 4.7.** For any prime number $p$, the following equation holds:

$$x^p - x = \prod_{a \in \mathbb{Z}/p\mathbb{Z}} (x - a) \text{ in } (\mathbb{Z}/p\mathbb{Z})[x].$$

**Proof.** Fermat’s Little Theorem states that for any prime $p$ and integer $a$, $a^p - a \equiv 0 \pmod{p}$ ([6] 75). Therefore, every element of $\mathbb{Z}/p\mathbb{Z}$ is a root of the polynomial $x^p - x$. Using the fact that $\mathbb{Z}/p\mathbb{Z}$ is a field with cardinality $p$, the statement now follows by the observation that the number of roots of $x^p - x$ coincides with its degree. \qed

By Lemma 4.7, the greatest common divisor of $x^p - x$ and an arbitrary polynomial $r(x)$ in $\mathbb{Z}/p\mathbb{Z}$ is the product of the distinct linear factors of $r(x)$, that is to say, its distinct-degree factorization. Distinct-degree factorization reduces the problem of finding the roots of an arbitrary polynomial to finding the factorization into irreducibles of a squarefree product of linear factors, the process of equal-degree factorization. This is accomplished by iterating the process of splitting products of this type into two proper factors, using the following algorithm:

**Algorithm 4.8 (Equal-Degree Splitting).**

**Input:** A monic polynomial $r(x) \in (\mathbb{Z}/p\mathbb{Z})[x]$ where $p$ is an odd prime so that $r(x)$ is a product of several distinct linear factors.

**Output:** A proper monic factor $f(x)$ of $r(x)$ or "failure".

**Step 1:** Randomly choose $a(x) \in (\mathbb{Z}/p\mathbb{Z})[x]$ with $0 < \deg a(x) < \deg r(x)$

If $\gcd(a(x), r(x)) \neq 1$

Return $\gcd(a(x), r(x))$
Step 2: \(b(x) = a(x)^{(p-1)/2} \mod r(x)\)

- If \(\gcd(b(x) - 1, r(x)) \notin \{1, r(x)\}\)
  - Return \(\gcd(b(x) - 1, r(x))\)
- Else Return "failure"

Provided that it does not output "failure", the above algorithm clearly outputs a proper factor of \(r(x)\), which proves its correctness. However, informally speaking, the usefulness of Algorithm 4.8 is inversely proportional to the probability that it outputs "failure". Therefore, we need to show that the probability of "failure" is low, to proof of which is based on a combination of number theoretic concepts.

Let \(p\) be an odd prime and \(r(x)\) a polynomial in \((\mathbb{Z}/p\mathbb{Z})[x]\) of the form \(r(x) = (x - x_1) \cdots (x - x_d)\) where the \(x_i \in \mathbb{Z}/p\mathbb{Z}\) are pairwise distinct.

Consider the morphism of rings given by

\[
\chi : (\mathbb{Z}/p\mathbb{Z})[x]/r(x) \to \mathbb{Z}/p\mathbb{Z} \times \cdots \times \mathbb{Z}/p\mathbb{Z};
\]

\[
a(x) \mapsto (a(x_1), \ldots, a(x_d)).
\]

According to the Chinese Remainder Theorem ([6] 103), \(\chi\) is an isomorphism. In particular, it induces an isomorphism on the respective groups of unity, which we also denote by \(\chi\).

In general, any unit \(a\) in \(\mathbb{Z}/p\mathbb{Z}\) satisfies \(a^{p-1} = 1\) by Fermat’s Little Theorem ([6] 75), and hence is also satisfies \(a^{(p-1)/2} = \pm 1\). In consequence, if a polynomial \(a(x) \in (\mathbb{Z}/p\mathbb{Z})[x]\) is invertible modulo \(r(x)\), we have that

\[
\chi(a(x)^{(p-1)/2}) = (a(x_1)^{(p-1)/2}, \ldots, a(x_d)^{(p-1)/2}) \in \{-1, 1\}^d.
\]

Setting \(b(x) = a(x)^{(p-1)/2}\), we deduce that

\[
\gcd(b(x) - 1, r(x)) = \prod_{i:b(x_i)=1} (x - x_i)
\]

is a proper factor of \(r(x)\) whenever \(\chi(b(x)) \notin \{(-1, \ldots, -1), (1, \ldots, 1)\}\). Therefore, for a given invertible polynomial \(a(x)\), the probability that Algorithm 4.8 outputs "failure" is equal to the probability that \(\chi(a(x)^{(p-1)/2}) = \pm(1, \ldots, 1)\). In order to make use of this equality we introduce a new concept.

**Definition 4.9.** A quadratic residue modulo a prime \(p\) is a square in the group of units of \(\mathbb{Z}/p\mathbb{Z}\).

The only quadratic residue modulo 3, for instance, is (the equivalence class of) 1 since no element \(a\) of \(\mathbb{Z}/3\mathbb{Z}\) satisfies \(a^2 = -1\).

The following lemma provides the facts about quadratic residues which are relevant to equal-degree factorization:
Lemma 4.10. Let \( p \) be an odd prime number. The set of all quadratic residues modulo \( p \) is a subgroup of the group of units of \( \mathbb{Z}/p\mathbb{Z} \) of cardinality \( (p-1)/2 \), characterized by the property that \( a \in \mathbb{Z}/p\mathbb{Z} \) is a quadratic residue modulo \( p \) if and only if \( a^{(p-1)/2} = 1 \).

Proof. Consider the morphism \( \sigma : (\mathbb{Z}/p\mathbb{Z})^* \to (\mathbb{Z}/p\mathbb{Z})^* ; a \mapsto a^2 \), which maps every element to its square. By definition, the image of \( \sigma \) is the set of all quadratic residues modulo \( p \), which in consequence is a subgroup of \( (\mathbb{Z}/p\mathbb{Z})^* \).

Given that \( p \) is an odd prime, the kernel of \( \sigma \) consists of the two distinct elements 1 and \(-1\), entailing that

\[
|\text{im}\sigma| = \frac{|(\mathbb{Z}/p\mathbb{Z})^*|}{|\ker\sigma|} = \frac{p-1}{2}.
\]

It remains to show that a unit \( a \) lies in the image of \( \sigma \) if and only if \( a^{(p-1)/2} = 1 \). One implication is a consequence of Fermat’s Little Theorem, stating that the \((p-1)\)-th power of any unit in \( \mathbb{Z}/p\mathbb{Z} \) equals 1 ([6] 75). Indeed, if \( a \) is a quadratic residue, there exists a unit \( b \) so that \( b^2 = a \), and hence

\[
a^{(p-1)/2} = (b^2)^{(p-1)/2} = b^{p-1} = 1.
\]

The other implication now follows by comparing the cardinalities of the two sets: As \( \mathbb{Z}/p\mathbb{Z} \) is a field the polynomial \( x^{(p-1)/2} - 1 \) has at most \( (p-1)/2 \) roots. Therefore, the cardinality of the set of units whose \((p-1)/2\)-power equals 1 is less than \( (p-1)/2 \), allowing us to conclude that it coincides with the image of \( \sigma \).

Corollary 4.11. The probability that Algorithm 4.8 outputs "failure" is less than \( 2^{1-d} \) where \( d \) is the number of linear factors of \( r(x) \).

Proof. Let a nonconstant polynomial \( a(x) \in (\mathbb{Z}/p\mathbb{Z})[x] \) be the standard representative of an equivalence class modulo \( r(x) \). If \( a(x) \) and \( r(x) \) are not coprime their greatest common divisor is a proper factor of \( r(x) \); otherwise, \( a(x) \) is invertible modulo \( r(x) \), in which case it suffices to prove that the probability that \( a(x)^{(p-1)/2} \) is mapped to \((-1,\ldots,-1) \) or \((1,\ldots,1) \) is \( 2^{1-d} \).

By Lemma 4.10, each entry of the vector \( \chi(a(x)^{(p-1)/2}) \) is 1 or \(-1\) with equal probability. Hence, the probability that all its entries are either 1 or \(-1\) is \( 2(1/2)^d \) (\( d \) being the number of its entries).

Having established that the probability that Algorithm 4.8 returns "failure" instead of a proper factor is less than \( 1/2 \), we iterate its procedure to obtain an equal-degree factorization algorithm.

Algorithm 4.12 (Equal-Degree Factorization).

**Input:** A monic polynomial \( r(x) \in (\mathbb{Z}/p\mathbb{Z})[x] \) where \( p \) is an odd prime so that \( r(x) \) is a (nonempty) product of distinct linear factors.

**Output:** The monic linear factors of \( r(x) \) in \( \mathbb{Z}/p\mathbb{Z} \).
Step 1: If \( \deg(r(x)) = 1 \)
\[\text{Return } r(x)\]

Step 2: \( f(x) = \text{"failure"} \)
\[\text{While } f(x) = \text{"failure"} \]
\[\text{Call the Equal-Degree Splitting Algorithm 4.8}\]

Step 3: Call this algorithm recursively with input \( f(x) \) and \( r(x)/f(x) \)

Composing distinct-degree and equal-degree factorization, results in a list of all distinct monic linear factors of a given polynomial \( r(x) \), and thus in a list of all distinct roots of \( r(x) \), given rise to the Root Finding over Finite Fields Algorithm, which is a misnomer since we have only derived the algorithm for finite fields whose cardinality is some odd prime; however, the same argumentation may be applied to general finite fields with odd characteristic.

Algorithm 4.13 (Root Finding over Finite Fields).
Input: A nonconstant polynomial \( r(x) \in \mathbb{Z}/p\mathbb{Z}[x] \) where \( p \) is an odd prime.
Output: The distinct roots of \( r(x) \) in \( \mathbb{Z}/p\mathbb{Z} \).

Step 1: \{Distinct-degree factorization\}
\[r(x) = \gcd(r(x), x^p - x)\]
\[d = \deg(r(x))\]
\[\text{If } d = 0 \]
\[\text{Return } \emptyset\]

Step 2: \{Equal-degree factorization\}
\[\text{Call the Equal-Degree Factorization Algorithm 4.12}\]
\[x - x_1, \ldots, x - x_d\]

Step 3: Return \( \{x_1, \ldots, x_d\} \)

4.2.2 Hensel Lifting
The next step towards an algorithm which outputs the bounded roots of any polynomial over the integers consists of lifting its linear factors modulo a prime \( p \) to factors modulo some power of \( p \). This is achieved by the so-called Hensel Lifting Algorithm, which will be discussed in this section.

The basic building block of the Hensel Lifting Algorithm is the Hensel Step Algorithm, which lifts a product of two polynomials in \( (\mathbb{Z}/m\mathbb{Z})[x] \) - satisfying a myriad of additional conditions - to a product in \( (\mathbb{Z}/m^2\mathbb{Z})[x] \) that satisfies analogous conditions. The fact that the Hensel Step Algorithm retains its input conditions will allow us to apply it repeatedly thereby lifting
monic, there exist unique polynomials \( c \) and \( q \) of degree \( q \) and degree \( p \) such that \( r(x) \equiv p(x)q(x) \mod n \) and \( s(x)p(x) + t(x)q(x) \equiv 1 \mod n \).

n and the leading coefficient of \( r(x) \) are coprime, \( q(x) \) is monic, the degree of \( r(x) \) is the sum of the degrees of \( p(x) \) and \( q(x) \), the degree \( s(x) < \deg q(x) \) and \( \deg t(x) < \deg p(x) \).

Output: Polynomials \( p^*(x), q^*(x), s^*(x), t^*(x) \) in \( \mathbb{Z}[x] \) such that

\[
 r(x) \equiv p^*(x)q^*(x) \mod n^2 \quad \text{and} \quad s^*(x)p^*(x) + t^*(x)q^*(x) \equiv 1 \mod n^2,
\]

\( q^*(x) \) is monic, \( p^*(x) \equiv p(x) \mod n \), \( q^*(x) \equiv q(x) \mod n \), \( s^*(x) \equiv s(x) \mod n \), \( t^*(x) \equiv t(x) \mod n \), \( \deg p^*(x) = \deg p(x) \), \( \deg q^*(x) = \deg q(x) \), \( \deg s^*(x) < \deg q^*(x) \) and \( \deg t^*(x) < \deg p^*(x) \).

Step 1: \( c(x) \equiv r(x) - p(x)q(x) \mod n^2 \\
\quad d(x) \equiv s(x)c(x) \mod q(x) \mod n^2 \\
\quad e(x) \equiv (s(x)c(x) - d(x))/q(x) \mod n^2 \\
\quad p^*(x) \equiv p(x) + t(x)c(x) + e(x)p(x) \mod n^2 \\
\quad q^*(x) \equiv q(x) + d(x) \mod n^2
\]

Step 2: \( f(x) \equiv s(x)p^*(x) + t(x)q^*(x) - 1 \mod n^2 \\
\quad g(x) \equiv s(x)f(x) \mod q^*(x) \mod n^2 \\
\quad h(x) \equiv (s(x)f(x) - g(x))/q^*(x) \mod n^2 \\
\quad s^*(x) \equiv s(x) - g(x) \mod n^2 \\
\quad t^*(x) \equiv t(x) - t(x)f(x) - h(x)p^*(x) \mod n^2
\]

Step 3: Return \( p^*(x), q^*(x), s^*(x), t^*(x) \)

Lemma 4.15. Algorithm 4.14 is correct.

Proof. The justification of the algorithm is based on the following observation: Given two polynomials \( a(x) \) and \( b(x) \) over the integers so that \( b(x) \) is monic, there exist unique polynomials \( c(x) \) and \( d(x) \) in \( \mathbb{Z}[x] \) with the properties that \( a(x) = b(x)c(x) + d(x) \) and \( \deg d(x) < \deg b(x) \). Furthermore, if \( a(x) \) vanishes modulo some integer \( n \), then so do \( c(x) \) and \( d(x) \).

According to the above observation, \( d(x) \equiv 0 \equiv e(x) \mod n \) given that \( s(x)c(x) \equiv q(x)e(x) + d(x) \mod n^2 \) and that \( s(x)c(x) \equiv 0 \mod n \), by the
definition of $c(x)$. We may therefore infer that

$$r(x) - p^*(x)q^*(x) \equiv r(x) - [p(x) + t(x)c(x) + e(x)p(x)] [q(x) + d(x)] \equiv r(x) - p(x)q(x) - p(x)d(x) - [t(x)c(x) + e(x)p(x)] q(x) =$$

$$= [t(x)c(x) + e(x)p(x)]d(x) \equiv r(x) - p(x)q(x) - p(x)d(x) - [t(x)c(x) + e(x)p(x)] q(x),$$

and by substituting $s(x)c(x) - e(x)q(x)$ for $d(x)$, we obtain

$$\equiv r(x) - p(x)q(x) - c(x)[s(x)p(x) + t(x)q(x)]$$

$$\equiv c(x)[1 - (s(x)p(x) + t(x)q(x))] \equiv 0 \mod n^2$$

since $s(x)p(x) + t(x)q(x) \equiv 1 \mod n$.

Another consequence of the fact that $c(x), d(x)$ and $e(x)$ vanish modulo $n$ is that $p^*(x)$ and $q^*(x)$ are congruent to $p(x)$ and $q(x)$ modulo $n$, respectively. Moreover, as the degree of $d(x)$ is strictly less than that of $q(x)$, $q^*(x)$ is of the same degree as $q(x)$ and is also monic. The degrees of $p^*(x)$ and $p(x)$ coincide since the equivalence $r(x) \equiv p^*(x)q^*(x) \mod n^2$ together with the fact that $q^*(x)$ is monic implies that $\deg p^*(x) = \deg r(x) = \deg q^*(x) = \deg r(x) - \deg q^*(x) = \deg r(x) - \deg q(x) = \deg p(x)$.

A similar argument proves the claims about $s^*(x)$ and $t^*(x)$, which we include for the sake of completeness. Recalling that $p^*(x)$ is congruent to $p(x)$ modulo $n$ and $q^*(x)$ to $q(x)$, we deduce that $f(x)$ vanishes modulo $n$. Therefore, we have that

$$s^*(x)p^*(x) + t^*(x)q^*(x) - 1 \equiv [s(x) - g(x)]p^*(x) + [t(x) - t(x)f(x) - h(x)p^*(x)]q^*(x) - 1 \equiv s(x)p^*(x) + t(x)q^*(x) - 1 - g(x)p^*(x) - t(x)f(x)q^*(x) - h(x)p^*(x)q^*(x),$$

replacing $s(x)p^*(x) + t(x)q^*(x) - 1$ by $f(x)$ and $g(x)$ by $s(x)f(x) - q^*(x)h(x)$, we obtain

$$\equiv f(x) - s(x)f(x)p^*(x) - t(x)f(x)q^*(x)$$

$$\equiv f(x)[1 - (s(x)p(x) - t(x)q^*(x))] \equiv -f^2(x) \equiv 0 \mod n^2.$$

Moreover, the fact that $f(x) \equiv 0 \mod n$ further implies that $g(x) \equiv 0 \equiv h(x) \mod n$, according to the above observation, and hence that $s^*(x) \equiv s(x) \mod n$ and $t^*(x) \equiv t(x) \mod n$. It remains to show that the degrees of $s^*(x)$ and $t^*(x)$ are strictly less than the degrees of $q^*(x)$ and $p^*(x)$, respectively. Since $s^*(x)$ is the sum of two polynomials whose degrees are strictly less than the degree of $q^*(x)$, so is its degree. Together with the fact that $q^*(x)$ is monic and that $s^*(x)p^*(x) + t^*(x)q^*(x)$ is congruent to 1 modulo $n^2$, this implies that the degree of $t^*(x)$ is strictly less than the degree of $p^*(x)$. \(\square\)
Having established the correctness of the Hensel Step Algorithm, we now apply it to lift factorizations modulo a prime number $p$. Suppose that a monic polynomial $r(x)$ over the integers is the product of two coprime monic factors modulo $p$. Given that $\mathbb{Z}/p\mathbb{Z}$ is a field, the extended Euclidean algorithm in $\mathbb{Z}/p\mathbb{Z}[x]$ provides polynomials $s(x)$ and $t(x)$ such that the input conditions of the Hensel Step Algorithm are satisfied for $n = p$. One application of the algorithm yields four polynomials, say $p_1(x), q_1(x), s_1(x)$ and $t_1(x)$ which in turn satisfy the input conditions of the algorithm for $n = p^2$, allowing us to apply the Hensel Step Algorithm to $p_1(x), q_1(x), s_1(x)$ and $t_1(x)$. Iterating this process $l$ times, we obtain polynomials $p_l(x), q_l(x), s_l(x)$ and $t_l(x)$, having the properties that $r(x) \equiv p_l(x)q_l(x) \mod p^2$, $p_l(x) \equiv p(x) \mod p$ and $q_l(x) \equiv q(x) \mod p$. The Hensel Lifting Algorithm formalizes this procedure.

**Algorithm 4.16 (Hensel Lifting).**

**Input:** A prime number $p \in \mathbb{Z}$, a nonconstant polynomial $r(x)$ in $\mathbb{Z}[x]$ such that $p$ does not divide the leading coefficient of $r(x)$, monic nonconstant polynomials $r_0(x), \ldots, r_d(x)$ in $\mathbb{Z}[x]$ which are pairwise coprime modulo $p$ and satisfy $r(x) \equiv \text{lc}(r(x))r_0(x) \cdots r_d(x) \mod p$. A natural number $m$.

**Output:** Monic polynomials $r^*_0(x), \ldots, r^*_d(x)$ in $\mathbb{Z}[x]$ with $r(x) \equiv \text{lc}(r(x)) \cdot r^*_0(x) \cdots r^*_d(x) \mod p^m$ and $r^*_i(x) \equiv r_i(x) \mod p$ for all $i$.

**Step 1:** If $d = 0$

Return $r^*_0(x) = r(x)/\text{lc}(r(x))$

**Step 2:** $k = \lceil d/2 \rceil$

$p_0(x) \equiv \text{lc}(r(x))r_0(x) \cdots r_k(x) \mod p$

$q_0(x) \equiv r_{k+1}(x) \cdots r_d(x) \mod p$

**Step 3:** Call the extended Euclidean algorithm with input $p_0(x), q_0(x)$

$s_0(x), t_0(x)$ in $(\mathbb{Z}/p\mathbb{Z})[x]$

**Step 4:** $l = \lceil \log_2 m \rceil$

For $j = 1, \ldots, l$

Call the Hensel Step Algorithm 4.14 with $n = p^{2^{j-1}}$

$p_j(x), q_j(x), s_j(x), t_j(x)$

**Step 5:** Call this algorithm recursively with input $p_l(x)$

**Step 6:** Call this algorithm recursively with input $q_l(x)$

The inductive argument that shows the correctness of the Hensel Lifting Algorithm is left to the reader. Before turning our attention to the algorithm for finding integer roots, we prove an auxiliary statement about Hensel Lifting needed for the justification thereof.
Lemma 4.17. Suppose that nonzero polynomials \( p(x), p^*(x), q(x) \) and \( q^*(x) \) over the integers satisfy the following conditions for some nonzero integer \( n \):

(i) \( p(x) \equiv p^*(x) \) and \( q(x) \equiv q^*(x) \mod n \)

(ii) The polynomials \( p(x) \) and \( p^*(x) \) as well as \( q(x) \) and \( q^*(x) \) have the same degree and leading coefficient.

(iii) The leading coefficients of \( p(x) \) and \( q(x) \) are coprime to \( n \).

(iv) There are \( s(x) \) and \( t(x) \) in \( \mathbb{Z}[x] \) such that \( s(x)p(x)+t(x)q(x) \equiv 1 \mod n \).

If there exists a natural number \( m \) such that \( p(x)q(x) \equiv p^*(x)q^*(x) \mod n^m \), then \( p(x) \equiv p^*(x) \mod n^m \) and \( q(x) \equiv q^*(x) \mod n^m \).

In other words, the above lemma states that the polynomials which the Hensel Lifting Algorithm 4.16 outputs are unique when viewed as elements of \( \mathbb{Z}/p^m\mathbb{Z}[x] \).

Proof. If on the contrary \( p(x) \not\equiv p^*(x) \mod n^m \) or \( q(x) \not\equiv q^*(x) \mod n^m \), we may assume without loss of generality that there exists a natural number \( 0 \leq i < m \) such that \( p(x) - p^*(x) = u(x)n^i \) for a polynomial \( u(x) \) and \( q(x) - q^*(x) = v(x)n^i \) for a polynomials \( v(x) \) which is not divisible by \( n \). By assumption, we then have that

\[
0 \equiv p(x)q(x) - p^*(x)q^*(x) \\
\equiv p(x)(q(x) - q^*(x)) + q^*(x)(p(x) - p^*(x)) \\
\equiv n^i[p(x)v(x) + q^*(x)u(x)] \mod n^m.
\]

Since the exponent \( i \) is strictly less than \( m \), \( n \) divides \( p(x)v(x) + q^*(x)u(x) \). By conditions (i) and (iv), we may infer that

\[
0 \equiv s(x)[p(x)v(x) + q^*(x)u(x)] \\
\equiv [1 - t(x)q(x)]v(x) + s(x)q^*(x)u(x) \\
\equiv v(x) + q(x)[s(x)u(x) - t(x)v(x)] \mod n.
\]

Therefore, \( q(x) \) is a factor of \( v(x) \) when viewed as polynomials over \( \mathbb{Z}/n\mathbb{Z} \). However, the polynomial \( v(x) \) is of degree strictly less than \( q(x) \), according to condition (ii). As the leading coefficient of \( q(x) \) is not a zero divisor modulo \( n \) by condition (iii), this implies that \( v(x) \) congruent to the zero polynomial modulo \( n \), which yields a contradiction to the assumption that \( v(x) \) is not divisible by \( n \). 

\[\Box\]
4.2.3 The Integer Root Finding Algorithm

Combining the Root Finding Algorithm over Finite Fields 4.13 and the Hensel Lifting Algorithm 4.16 results in an efficient algorithm for finding the distinct bounded integer roots of an arbitrary polynomial over the integers.

Algorithm 4.18 (Root Finding over the Integers).

**Input:** A nonzero polynomial \( r(x) \in \mathbb{Z}[x] \) and a natural number \( X \).

**Output:** The distinct integer roots \( \{x_1, \ldots, x_n\} \) of \( r(x) \) that are bounded by \( X \).

**Step 1:** \{Without loss of generality \( r(x) \) is squarefree and primitive\}

If \( \deg(r(x)) = 0 \)

Return \( \emptyset \)

Else

\[
    r_0(x) = \gcd(r(x), r'(x))
\]

\[
    r(x) = \frac{r(x)}{r_0(x)}
\]

**Step 2:** \{Finding a suitable prime number\}

\( p = 3 \)

While \( p \mid \text{lc}(r(x)) \) or \( \gcd(r(x), r'(x)) \neq 1 \) in \( \mathbb{Z}/p\mathbb{Z}[x] \)

\( p = \text{nextprime}(p) \)

**Step 3:** \{Root finding modulo \( p \)\}

Call the Root Finding over Finite Fields Algorithm 4.13 \( \{x_1, \ldots, x_d\} \)

**Step 4:** For \( i = 1, \ldots, d \)

\[
    r_i(x) = x - x_i
\]

\[
    r_0(x) = (\text{lc}(r(x)))^{-1} \frac{r(x)}{r_1(x) \cdots r_d(x)} \text{ in } \mathbb{Z}/p\mathbb{Z}[x]
\]

**Step 5:** \{Hensel Lifting\}

\( m = 1 \)

While \( p^m < 2X \)

\( m = m + 1 \)

Call the Hensel Lifting Algorithm 4.16 \( r_0^*(x), \ldots, r_d^*(x) \)

**Step 6:** \( n = 0 \)

For \( i = 1, \ldots, d \)

\( y_i \in \mathbb{Z} \) so that \( (x - y_i) \equiv r_i^*(x) \mod p^m \) and \( |y_i| \leq p^m/2 \)

If \( |y_i| \leq X \) and \( r(y_i) = 0 \) in \( \mathbb{Z} \)

\( n = n + 1 \)
Step Algorithm.

Lifting Algorithm basically consists of iterative applications of the Hensel algorithm in \( \mathbb{Z} \) obtaining having determined the polynomials

\[
x_1 = a, \quad r_1(x) = x^2 + 2.
\]

To factor the remaining quadratic term we choose another polynomial, say \( p(x) = x^2 + 3x + 2 \), resulting in a first factorization of the form

\[
x = (x^2 + 2)(x^2 + 3x + 2).
\]

Let us illustrate this procedure by applying the above algorithm to an example. Suppose that we want to find the distinct roots of the polynomial

\[ r(x) = 6x^6 + 144x^5 - 1708x^4 - 14380x^3 + 145230x^2 - 4806x + 48600 \]

that are bounded by \( X = 10 \).

For Step 1, we employ the Euclidean algorithm to compute the greatest common divisor of \( r(x) \) and \( r'(x) = 36x^5 + 570x^4 - 6832x^3 - 43140x^2 + 290460x - 4806 \), obtaining \( r_0(x) = 2x - 18 \). By polynomial long division, we thus have \( r(x) = 3x^5 + 84x^4 - 98x^3 - 8072x^2 - 33x - 2700 \). Step 2 yields \( p = 5 \) since 3 clearly divides the leading coefficient of \( r(x) \).

In Step 3, we apply Algorithm 4.13 to find the roots of \( r(x) \) modulo 5: The greatest common divisor of \( r(x) \) and \( r'(x) \) is equal to \( x^3 - 2x^2 + 2x \), meaning that \( r(x) \) has three roots in \( \mathbb{Z}/5\mathbb{Z} \). In order to determine those, we choose a random polynomial \( a(x) \in \mathbb{Z}/5\mathbb{Z}[x] \), say \( a(x) = x - 1 \). As \( a(x) \) and \( r(x) \) are coprime in \( \mathbb{Z}/5\mathbb{Z}[x] \), we compute the greatest common divisor of \( b(x) = a(x)(x^3 - 2x^2 + 2x) \) and \( r(x) \), resulting in a first factorization of the form \( r(x) = (x^2 - 2x + 2)(x^2 + 2) \).

To factor the remaining quadratic term we choose another polynomial, say \( a(x) = x + 1 \), which happens not to be coprime to \( x^2 - 2x + 2 \), yielding the factorization \( x^2 - 2x + 2 = (x + 1)(x + 2) \). Therefore, the roots of \( r(x) \) modulo 5 are 0, -1 and -2.

In preparation for Hensel Lifting, we thus set \( r_1(x) = r_2(x) = x + 1 \), \( r_3(x) = x + 2 \), \( r_0(x) = x^2 + 2 \) and \( m = 2 \). Recall that the Hensel Lifting Algorithm basically consists of iterative applications of the Hensel Step Algorithm.

We first apply the Hensel Step Algorithm to \( r(x) \),

\[
p_0(x) = l_0(r(x))r_0(x)r_1(x) = -2x^3 + x, \\
q_0(x) = r_2(x)r_3(x) = x^2 - 2x + 2, \\
s_0(x) = 2x - 2 \\text{and} \\
t_0(x) = -x^2 - x - 2,
\]

having determined the polynomials \( s_0(x) \) and \( t_0(x) \) by the Euclidean algorithm in \( \mathbb{Z}/5\mathbb{Z}[x] \): Following the procedure outlined in Step 1, we compute

\[
c(x) = 5x^5 + 5x^4 + 5x^3 + 5x^2 - 10x, \\
d(x) = 5x - 10, \\
e(x) = 10x^4 - 5x^3 - 5x^2 + 5,
\]

obtaining

\[
p_1(x) = p_0(x) + t_0(x)c(x) + e(x)p_0(x) = 3x^2 + x \\text{and} \\
q_1(x) = q_0(x) + d(x) = x^2 + 3x - 8.
\]
We may skip Step 2 of the Hensel Step Algorithm since we are not interested in $s_1(x)$ or $t_1(x)$.

In order to lift the congruences

\[ p_1(x) \equiv \text{lcm}(p(x))r_0(x)r_1(x) \quad \text{and} \quad q_1(x) \equiv r_2(x)r_3(x) \]

modulo 5 to congruences modulo $5^2$, we apply the Hensel Step Algorithm twice more - first with input $r(x) = p_1(x)$ and then with input $r(x) = q_1(x)$. We follow the same procedure as above with input values

\[
\begin{align*}
 r(x) &= p_1(x) = 3x^3 + x, \\
p_0(x) &= r_0(x) = -2x^2 + 1, \\
q_0(x) &= r_1(x) = x, \\
s_0(x) &= 1 \quad \text{and} \quad t_0(x) = 2x
\end{align*}
\]

and compute

\[
\begin{align*}
 c(x) &= 5x^3, \quad d(x) = 0, \quad e(x) = 5x^2
\end{align*}
\]

to obtain

\[
\begin{align*}
p_1(x) &= 3x^2 + 1 \quad \text{and} \quad q_1(x) = x.
\end{align*}
\]

As the multiplicative inverse of 3 is $-8$ in $\mathbb{Z}/25\mathbb{Z}$, we conclude that $r^*_0(x) = x^2 - 8$ and $r^*_1(x) = x$.

A final application of the Hensel Step Algorithm with input values

\[
\begin{align*}
 r(x) &= q_1(x) = x^2 + 3x - 8, \\
p_0(x) &= r_2(x) = x + 1, \\
q_0(x) &= r_3(x) = x + 2, \\
s_0(x) &= -1 \quad \text{and} \quad t_0(x) = 1
\end{align*}
\]

yields $r^*_2(x) = p_1(x) = x - 9$ and $r^*_3(x) = q_1(x) = x + 12$.

We thus have the following potential roots of the polynomial $r(x)$ over the integers:

\[
y_1 = 0, \quad y_2 = 9 \quad \text{and} \quad y_3 = -12.
\]

In Step 6, we eliminate $y_1$ and $y_3$ since $r(y_1) = r(0) = -2700 \neq 0$ and $|y_3| = 12 > 10 = X$, resulting in the fact that the algorithm only returns $x_1 = y_2 = 9$ (as $r(9) = 0$).

Given the factorization $r(x) = (x + 25)(x - 9)(x + 12)(3x^2 + 1)$, it is easy to conclude that 9 is indeed the only integer root of $r(x)$ which is bounded by 10, establishing the correctness of the algorithm for this particular example.

Before turning our attention to the general correctness of the above root finding algorithm, let us discuss the probability of finding a suitable prime number for Step 2 in a reasonable amount of time.

We now prove that the algorithm is guaranteed to find all distinct bounded roots of any polynomial over the integers.
Proposition 4.19. Algorithm 4.18 is correct.

Proof. Let \( r(x) \) be a polynomial with integer coefficients and let \( a \in \mathbb{Z} \) be a root of \( r(x) \) which is bounded by some natural number \( X \). We will show that Algorithm 4.18 with input \( r(x) \) and \( X \) returns the root \( a \).

To justify the first step of the algorithm, it is sufficient to verify that \( a \) is a root of the original polynomial \( r(x) \) if and only if it is a root of the polynomial \( f(x) = r(x)/\gcd(r(x), r'(x)) \) where \( r'(x) \) denotes the derivative of \( r(x) \). As \( f(x) \) is a factor of \( r(x) \), one implication is clear. For the other direction, we prove a more general statement: If the \( m \)-th power of an irreducible non-constant polynomial \( q(x) \) divides \( r'(x) \), then \( q^{m+1}(x) \) divides \( r(x) \). Indeed, assuming \( r(x) \) to be the product of a polynomial \( p(x) \) which is not divisible by \( q(x) \) and \( q^n(x) \) with \( n \leq m \), we have that

\[
r'(x) = q^{n-1}(x)[nq'(x)p(x) + q(x)p'(x)].
\]

Since \( q^n(x) \) divides \( r(x) \), the above equation implies that \( q(x) \) is a factor of \( nq'(x)p(x) \), which is contradictory to the irreducibility of \( q(x) \), thus proving the claim. Notice that equation \( \ast \) further entails that any factor of \( r(x) \) with multiplicity \( m \) is a factor of its derivative with multiplicity \( m-1 \), which allows us to infer that \( f(x) \) is squarefree, and hence that Step 2 terminates after a finite number of loops. Thus far, we have reduced the problem to finding the bounded integer roots of a squarefree polynomial \( r(x) \) given a prime number \( p \) with the properties that it does not divide the leading coefficient of \( r(x) \) and that \( r(x) \) is squarefree in \( \mathbb{Z}/p\mathbb{Z}[x] \).

As \( a \) is a root of \( r(x) \) over the integers, it is also a root over \( \mathbb{Z}/p\mathbb{Z} \), that is to say, there exists an index \( i \) such that \( r_i(x) \) is congruent to \( x - a \) modulo \( p \). As \( r(x) \) is squarefree in \( \mathbb{Z}/p\mathbb{Z}[x] \), we may apply the Hensel Lifting Algorithm, thus obtaining a monic linear factor \( r_i^*(x) \) of \( r(x) \) in \( \mathbb{Z}/p^m\mathbb{Z}[x] \) with satisfies \( r_i^*(x) \equiv x - a \mod p \). Let \( y_i \) be the unique integer so that \( r_i^*(x) \) is congruent to \( x - y_i \) modulo \( p^m \) and \( y_i \) is less than \( p^m/2 \) in absolute value.

We claim that the integers \( a \) and \( y_i \) are equal, which proves that the algorithm is correct. Indeed, Lemma 4.17 states that \( a \) and \( y_i \) are congruent modulo \( p^m \). Using the fact that \( X \) is less than \( p^m/2 \), we see that both integers are bounded by \( p^m/2 \), which entails that they coincide. \( \square \)

4.3 The Lattice-Based PACDP Algorithm

In this section, we combine the results presented in the two preceding sections, thus obtaining a lattice-based algorithm for solving partially approximate common divisor problems. Indeed, in Section 4.1, we saw how to construct a polynomial \( r(x) \) in \( \mathbb{Z}[x] \) having the property that the noise of any solution to the PACDP with parameters \( a, b, M \) and \( X \) is guaranteed to be an integer root of \( r(x) \), provided that \( \log_b X < (\log_b M)^2 \). Given that
the noise of any approximate common divisor is bounded by $X$, it therefore remained to calculate the distinct bounded roots of an arbitrary polynomial over the integers. In consequence, Section 4.2 is dedicated to the justification of Algorithm 4.18, which takes a bound $X$ and a polynomial $r(x)$ in $\mathbb{Z}[x]$ as input and returns the set of all roots of $r(x)$ that are less than $X$ in absolute value. Implementing this procedure, results in the following algorithm:

**Algorithm 4.20 (Lattice-Based PACDP).**

**Input:** Natural numbers $a$, $b$, $M$ and $X$ so that $\log_b X < (\log_b M)^2$ and $\max\{a, M, X\} < b$.

**Output:** The solution set $S$ of all $(d, x_0)$ in $\mathbb{Z}^2$ so that $d$ is the greatest common divisor of $a + x_0$ and $b$, $d$ is greater than $M$ and $x_0$ is less than $X$ in absolute value, or ”failure”.

**Step 1:**

\[
\mu = \log_b M \\
\xi = \log_b X \\
n = \lfloor \mu \left(\frac{\mu(1 - \mu)}{\mu^2 - \xi}\right) \rfloor \\
l = \lfloor \frac{\mu(1 - \mu)}{\mu^2 - \xi} \rfloor - n
\]

**Step 2:** \{Construction of a basis of the lattice $\Lambda(n, l)$\}

For $i = 0, \ldots, n$

For $j = 0, \ldots, i$

$v_{i+1,j} = b^{n-i} \binom{i}{j} X^j a^{i-j}$

For $j = i + 1, \ldots, n + l$

$v_{i+1,j} = 0$

$v_{i+1} = (v_{i+1,0}, \ldots, v_{i+1,n+l})$

For $i = 1, \ldots, l$

For $j = i, \ldots, i + n$

$v_{i+n+1,j} = \binom{n}{j-i} a^{n+i-j} X^j$

For $j = 0, \ldots, i - 1$ and $j = n + i + 1, \ldots, n + l$

$v_{i+n+1,j} = 0$

$v_{i+n+1} = (v_{i+n+1,0}, \ldots, v_{i+n+1,n+l})$

**Step 3:** Call the LLL-Algorithm A.10 with input $v_1, \ldots, v_{n+l+1}$

\[w \in \Lambda(n, l) \subset \mathbb{Z}^{n+l+1}\]

**Step 4:** write $w = (w_0, \ldots, w_{n+l})$ with $w_i \in \mathbb{Z}$

If $(n + l + 1)\|w\| \geq M^n$

Return ”failure”

Else

For $i = 0, \ldots, n + l$

$r_i = w_i / X^i$
\[ r(x) = \sum_{i=0}^{n+l} r_i x^i \in \mathbb{Z}[x] \]

**Step 5:** Call Algorithm 4.18 with input \( r(x) \) and \( \{y_1, \ldots, y_k\} \)

**Step 6:** \( S = \emptyset \)

For \( i = 1, \ldots, k \)

\[
d = \gcd(a + y_i, b)
\]

If \( d \geq M \)

\[
S = S \cup \{(d, y_i)\}
\]

**Step 7:** Return \( S \)

The probability that Algorithm 4.20 fails is extremely low. In fact, the larger the parameters, the lower the probability of a failure. The option to return "failure" merely guarantees theoretical correctness.

**Proposition 4.21.** Algorithm 4.20 is correct.

As the two preceding sections were written with a view towards establishing a PACDP algorithm, the proof of correctness contains nothing new, but is merely a summary.

**Proof.** We first observe that the vectors \( v_1, \ldots, v_{n+l+1} \) defined in Step 2 form a basis of the lattice \( \Lambda(n, l) \) since, for all indices \( i \), \( v_i \) is the \( i \)-th row vector of the generating matrix \( A(n, l) \), given by 4.2. Moreover, owing to the choice of \( n \) and \( l \), the norm of the vector \( w \), computed by the LLL-Algorithm with input \( v_1, \ldots, v_{n+l+1} \), should satisfy \((n+l+1)||w|| < M^n\).

Supposing this to be the case, we infer that the polynomial \( r(x) \) derived form \( w \) in Step 4 has the properties that

(i) \( r(x) = \sum_{i=0}^{n} h_i(x)q_i(x) \) for some \( h_i(x) \in \mathbb{Z}[x] \),

(ii) \( (\deg r(x) + 1)||r(xX)|| < M^n \),

(iii) \( r(x) \) is not equal to zero in \( \mathbb{Z}[x] \).

By Proposition 4.4, this implies that the noise of any solution to the PACDP is a root of the polynomial \( r(x) \). Hence, if \( x_0 \) is the noise corresponding to an approximate common divisor \( d \) of \( a \) and \( b \), there exists an index \( 1 \leq i \leq k \) such that \( y_i \) is equal to \( x_0 \). By definition, \( d \) is the greatest common divisor of \( a + y_i \) and \( b \), which is greater than \( M \), entailing that the pair \( (d, y_i) = (d, x_0) \) is an element of the solution set \( S \).

In order to illustrate the Lattice-Based PACDP Algorithm as well as a difference between the approaches based on continued fraction approximation
and lattice reduction, respectively, we apply Algorithm 4.20 to the example considered at the beginning of Section 3.2. Recall that, according to the definition used in Section 3.2, the natural numbers \( a = 49007 \) and \( b = 100,000 \) posses exactly one approximate common divisor, namely \( d = 50,000 \) with noise \( x_0 = 993 \). On the other hand, when setting \( M = 1000 \) and \( X = 10 \), the lattice-based algorithm will output that \( d = 1000 \) and \( x_0 = -7 \) is the only solution to the partially approximate common divisor problem:

As the values for \( n \) and \( l \) determined by algorithm are primarily intended for large parameters, we skip Step 1 and set \( n = 2 \) and \( l = 1 \) instead. Notice that Inequality 4.3 is not satisfied, which means that it is possible that the LLL-Algorithm will not return a short enough vector. In Step 2, we obtain

\[
\begin{pmatrix}
 v_1 \\
 v_2 \\
 v_3 \\
 v_4 \\
\end{pmatrix} =
\begin{pmatrix}
 b^2 & 0 & 0 & 0 \\
 ab & Xb & 0 & 0 \\
 a^2 & 2Xa & X^2 & 0 \\
 0 & Xa^2 & 2X^2a & X^3 \\
\end{pmatrix}
= 
\begin{pmatrix}
 10^{10} & 0 & 0 & 0 \\
 10^6 a & 10^6 & 0 & 0 \\
 a^2 & 2 \cdot 10^a & 10^2 & 0 \\
 0 & 10a^2 & 2 \cdot 10^2a & 10^3 \\
\end{pmatrix}.
\]

Calling the LLL-Algorithm A.10 with input \( v_1, v_2, v_3 \) and \( v_4 \) yields a vector \( w = (3430, 14700, 21000, 10000) = -117649v_1 + 720300v_2 - 979930v_3 + 10v_4 \) in \( \Lambda(n,l) \). As its norm \( ||w|| = 10\sqrt{7688549} \) is less than \( 10^6 = M^n \), we set

\[
r(x) = 3430 + 1470x + 210x^2 + 10x^3
\]

in Step 4. According to Algorithm 4.18, the number \(-7\) is the only root of the polynomial \( r(x) \) in \( \mathbb{Z}[x] \) that is smaller than 10 in absolute value. We therefore conclude that \( x_0 = -7 \) and \( d = 1000 \) is the only solution.

Notice that for solving this specific example a brute force attack would have been more efficient; however, we chose the parameters to be so small that complexity is hardly an issue. In general, whether a brute force attack or the Lattice-Based PACDP Algorithm is more efficient, depends on the parameters of the PACDP in question. Given that the complexity of Step 3 dominates that of the other Steps in Algorithm 4.20, the algorithm uses \( O((n + l)^9(\log b)^3) \) bit operations, according to Formula 4.5. Hence, the number of bit operations needed primarily depends on the size of the two bounds \( M \) and \( X \) relative to \( b \). The cost of the brute force attack, on the other hand, is mainly determined by the absolute size of \( X \) since it requires \( O(X(\log b)^2) \) bit operations.

### 4.4 A Generalization of the Partially Approximate Common Divisor Problem

In [3] (57), Howgrave-Graham mentions the following generalization of the approximate common divisor problem without discussing it any further: Let \( a, b, M \) and \( X \) be parameters to a PACDP and let \( s \) and \( t \) be natural numbers.
The generalized problem is to find the set of all pairs \((d, x_0)\) in \(\mathbb{Z}^2\) with the properties that

(i) \(d \geq M\),
(ii) \(|x_0| \leq X\),
(iii) \(d^s\) divides \(a + x_0\) and \(d^t\) is a factor of \(b\).

The Lattice-Based PACDP Algorithm 4.20 presented in the previous section is, in fact, even better suited to this problem. In what follows, we will explain the slight generalization necessary.

As in Section 4.1, we consider the polynomials \(p_a(x) = x + a\) and \(p_b(x) = b\) in \(\mathbb{Z}[x]\). In order to solve this generalized problem, we need to find all integers \(x_0\) bounded by \(X\) which satisfy

\[
p_a(x_0) \equiv 0 \mod d^s \quad \text{as well as} \quad p_b(x_0) \equiv 0 \mod d^t
\]

for some integer \(d \geq M\). Using the ideas developed for the standard problem, we will reduce this generalized problem to the problem of finding bounded integer roots of a polynomial \(r(x)\) in \(\mathbb{Z}[x]\), too.

We consider the three cases \(s = t\), \(s > t\) and \(s < t\) separately.

**case 1:** Applying Lemma 4.1 to \(d^s = d^t\) instead of \(d\), we infer that the noise of a solution \((d, x_0)\) to the generalized problem is a root over \(\mathbb{Z}/d^{sn}\mathbb{Z}\) of any \(\mathbb{Z}[x]\)-linear combination of the polynomials \(q_0(x), \ldots, q_n(x)\), given by

\[
q_i(x) = (x + a)^ib^n - i \quad \text{in} \quad \mathbb{Z}[x].
\]

Therefore, we consider polynomials that satisfy the following two conditions for some natural number \(n\):

(i) \(r(x) = \sum_{i=0}^{n} h_i(x)q_{i}(x)\) for some \(h_i(x) \in \mathbb{Z}[x]\),
(ii') \((\deg r(x) + 1)||r(x|| < M^n\).

By the same argumentation as in Section 4.1, the set of all solutions to the generalized PACDP with parameters \(a, b, M, X, s\) and \(t\) is a subset of the set of the distinct integer roots of any polynomial \(r(x)\) satisfying (i) and (ii').

It remains to construct a polynomial with the desired properties. According to Lemma 4.5, the polynomials of degree at most \(n + l\) which satisfy condition (i) are in exact correspondence with the elements of the lattice \(\Lambda(n, l)\) generated by the matrix \(A(n, l)\), defined by 4.2. By the analysis for the standard case \(s = 1\), applying the LLL-Algorithm to the lattice \(\Lambda(n, l)\) yields a sufficiently short vector to satisfy (ii'), given that the following in-
equality holds:

\[ 2^{(l+n)/2}(l + n + 1)^{3/2} \left( \prod_{i=1}^{n} b_i \prod_{j=1}^{n+l} X^j \right)^{\frac{1}{(l+n+1)}} < M^s. \]

As before, we set \( \xi = \log_b X \) and \( \mu = \log_b M \). By neglecting the relatively small term \( 2^{(l+n)/2}(l + n + 1)^{3/2} \), the inequality simplifies to

\[ \xi < \frac{2\mu sn(k + 1) - n(n + 1)}{(k + 1)k} \]

for a pair \((n, k)\) of natural numbers with \( n \leq k \). Notice that this inequality is not identical to the one considered for solving the standard problem since there is an additional factor \( s \) in the first summand of the numerator. Further notice that since, for any approximate common divisor \( d, b \) is of the hidden form \( b = db' = d^s b' \) for some natural number \( b' \), if the product \( \mu s \) were greater than 1, it would imply that \( d^s \geq M^s = (b^\mu)^s > b = d^s b' \), which is contradictory. Thus we may set \( n = \lfloor \mu sk \rfloor \) (since \( \lfloor \mu sk \rfloor \leq k \)). If we replace \( n \) by \( \mu sk \), we obtain

\[ \xi < (\mu s)^2 - \frac{\mu s(1 - \mu s)}{k + 1}. \]

Hence, if there exists \( \varepsilon > 0 \) such that \( \xi = (\mu s)^2 - \varepsilon \), it is possible to find natural numbers \( n \) and \( k \) with the desired properties, namely

\[ n = \lfloor \mu sk \rfloor \text{ and } k = \left\lfloor \frac{\mu s(1 - \mu s)}{\varepsilon} \right\rfloor = \left\lfloor \frac{\mu s(1 - \mu s)}{(\mu s)^2 - \xi} \right\rfloor. \]

Notice that this generalization of the Lattice-Based PACDP Algorithm amounts to applying Algorithm 4.20 to the PACDP with parameters \( a, b, M^s \) and \( X \), which entails that it returns not only solutions to the generalized problem, but also all pairs \((d, x_0)\) in \( \mathbb{Z}^2 \) with \( |x_0| \leq X \) so that \( d = \gcd(a + x_0, b) \geq M^s \).

Recall that Howgrave-Graham’s technique works for the standard PACDP provided that \( \mu^2 > \xi \), which is stronger than the condition that \( (\mu s)^2 > \xi \) needed for this generalization of his attack. In consequence, the generalized PACDP can be solved theoretically for a wider range of the parameters \( X, M \) and \( b \) than the standard problem. In addition, considering the same values of the parameters \( X, M \) and \( b \) for both problems, the size of the matrix \( A(n, l) \) - which dominates the complexity of the algorithm - is larger for the standard attack.

**case 2:** Let us now turn to the case \( s > t \). Assuming \((d, x_0)\) to be a solution to the generalized PACDP, we have that \( p_a(x_0) \equiv 0 \mod d^s \) and \( p_b(x_0) \equiv 0 \mod d^t \) and thus

\[ q_i(x_0) = p_a(x_0)^i p_b(x_0)^{n-i} \equiv 0 \mod d^{si+t(n-i)} \]
for all indices $0 \leq i \leq n$. However, if we now consider a polynomial linear combination

$$r(x) = \sum_{i=0}^{n} h_i(x)q_i(x),$$

we may only infer that $d^{\min\{si+t(n-i): 0 \leq i \leq n\}} = d^t$ divides $r(x_0)$, and in consequence we do not exploit the fact that $s$ is potentially much greater than $t$. The following generalization of Lemma 4.1 overcomes this difficulty:

**Lemma 4.22.** Let $\nu \geq 1$ be a natural number. Then any polynomial of the form

$$r(x) = \sum_{i=0}^{n} h_i(x)q_i(x)p_b(x)^{(n-i)(\nu-1)}$$

where $h_0(x), \ldots, h_n(x)$ in $\mathbb{Z}[x]$ has the following property: If $x_0$ is an integer such that $p_a(x_0) \equiv 0 \mod d^s$ and $p_b(x_0) \equiv 0 \mod d^t$, then $x_0$ satisfies

$$r(x_0) \equiv 0 \mod d^{\min\{si+\nu t(n-i): 0 \leq i \leq n\}}.$$

**Proof.** Assume that $d^s$ divides $p_a(x_0)$ and $d^t$ divides $p_b(x_0)$, then $d^{si+t(n-i)}$ and $d^{\nu t(n-i)}$ are factors of $q_i(x_0)$ and $p_b(x_0)^{(n-i)(\nu-1)}$, respectively. Hence, $q_i(x_0)p_b(x_0)^{(n-i)(\nu-1)} \equiv 0 \mod d^{si+\nu t(n-i)}$ for $0 \leq i \leq n$, which entails that $d^{\min\{si+\nu t(n-i): 0 \leq i \leq n\}}$ divides $r(x_0)$. \hfill \Box

Given that $s > t$, $\min\{si+\nu t(n-i): 0 \leq i \leq n\}$ is equal to $\min\{\nu t, s\}$. Setting $\nu = [s/t]$, we obtain $\min\{si+\nu t(n-i): 0 \leq i \leq n\} = \nu t n \approx s n$. This yields the following adaptation of Proposition 4.4:

**Lemma 4.23.** Let $s > t$ and set $\nu = [s/t]$. If a polynomial $r(x)$ satisfies the following two conditions for some natural number $n$:

1. $(i')$ $r(x) = \sum_{i=0}^{n} h_i(x)q_i(x)p_b(x)^{(n-i)(\nu-1)}$ for some $h_i(x) \in \mathbb{Z}[x],$

2. $(ii')$ $(\deg r(x) + 1)||r(x X)|| < M^\nu t n$,

then any solution to the generalized PACDP with parameters $a, b, M, X, s$ and $t$ is a root of $r(x)$ over the integers.

We now give an explicit construction of a polynomial $r(x)$ with the desired properties. Let us first focus on condition $(i')$. Consider the $(n+l+1) \times (n+l+1)$-matrix $\hat{A}(n, l)$ given by

$$
\begin{pmatrix}
q_0,0 & b^{n(\nu-1)} & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
q_1,0 & b^{n(\nu-1)} & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
q_n,0 & b^{n(\nu-1)} & q_{n,1}X & q_{n,2}X^2 & \cdots & q_{n,n}X^n & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
0 & 0 & q_{n,0}X^l & 0 & \cdots & \cdots & \cdots & \cdots & q_{n,n}X^{n+l}
\end{pmatrix}
$$
where $q_{i,j}$ is the $j$-th coefficient of the polynomial $q_i(x)$.  

For $1 \leq i \leq n$, the $i$-th row of $\tilde{A}(n,l)$ corresponds to the polynomial $q_i(x) b^{(n-i)(\nu-1)} = q_i(x) p_0(x X)^{(n-i)(\nu-1)}$, while the $(n+j)$-th row can be identified with the polynomial $(x X)^j q_n(x)$ if $1 \leq j \leq l$. We claim that any polynomial of degree at most $n + l$ that satisfies condition (i') corresponds to an integer linear combination of the rows of $\tilde{A}(n,l)$. Indeed, we can apply Lemma 4.5 to the matrix $\tilde{A}(n,l)$ by viewing it as a matrix of the form $A(n,l)$ for a different value of the parameter $b$: Defining $\tilde{q}_i(x)$ to be the polynomial $q_i(x)$ obtained through replacing $b$ by $b^\nu$, we have that $\tilde{q}_i(x) = q_i(x) p_0(x X)^{(n-i)(\nu-1)}$ as desired. Hence, any vector in the lattice $\tilde{\Lambda}(n,l)$ generated by $\tilde{A}(n,l)$ can be identified with a polynomial satisfying (i').

It remains to choose the variables $n$ and $l$ so that the LLL-Algorithm with input $\tilde{\Lambda}(n,l)$ is guaranteed to return a sufficiently short vector to satisfy condition (ii'). As

$$\det \tilde{\Lambda}(n,l) = \left| \det \tilde{A}(n,l) \right| = \left| \det A(n,l) \prod_{i=1}^{n} b^{(\nu-1)i} \right| = \prod_{i=1}^{n} b^\nu \prod_{j=0}^{n+l} X^j,$$

the LLL-Algorithm outputs a vector $v \in \tilde{\Lambda}(n,l)$ with the property that

$$||v|| \leq 2^{(l+n)/2} \sqrt{l+n+1} \left( \prod_{i=1}^{n} b^\nu \prod_{j=1}^{n+l} X^j \right)^{1/(l+n+1)},$$

according to Formula A.2. Therefore, if $n$ and $l$ satisfy the inequality

$$2^{(l+n)/2(l+n+1)} \left( \prod_{i=1}^{n} b^\nu \prod_{j=1}^{n+l} X^j \right)^{1/(l+n+1)} < M^\nu \mu,$$

we are assured of finding a suitable polynomial $r(x)$. In analogy to the standard attack, this simplifies to determining a pair $(n,k)$ of natural numbers with the property that $n \leq k$ and

$$\xi < \frac{2\nu \mu t n(k+1) - v n(n+1)}{(k+1)k},$$

where $\mu = \log_b M$ and $\xi = \log_b X$.

Setting $n$ equal to $\mu k$, this is equivalent to

$$\xi < \nu \left( \left(\mu t\right)^2 - \frac{\mu t(1-\mu t)}{k+1} \right)$$

since $\mu t \leq 1$ and thus $n \leq k$. Hence, if there exists $\varepsilon > 0$ such that $\xi = \nu((\mu t)^2 - \varepsilon)$, it is possible to find natural numbers $n$ and $k$ with the desired properties, namely

$$n = \lceil \mu k \rceil \text{ and } k = \left\lfloor \frac{\mu t(1-\mu t)}{\varepsilon} \right\rfloor = \left\lfloor \frac{\nu \mu t(1-\mu t)}{\nu(\mu t)^2 - \xi} \right\rfloor$$
where \( \nu = \lfloor s/t \rfloor \). Notice that this generalization of the attack amounts to applying Algorithm 4.20 to the PACDP with parameters \( a, b, M^\nu t \) and \( X \).

Recall that the generalization of the standard algorithm to the case \( s = t \) where \( t \) need not be equal to 1 presupposes that \((\mu t)^2 - \xi\). The generalization to the case \( s > t \), on the other hand, works whenever

\[
\lfloor s/t \rfloor (\mu t)^2 > \xi,
\]

which is a potentially weaker condition. Therefore, this adaptation of the attack makes us of the fact that \( s \) is greater than \( t \).

case 3: When considering the case \( s < t \), it is tempting to try and introduce a factor, say \( \mu \), so that we could consider the exponent \( \min \{\mu i + t(n - i) : 0 \leq i \leq n\} \) but then the corresponding generalization of the matrix \( A(n, l) \) loses its nice triangular form which allows us to bound the determinant of the lattice it generates. This means that we use the same attack for \( s = t \) and \( s < t \), and in consequence, find the same solutions. In brief, our generalization of the reduction presented in Section 4.1 is not well suited to the case \( s < t \), as it does exploit the fact that \( t \) is greater than \( s \).

4.5 The General Approximate Common Divisor Problem

In this section, we will generalize the reduction presented in Section 4.1 to the general approximate common divisor problem. To be more concrete, we will reduce the following problem to the problem of finding the discrete bounded roots of a bivariate polynomial \( r(x, y) \) in \( \mathbb{Z}[x, y] \): Let \( a, b, M, X \) and \( Y \) be natural numbers so that \( \max \{a, M, X, Y\} < b \). Then the GACDP with parameters \( a, b, M, X \) and \( Y \) consists of determining the solution set \( S \) given by

\[
S = \{(d, x_0, y_0) \in \mathbb{Z}^3 : d \geq M, |x_0| \leq X, |y_0| \leq Y \quad \text{and} \quad d = \gcd(a + x_0, b + y_0)\}.
\]

Throughout this section, the letters \( a, b, M, X \) and \( Y \) denote the parameters of a given general approximate common divisor problem.

In analogy to the justification of the Lattice-Based PACDP Algorithm, we consider the polynomials \( p_a(x, y) = x + a \) and \( p_b(x, y) = y + b \) in \( \mathbb{Z}[x, y] \). By definition, any solution \((d, x_0, y_0)\) satisfies

\[
p_a(x_0, y_0) \equiv 0 \quad \text{and} \quad p_b(x_0, y_0) \equiv 0 \mod d.
\]

Fixing a natural number \( n \), we infer that the polynomial \( q_i(x, y) \) in \( \mathbb{Z}[x, y] \) given by

\[
q_i(x, y) = p_a(x, y)^i p_b(x, y)^{n-i}
\]
for all $0 \leq i \leq n$, has the property that $q_i(x_0, y_0) \equiv 0 \mod d^n$. This connection between roots of bivariate polynomials and solutions to general approximate common divisor problems, will allow us to generalize the approach presented in Section 4.1.

Before stating the generalization of Proposition 4.4, which provides sufficient properties of a polynomial so that any solution to the PACDP corresponds to one of its roots, we define the norm of a bivariate polynomial.

**Definition 4.24.** The norm of a bivariate polynomial is the standard Euclidean norm of its coefficient vector, i.e. the norm of a polynomial $r(x, y)$ in $\mathbb{R}[x, y]$ is given by

$$\|r(x, y)\| = \|\sum_{i,j} r_{ij} x^i y^j\| = (\sum_{i,j} r_{ij}^2)^{\frac{1}{2}}.$$

For want of a standard notation, we let $N(r(x, y))$ denote the number of nonzero coefficients of the polynomial $r(x, y)$.

**Proposition 4.25.** Let $n$ be a natural number. Let $r(x, y)$ in $\mathbb{Z}[x, y]$ be a nonzero polynomial which satisfies the following two conditions:

(i) $r(x, y) = \sum_{i=0}^{n} h_i(x, y)q_i(x, y)$ where $h_i(x, y)$ in $\mathbb{Z}[x, y]$,

(ii) $N(r(x, y))\|r(xX, yY)\| < M^n$.

If the three tuple $(d, x_0, y_0)$ in $\mathbb{Z}^3$ is a solution to the GACDP, then $r(x_0, y_0)$ is equal to 0 in $\mathbb{Z}$.

**Proof.** Let $(d, x_0, y_0)$ lie in the solution set of the GACDP. As $d^n$ divides $q_i(x_0, y_0)$ for all indices $i$, condition (i) implies that $r(x_0, y_0)$ is congruent to zero modulo $d^n$. If $r(x_0, y_0)$ were strictly less than $d^n$ in absolute value, we could conclude that $r(x_0, y_0)$ is, in fact, equal to 0 in $\mathbb{Z}$.

We claim that this is indeed the case: By definition of the GACDP, $(x_0, y_0)$ lies in $[-X, X] \times [-Y, Y]$, and in consequence, we have that

$$|r(x_0, y_0)| = \left|\sum_{i,j} r_{ij} x_0^i y_0^j\right| \leq \sum_{i,j} |r_{ij}|X^i Y^j \leq N(r(x, y)) \max_{i,j} \{|r_{ij}|X^i Y^j|\}$$

$$\leq N(r(x, y))\|r(xX, yY)\| < M^n \leq d^n$$

where we used condition (ii) and the fact that $M$ is less than $d$. 

The next step is to construct a polynomial $r(x, y)$ with the desired properties. Like in the partially approximate case, this will be achieved by applying the LLL-Algorithm to a suitable lattice. Unlike before, however, the lattice will not be of full rank, which renders bounding its determinant more difficult.
We consider the lattice $\Lambda(n, k)$ given by setting
\[
\Lambda(n, k) = \{ r(x, y) = \sum_{i=0}^{n} h_i(x, y)q_i(x, y) \in \mathbb{Z}[x, y] : h_i(x, y) \in \mathbb{Z}[x, y] \text{ for } 0 \leq i \leq n \text{ and } \deg(r(x, y)) \leq k \}.
\]
Notice that elements of the lattice $\Lambda(n, k)$ are closely related to polynomials that satisfy the two conditions of Proposition 4.25. Indeed, whenever a polynomial $r(x, y)$ lies in $\Lambda(n, k)$, the corresponding polynomial $r(x, y)$ automatically satisfies the first condition. Moreover, if we write the element of $\Lambda(n, k)$ as vectors with $(k+2)(k+1)/2$ entires, any vector in $\Lambda(n, k)$ whose norm is strictly less than $2M^n/(k+2)(k+1)$ corresponds to a polynomial that satisfies condition (ii).

The following slightly technical lemma provides a basis of the lattice $\Lambda(n, k)$.

**Lemma 4.26.** Let $n \leq k$ be natural numbers. The lattice $\Lambda(n, k)$ is generated by the elements of
\[
\{ (y)^j q_i(x, y) : 0 \leq j \leq k - n, 0 \leq i \leq n - 1 \} \cup
\{ (x)^l (y)^j q_n(x, y) : 0 \leq l + j \leq k - n \}.
\]
\[(4.6)\]

**Proof.** We first prove the statement without placing any restrictions on the total degrees of the polynomials. We claim that the $\mathbb{Z}[x, y]$-span of
\[
\{ q_i(x, y) : 0 \leq i \leq n \}
\]
is a subset of the $\mathbb{Z}$-span of
\[
\{ (y)^j q_i(x, y) : j \in \mathbb{N}, 0 \leq i \leq n - 1 \} \text{ and }
\{ (x)^l (y)^j q_n(x, y) : l, j \in \mathbb{N} \} \text{ (\ast)}
\]
(\text{where we denote the first set by } B). To justify this claim, we show that any polynomial of the form $(x)^\lambda (y)^\mu q_0(x, y)$ lies in the $\mathbb{Z}$-module generated by $B$ and $\{ (x)^l (y)^j q_n(x, y) : j \in \mathbb{N}, 0 \leq l \leq \lambda + \nu - n \}$, by induction on the $x$-degree $\lambda + \nu$.

For the base case $\lambda + \nu \leq n$, we have
\[
\sum_{i=0}^{\lambda} \binom{\lambda}{i} (-a)^{\lambda-i} \left( \sum_{j=0}^{i} \binom{i}{j} b^{i-j} (y)^{\mu+j} q_{\nu+i}(x, y) \right)
\]
\[
= (y)^{\mu} \sum_{i=0}^{\lambda} \binom{\lambda}{i} (-a)^{\lambda-i} (y+b)^i q_{\nu+i}(x, y)
\]
\[
= (y)^{\mu} (x + a)^{\mu} (y + b)^{n-\nu} \sum_{i=0}^{\lambda} \binom{\lambda}{i} (-a)^{\lambda-i} (x + a)^i
\]
\[
= (y)^{\mu} q_0(x, y)(x)^{\lambda}.
\]
Notice that \( q_{\nu+i}(xX, yY) \) is well-defined for all \( i \in \{0, \ldots, \lambda\} \) because we are in the base case \( \lambda + \nu \leq n \).

For the induction step, suppose that the statement holds for some \( \lambda + \nu \geq n \). As \( \nu \) is less than \( n \), it is sufficient to infer that any polynomial of the form \( (xX)^{\lambda+1}(yY)^{\mu}q_{\nu}(xX, yY) \) satisfies the claim:

By induction hypothesis, the polynomial \( (xX)^{\lambda+1}(yY)^{\mu}q_{\nu}(xX, yY) = (xX)(xX)^{\lambda}(yY)^{\mu}q_{\nu}(xX, yY) \) can be written as a \( \mathbb{Z} \)-linear combination of elements in the union of

\[
(xX) \cdot B \text{ and } \{(xX)^{l+1}(yY)^{j}q_{\nu}(xX, yY) : j \in \mathbb{N}, 0 \leq l \leq \lambda + \nu - n\}.
\]

The induction step follows immediately since by the base case, any polynomial in the first set lies in the \( \mathbb{Z} \)-module spanned by \( \{(yY)^{j}q_{\nu}(xX, yY) : j \in \mathbb{N}, 0 \leq i \leq n\} \).

Now, the lemma is a consequence of the fact that the elements in the set given by * are \( \mathbb{R} \)-linearly independent.

\[ \square \]

In particular, Lemma 4.26 allows us to determine the number of generators of \( \Lambda(k, n) \) as well as to bound its determinant form above.

**Corollary 4.27.** The lattice \( \Lambda(n, k) \) has \((k - n + 1)(k + n + 2)/2\) generators and its determinant is bounded by.

**Proof.** Clearly, the number of generators of \( \Lambda(n, k) \) is equal to the cardinality of the union of sets given in 4.6. Since the two sets are disjoint, we infer that this basis contains

\[
n(k - n + 1) + \frac{(k - n + 1)(k - n + 2)}{2} = \frac{(k - n + 1)(k + n + 2)}{2}
\]

elements.

In order to bound the determinant of \( \Lambda(n, k) \), we first show the following general fact about lattices: If a lattice \( \Lambda \) is generated by \( v_1, \ldots, v_n \) in \( \mathbb{R}^m \), its determinant satisfies \( \det \Lambda \leq \|v_1\| \cdots \|v_n\| \). Indeed, denoting the corresponding Gram-Schmidt basis by \( v_1^*, \ldots, v_n^* \), we have that \( \det \Lambda \) is equal to the product \( \|v_1^*\| \cdots \|v_n^*\| \), according to Lemma A.12. Moreover, by A.1, the norm \( \|v_i^*\| \) is less than \( \|v_i\| \) for any index \( i \), which proves the statement.

It remains to bound the norms of the polynomials lying in the above basis of \( \Lambda(n, k) \). The crucial observation is that

\[
\|q_i(xX, yY)\| \leq \sqrt{(n+1)(n+2)b^n} \text{ for all } 0 \leq i \leq n.
\]

Therefore, the norm of a polynomial of the form \((xX)^l(yY)^jq_i(xX, yY)\) is less than \((n+1)^{1/2}(n+2)^{1/2}X^lY^jb^n\). Multiplying these upper bounds over all elements of the basis we infer that

\[
[(n+1)^{1/2}(n+2)^{1/2}b^n]^{(k-n+1)(k+n+2)/2} \left(X^{(k-n+2)}Y^{(k-n+5)}\right)^{(k-n)(k-n+1)/6}
\]

bounds the determinant of \( \Lambda(n, k) \) from above. \[ \square \]
For ease of notation, we let $\varrho(n, k)$ denote the number of generators of $\Lambda(n, k)$ in relation to the variables $n$ and $l$. According to Proposition A.20, the LLL-Algorithm with the above basis of $\Lambda(n, k)$ as input, returns a lattice element whose norm is less than

$$2^{(\varrho(n, k) - 1)/2} \sqrt{\varrho(n, k) \langle \text{det } \Lambda(n, k) \rangle^{1/\varrho(n, k)}}.$$ 

By Corollary 4.27, the LLL-Algorithm thus outputs a polynomial whose norm is guaranteed to be less than

$$2^{(\varrho(n, k) - 1)/2} \sqrt{\varrho(n, k) \langle \mathcal{B}[\text{det } \Lambda(n, k)] \rangle^{1/\varrho(n, k)},}$$

where $\mathcal{B}[\text{det } \Lambda(n, k)]$ denotes the upper bound given. We therefore conclude that the LLL-Algorithm provides an element in $\Lambda(n, k)$ which satisfies all conditions of Proposition 4.25 whenever

$$2^{(\varrho(n, k) - 1)/2} \sqrt{\varrho(n, k) \langle \mathcal{B}[\text{det } \Lambda(n, k)] \rangle^{1/\varrho(n, k)}} < 2M^n / (k + 2)(k + 1).$$

By Corollary 4.27, this yields an explicit inequality in the variables $n$ and $k$. In theory, it is possible to find natural numbers $n$ and $k$ for which the above inequality is satisfied, provided that the parameters $b$, $M$, $X$ and $Y$ satisfy certain conditions. In practice, this requires a rather long calculation, which we decided to skip since this section is only of theoretical interest, anyway. Indeed, we have reduced the general approximate common divisor problem, to the problem of solving an equation over the integers in two unknowns, which, in general, we cannot solve.

5 Conclusion

In Sections 3 and 4, we have discussed a continued fraction-based as well as a lattice-based approach to approximate common divisor problems, which resulted in three algorithms that compute solutions to three specific types of the approximate common divisor problem. Before concluding this paper, we briefly compare Algorithms 3.10, 3.19 and 4.20.

The principal advantage of the continued fraction-based over the lattice-based algorithms lies in their complexity: Both Algorithm 3.10 and 3.19 require $O((\log b)^3)$ bit operations to compute the approximate common divisors of two natural numbers $a$ and $b$, according to Remarks 3.14 and 3.22. In contrast, the Lattice-Based PACDP Algorithm has a running time of $O((n + l)^9(\log b)^3)$ bit operations for $n + l$ given by

$$n + l = \left\lfloor \frac{\log_b M (1 - \log_b M)}{(\log_b M)^2 - \log_b X} \right\rfloor$$

where $M$ and $X$ are the bounds on the approximate common divisors and their noise, respectively. Setting $X$ equal to $M^2/2b$ in order to compare the
complexities of the two partially approximate common divisor algorithms, we infer the following approximate value for the logarithm of $X$:

$$\log_b X = 2 \log_b M - \log_b 2b \approx 2 \log_b M - 1,$$

which yields

$$n + l \leq \frac{\log_b M (1 - \log_b M)}{(\log_b M)^2 - \log_b X} \approx \frac{\log_b M (1 - \log_b M)}{(1 - \log_b M)^2} = \frac{\log_b M}{1 - \log_b M}.$$ 

As $M$ is less than $b$, we thus have that $n + l$ lies in $O(1/(1 - \log_b M))$. Therefore, the lattice-based algorithm uses more bit operations than the continued fraction-based algorithm by a factor of $O(1/(1 - \log_b M)^9)$ to solve comparable problems.

An additional, rather obvious, disadvantage of the lattice-based method is that we are unable to establish a general approximate common divisor algorithm. However, this drawback might not be inherent to the approach. An advantage of the lattice-based technique, which is inherent, is that the bound $X$ may be chosen independently of the input $b$.

In conclusion, the continued fraction-based algorithm has a shorter running time, while the lattice-based technique can be applied to a wider range of parameters.

References


A  Appendix: The LLL-Algorithm

The Lenstra Lenstra and Lovász’ basis reduction algorithm - or in brief the LLL-Algorithm - efficiently computes a so-called reduced basis of a lattice Λ in \( \mathbb{Z}^n \), given an arbitrary basis of Λ, which in turn allows us to find a short nonzero vector in Λ. In fact, we will see that the algorithm returns a vector in the lattice Λ whose norm is bounded by

\[
2^{(n-1)/2} \sqrt{n(\det \Lambda)^{1/n}}.
\]

It might not seem particularly satisfactory to study an algorithm that computes a short but not necessarily the shortest nonzero vector of a lattice; however, the problem of finding the shortest vector of a general lattice can be shown to be NP-hard. Therefore, we must content ourselves with an efficient algorithm which computes a vector whose norm satisfies an upper bound independent of the given basis.

Before turning our attention to reducing bases of integer lattices, we briefly clarify the relevant terminology.

**Definition A.1.** Let \( v_1, \ldots, v_n \) be \( n \) linearly independent vectors in \( \mathbb{R}^m \). Then the \( \mathbb{Z} \)-module spanned by \( v_1, \ldots, v_n \) is called the lattice generated by \( v_1, \ldots, v_n \), which we denote by \( \Lambda(v_1, \ldots, v_n) \) or simply \( \Lambda \). More concretely,

\[
\Lambda(v_1, \ldots, v_n) = \left\{ \sum_{i=1}^{n} \lambda_i v_i : \lambda_1, \ldots, \lambda_n \in \mathbb{Z} \right\} \subset \mathbb{R}^m.
\]

Any set of linearly independent vectors in \( \mathbb{R}^m \) that generate a lattice \( \Lambda \) is called a basis of \( \Lambda \). We say that a matrix \( A \) generates a lattice \( \Lambda \) if its row vectors form a basis of \( \Lambda \). A lattice is said to have full rank if it is generated by \( n \) vectors in \( \mathbb{R}^n \).

One can associate a real number to any lattice \( \Lambda \) in \( \mathbb{R}^m \).

**Definition A.2.** Let \( A \) be a matrix that generates a lattice \( \Lambda \) in \( \mathbb{R}^m \). The determinant or volume of \( \Lambda \) is given by

\[
\det \Lambda = \sqrt{\det(AA^T)}
\]

It can be shown that the volume of a lattice is well defined ([6], 462). Furthermore, if a lattice \( \Lambda \) is of full rank, that is to say, if it is generated by a square matrix, say \( A \), the volume of \( \Lambda \) is equal to the absolute value of the determinant of \( A \). Indeed,

\[
\det \Lambda = \sqrt{\det(AA^T)} = \sqrt{(\det A)^2} = |\det A|.
\]

In order to define the notion of a reduced basis of a lattice, we recall a version of Gram-Schmidt orthogonalization, leaving the proof that algorithm below is correct to the reader.
Algorithm A.3 (Gram-Schmidt).

**Input:** Linearly independent vectors \( v_1, \ldots, v_n \) in \( \mathbb{R}^m \) and an inner product \( \langle \cdot, \cdot \rangle \) on \( \mathbb{R}^m \).

**Output:** Orthogonal vectors \( v_1^*, \ldots, v_n^* \) in \( \mathbb{R}^m \) so that \( v_1^*, \ldots, v_i^* \) and \( v_1, \ldots, v_i \) span the same subspace of \( \mathbb{R}^m \) for all \( i \).

**Step 1:** For \( i = 1, \ldots, n \)

\[
v_i^* = v_i
\]

**Step 2:** For \( i = 2, \ldots, n \)

For \( j = 1, \ldots, i - 1 \)

\[
\mu_{ij} = \frac{\langle v_i, v_j^* \rangle}{\langle v_j^*, v_j^* \rangle}
\]

\[
v_i^* = v_i^* - \mu_{ij} v_j^*
\]

**Step 3:** Return \( v_1^*, \ldots, v_n^* \)

**Remark A.4.** We note for later reference that the vector \( v_i^* \) is the projection of \( v_i \) onto the orthogonal complement of the subspace of \( \mathbb{R}^m \) generated by \( v_1, \ldots, v_{i-1} \). In particular,

\[
\| v_i^* \| \leq \| v_i \| \tag{A.1}
\]

for all \( 1 \leq i \leq n \). Indeed, we have that \( v_i = v_i^* + w \) for some vector \( w \) in the span of \( v_1, \ldots, v_{i-1} \), and hence

\[
\| v_i \|^2 = \| v_i^* + w \|^2 = \| v_i^* \|^2 + \| w \|^2 \geq \| v_i^* \|^2,
\]

using the fact that \( v_i^* \) is orthogonal to \( w \).

From now on, we restrict our attention to the standard inner product on \( \mathbb{R}^m \), that is to say, the inner product of any two vectors \( v = (v_1, \ldots, v_m) \) and \( w = (w_1, \ldots, w_m) \) in \( \mathbb{R}^m \) is given by \( \langle v, w \rangle = v_1 w_1 + \cdots + v_m w_m \), and in consequence, the induced norm of a vector \( v \) coincides with its Euclidean norm. By definition, Gram-Schmidt bases are orthogonal with respect to the standard inner product from here onwards.

**Definition A.5.** Let \( n \) linearly independent vectors \( v_1, \ldots, v_n \) in \( \mathbb{R}^m \) generate a lattice \( \Lambda \) and let \( v_1^*, \ldots, v_n^* \) be the corresponding Gram-Schmidt basis. The basis \( v_1, \ldots, v_n \) of \( \Lambda \) is called reduced if \( \| v_i^* \|^2 \leq 2 \| v_{i+1}^* \|^2 \) for all \( 1 \leq i \leq n-1 \).

The LLL-Algorithm will allow us to explicitly construct a reduced basis of any given lattice \( \Lambda(v_1, \ldots, v_n) \) provided that it is a subset of \( \mathbb{Z}^m \). Before presenting the algorithm, however, we will establish the connection between reduced bases and short vectors, which constitutes the main reason for studying basis reduction algorithms in cryptography.

For the moment, we will only consider lattices of full rank and generalize our findings to lattices of arbitrary rank later on.
Proposition A.6. If the vectors $v_1, \ldots, v_n$ in $\mathbb{R}^n$ form a reduced basis of a lattice $\Lambda$, then

$$||v_1|| \leq 2^{(n-1)/2} \min\{||v|| : v \in \Lambda \setminus \{0\}\}.$$ 

In other words, Proposition A.6 states that the norm of the first element of a reduced basis of a lattice is minimal up to a factor which does not depend on the choice of generators. The proof of the above proposition is based on the following lemma:

Lemma A.7. Let $\Lambda$ be a lattice generated by linearly independent vectors $v_1, \ldots, v_n$ in $\mathbb{R}^n$ and let $v_1^*, \ldots, v_n^*$ be the induced Gram-Schmidt basis. Then the following inequality holds:

$$\min\{||v|| : v \in \Lambda \setminus \{0\}\} \geq \min\{||v_1^*||, \ldots, ||v_n^*||\}.$$ 

Notice that in general the Gram-Schmidt basis corresponding to a basis of a lattice $\Lambda$ is not a subset of $\Lambda$.

Proof. It is sufficient to show that the norm of an arbitrary nonzero vector $v$ in $\Lambda(v_1, \ldots, v_n)$ is greater than the norm of $v_i^*$ for some $1 \leq i \leq n$.

By definition, there exists a nonzero element $(\lambda_1, \ldots, \lambda_n)$ in $\mathbb{Z}^n$ with the property that $v = \lambda_1 v_1 + \cdots + \lambda_n v_n$. Supposing $i$ to be the highest index such that $\lambda_i$ does not vanish, the vector $v$ lies in the subspace generated by the first $i$ elements of the corresponding Gram-Schmidt basis, i.e. there exist $v_1, \ldots, v_i$ in $\mathbb{R}$ so that $v_1 v_i^* + \cdots + v_i v_i^* = v$. In addition, the coefficients $\nu_i$ and $\lambda_i$ coincide since $v_i^*$ is the projection of $v_i$ onto the orthogonal complement of the space spanned by $v_1, \ldots, v_{i-1}$.

Using the orthogonality of $v_1^*, \ldots, v_i^*$ and the fact that $\lambda_i$ is an nonzero integer, we therefore conclude that

$$||v||^2 = \left(\sum_{j=1}^{i-1} \nu_j v_j^* + \lambda_i v_i^*\right)^2 = \left(\sum_{j=1}^{i-1} \nu_j v_j^*\right)^2 + ||\lambda_i v_i^*||^2 = \lambda_i^2 ||v_i^*||^2 \geq ||v_i^*||^2,$$

as desired.

The proof of Proposition A.6 now follows immediately.

Proof. As $v_1^*$ is equal to $v_1$, we may infer from the definition of a reduced basis that

$$||v_1||^2 = ||v_1^*||^2 \leq 2||v_2^*||^2 \cdots \leq 2^{n-1}||v_n^*||^2,$$

According to lemma A.7, we thus have that

$$2^{(n-1)/2} \min\{||v|| : v \in \Lambda \setminus \{0\}\} \geq 2^{(n-1)/2} \min\{||v_1^*||, \ldots, ||v_n^*||\} \geq ||v_1||.$$

\qed
Given a lattice \( \Lambda \) in \( \mathbb{R}^n \), Proposition A.6 provides an upper bound on the norm of the first vector in a reduced basis in terms of the dimension \( n \) and the length of a shortest nonzero vector in \( \Lambda \). Since we cannot in general determine the minimal length of a nonzero vector in \( \Lambda \), we cannot give the exact value of this bound. Minkowski’s Theorem will allow us to infer a slightly weaker upper bound on the norm of the first vector in a reduced basis, whose value can easily be computed. In order to state the theorem, we need the notion of a symmetric subset of \( \mathbb{R}^n \): A subset \( C \) of \( \mathbb{R}^n \) is called symmetric if the vector \( -v \) lies in \( C \) whenever \( v \) is an element of \( C \).

**Theorem A.8 (Minkowski).** Let \( \Lambda \) in \( \mathbb{R}^n \) be a lattice of full rank. Let \( C \) be a convex, compact and symmetric subset of \( \mathbb{R}^n \) with (Lebesgue) measure \( \text{vol} \, C \). If \( \text{vol} \, C \geq 2^n \, \det \, \Lambda \), the intersection of \( C \) and \( \Lambda \setminus \{0\} \) is not empty.

Its proof is beyond the scope of this text as it requires some background in algebraic number theory. It can be found in [4], page 261.

**Corollary A.9.** If the vectors \( v_1, \ldots, v_n \) are a reduced basis of a lattice \( \Lambda \) in \( \mathbb{R}^n \), the first basis element satisfies

\[
||v_1|| \leq 2^{(n-1)/2} \sqrt{n}(\det \Lambda)^{1/n}.
\]

(A.2)

**Proof.** Consider the \( n \)-dimensional cube \( C \) given by

\[
C = \left[ -(\det \Lambda)^{1/n}, (\det \Lambda)^{1/n} \right] \times \cdots \times \left[ -(\det \Lambda)^{1/n}, (\det \Lambda)^{1/n} \right].
\]

Clearly, \( C \) is a convex, compact and symmetric subset of \( \mathbb{R}^n \) of measure \( \text{vol} \, C = 2^n \, \det \, \Lambda \). By Minkowski’s Theorem, \( C \) therefore contains a nonzero element of the lattice \( \Lambda \). As the norm of any vector in the cube \( C \) is bounded by \( \sqrt{n}(\det \Lambda)^{1/n} \), the minimal length of a nonzero vector in the lattice \( \Lambda \) is at most \( \sqrt{n}(\det \Lambda)^{1/n} \). Using Proposition A.6, we conclude that

\[
||v_1|| \leq 2^{(n-1)/2} \min\{||v|| : v \in \Lambda \setminus \{0\}\} \leq 2^{(n-1)/2} \sqrt{n}(\det \Lambda)^{1/n}.
\]

\(\square\)

In the remainder of this section we discuss the LLL-Algorithm, which computes a reduced basis from any given basis of a lattice, and thus automatically returns a short vector of the lattice.

For this algorithm we use the notation that for any \( \mu \) in \( \mathbb{R} \), \( \lfloor \mu \rfloor \) denotes the integer nearest to \( \mu \). By convention, numbers of the form 0.5 are rounded up to avoid ambiguity.

**Algorithm A.10 (LLL).**

**Input:** \( \mathbb{R} \)-linearly independent vectors \( v_1, \ldots, v_n \in \mathbb{Z}^n \).

**Output:** A reduced basis \( b_1, \ldots, b_n \) of the lattice generated by \( v_1, \ldots, v_n \).
Step 1: For $i = 1, \ldots, n$
\[ b_i = v_i \]

Step 2: \{Gram-Schmidt orthogonalization\}
For $i = 1, \ldots, n$
\[ b_i^* = b_i \]
For $i = 2, \ldots, n$
For $j = 1, \ldots, i - 1$
\[ \mu_{ij} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \]
\[ b_i^* = b_i^* - \mu_{ij} b_j^* \]

$i = 2$
While $i \leq n$

Step 3: \{replacement step\}
For $j = i - 1, i - 2, \ldots, 1$
\[ b_i = b_i - [\mu_{ij}] b_j \]
For $k = 1, \ldots, j - 1$
\[ \mu_{ik} = \mu_{ik} - [\mu_{ij}] \mu_{jk} \]
\[ \mu_{ij} = \mu_{ij} - [\mu_{ij}] \]

Step 4: \{checking step\}
If $i = 1$ or $||b_{i-1}^*||^2 \leq 2||b_i^*||^2$
\[ i = i + 1 \]

Step 5: \{exchange step\}
Else
Exchange $b_{i-1}$ and $b_i$
\[ \tilde{b}_{i-1}^* = b_i^* + \mu_{i-1} b_{i-1}^* \]
For $j = 1, \ldots, i - 2$
Exchange $\mu_{ij}$ and $\mu_{i-1j}$
\[ \mu_{i-1j} = \frac{\langle b_i, \tilde{b}_j^* \rangle}{\langle \tilde{b}_j^*, \tilde{b}_j^* \rangle} \]
\[ b_i^* = b_i^* - \mu_{i-1} b_{i-1}^* \]
\[ b_{i-1}^* = \tilde{b}_{i-1}^* \]
For $j = i + 1, \ldots, n$
\[ \mu_{ji} = \frac{\langle b_j, b_{i-1}^* \rangle}{\langle b_i^*, b_{i-1}^* \rangle} \]
\[ \mu_{ji} = \frac{\langle b_j, b_{i-1}^* \rangle}{\langle b_i^*, b_{i-1}^* \rangle} \]
\[ i = i - 1 \]
Step 6: Return $b_1, \ldots, b_n$

Before turning our attention to the overall correctness of the algorithm, we verify that after each modification to the basis $b_1, \ldots, b_n$ its Gram-Schmidt basis $b_1^*, \ldots, b_n^*$ is altered accordingly.

**Lemma A.11.** After each execution of Step 2, Step 3, Step 4 and Step 5 the following statements hold:

(i) The vectors $b_1, \ldots, b_n$ form a basis of the lattice $\Lambda$ generated by $v_1, \ldots, v_n$.

(ii) The Gram-Schmidt basis corresponding to $b_1, \ldots, b_n$ is given by $b_1^*, \ldots, b_n^*$.

(iii) For all indices $1 \leq j < i \leq n$, the $\mu_{ij}$ satisfy $b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j^*$.

**Proof.** As Step 2 of the algorithm consists of the Gram-Schmidt orthogonalization of the basis $v_1, \ldots, v_n$, there is nothing to verify.

In order to show that all three statements remain true during each execution of Step 3, we fix an index $j < i$ and set $\breve{b}_i = b_i - [\mu_{ij}] b_j$, $\tilde{b}_k = b_k$ for $k \neq i$. Since $[\mu_{ij}]$ is an integer, the lattices generated by $b_1, \ldots, b_n$ and $\breve{b}_1, \ldots, \breve{b}_n$ coincide, which proves (i). Statement (ii) is a direct consequence of the fact that both bases induce the same Gram-Schmidt basis: In case $k \neq i$, $b_k^*$ and $\tilde{b}_k^*$ agree because $b_1, \ldots, b_{k-1}$ and $\breve{b}_1, \ldots, \breve{b}_{k-1}$ span the same subspace of $\mathbb{R}^n$ and $b_k$ is equal to $\tilde{b}_k$. Furthermore, the projection onto the orthogonal complement of span${b_1, \ldots, b_{i-1}} = \text{span}{\breve{b}_1, \ldots, \breve{b}_{i-1}}$ maps $b_i$ and $\breve{b}_i = b_i - [\mu_{ij}] b_j$ to the same element as $j$ is strictly less than $i$, which entails that $b_i^* = b_i^*$. Recalling that by construction of the Gram-Schmidt basis

$$
\begin{pmatrix}
1 & 0 & \cdots \\
\vdots & \ddots & \vdots \\
\mu_{n1} & \cdots & 1
\end{pmatrix}
\begin{pmatrix}
b_1^* \\
\vdots \\
b_n^*
\end{pmatrix}
$$

(A.3)
the following calculation shows statement (iii):

\[
\begin{pmatrix}
\tilde{b}_1 \\
\vdots \\
\tilde{b}_i \\
\vdots \\
\tilde{b}_n
\end{pmatrix}
\begin{pmatrix}
\tilde{b}_1^* \\
\vdots \\
\tilde{b}_i^* \\
\vdots \\
\tilde{b}_n^*
\end{pmatrix}^{-1}
= 
\begin{pmatrix}
0 \\
\vdots \\
\mu_{n1} 
\end{pmatrix}
- 
\begin{pmatrix}
\mu_{ij}
\end{pmatrix}
\begin{pmatrix}
b_1 \\
\vdots \\
b_n
\end{pmatrix}
= 
\begin{pmatrix}
\mu_{j1} \\
\vdots \\
\mu_{j,i-1} 
\end{pmatrix}
\begin{pmatrix}
e_i^t
\end{pmatrix}
\begin{pmatrix}
0 \\
\vdots \\
1
\end{pmatrix}
\]

where \(e_i\) stands for the \(i\)-th unit row vector.

Given that Step 4 only consists of relabelling indices, the statements remain true.

For any index \(2 \leq i \leq n\), executing Step 5 entails exchanging the basis vectors \(b_i\) and \(b_{i-1}\). Denote the basis of \(\Lambda\) we thus obtain by \(\hat{b}_1, \ldots, \hat{b}_n\). By the same argumentation as used in the justification of Step 3, we deduce that \(b_j^*\) is equal to \(\tilde{b}_j^*\) for \(j \neq i, i-1\). Hence, for all \(j \leq i-2\), the coefficients \(\hat{\mu}_{ij}\) and \(\hat{\mu}_{i-1,j}\) corresponding to the new Gram-Schmidt basis are given by \(\mu_{i-1,j}\) and \(\mu_{ij}\), respectively. Therefore,

\[
\tilde{b}_{i-1}^* = b_i^* + \mu_{i-1,i} b_{i-1}^* \quad \text{and} \quad \text{in turn} \quad b_i^* = b_{i-1}^* - \hat{\mu}_{i-1,i} \hat{b}_{i-1}^*,
\]

as desired. \(\square\)

By Lemma A.11, the correctness of the LLL-Algorithm is a direct consequence of the fact that it terminates. It is therefore sufficient to show that only finitely many exchange step may occur. To this end, we exhibit a value \(D\) having the properties that it is always a positive integer and that it does not change during the algorithm except that at each exchange step it decreases by at least a factor \(3/4\). At any point after Step 1 of the algorithm,
we may consider the determinant

\[ d_k = \det \begin{pmatrix} b_1 & b_1^t \\ \vdots & \vdots \\ b_k & b_k^t \end{pmatrix} \]

for all \(1 \leq k \leq n\). Letting \(D\) be the product of \(d_1\) up to \(d_{n-1}\), the following two lemmas show that it possesses the desired properties.

**Lemma A.12.** For \(1 \leq k \leq n\), the following equality holds:

\[ d_k = \prod_{l=1}^{k} ||b_l^*||^2. \]

*In particular, \(D\) is a strictly positive integer.*

**Proof.** By Formula A.3, we have that

\[
\begin{align*}
  d_k &= \det \begin{pmatrix} b_1 & b_1^t \\ \vdots & \vdots \\ b_k & b_k^t \end{pmatrix} \\
  &= \det \begin{pmatrix} 1 & 0 \\ \vdots & \ddots \\ \mu_{k1} & \ldots & 1 \\
  b_1^* & \ddots & \vdots \\
  b_k^* & \ddots & \mu_{k1} & \ldots & 1 \\
  1 & 0 \\ & & & & 0 \\
\end{pmatrix} \\
  &= \det \begin{pmatrix} 1 & 0 \\ & & & & 0 \\
\end{pmatrix} \det \begin{pmatrix} ||b_1^*||^2 & 0 \\ & \ddots & \vdots \\
0 & ||b_k^*||^2 & \mu_{k1} & \ldots & 1 \\
\end{pmatrix} \\
  &= ||b_1^*||^2 \cdots ||b_k^*||^2
\end{align*}
\]

where we used the orthogonality of \(b_1^*, \ldots, b_k^*\).

It now follows immediately that \(D\) is a positive integer. Indeed, no vector in the Gram-Schmidt basis equals zero, and hence any \(d_k\) is a product of positive real numbers and so is \(D\). In addition, each \(d_k\) is integral, being the determinant of a matrix with integer entries.

**Lemma A.13.** In Steps 2, 3 and 4 the value of \(D\) does not change, while the value of \(D\) is decreased by at least a factor \(3/4\) in Step 5.

**Proof.** According to Lemma A.11, the corresponding Gram-Schmidt basis is not altered by any of the modifications which are made to \(b_1, \ldots, b_n\) in Steps 2, 3 and 4. In combination with Lemma A.12, this shows the first part of the statement.
To show the second part, we claim that for all indices $k \neq i - 1$ the value of $d_k$ is not altered when $b_{i-1}$ and $b_i$ are exchanged, whereas the value of $d_{i-1}$ decreases by at least a factor $3/4$. Indeed, if $k \neq i - 1$, exchanging the two basis vectors is equivalent to multiplying the matrix whose determinant equals $d_k$ by a permutation matrix form the left and its transpose form the right, which does not affect the determinant $d_k$.

Let $\tilde{d}_{i-1}$ denote the value of $d_{i-1}$ corresponding to the basis $\tilde{b}_1, \ldots, \tilde{b}_n$ obtained by exchanging the basis vectors $b_i$ and $b_{i-1}$. By Lemma A.12, $\tilde{d}_{i-1}$ is given by the product $||\tilde{b}_1^*||^2 \cdots ||\tilde{b}_{i-1}^*||^2$, which is equal to $||b_1^*||^2 \cdots ||b_{i-2}^*||^2$, $||\tilde{b}_{i-1}^*||^2$, according to Lemma A.11. It therefore remains to show that $||\tilde{b}_{i-1}^*||^2$ is less than $3/4||b_{i-1}^*||^2$. Lemma A.11 also entails that

$$||\tilde{b}_{i-1}^*||^2 = ||b_i^* + \mu_{i-1}b_{i-1}^*||^2 = ||b_i^*||^2 + \mu_{i-1}^2 ||b_{i-1}^*||^2,$$

using the the fact that $b_i^*$ and $b_{i-1}^*$ are orthogonal. As one condition for the application of the exchange step is that $||b_{i-1}^*||^2 > 2||b_i^*||^2$, we may infer that

$$||\tilde{b}_{i-1}^*||^2 < (1/2 + \mu_{i-1}^2)||b_{i-1}^*||^2.$$

Given that in Step 3, $\mu_{i-1}$ has been replaced by $\mu_{i-1} - [\mu_{i-1}]$ whose norm is less than $1/2$, we conclude that $||\tilde{b}_{i-1}^*||^2 < 3/4||b_{i-1}^*||^2$. \hfill $\square$

**Proposition A.14.** The LLL-Algorithm is correct; in particular, it terminates.

**Proof.** We have seen that the algorithm is correct provided that it terminates. Clearly, the algorithm terminates if the number of passes through the while loop - which encompasses Steps 3, 4 and 5 - is finite, or equivalently, if it only passes through Step 5 finitely many times.

Let $D(t)$ denote the value of $D$ after the $t$-th execution of Step 5. Combining Lemma A.12 and A.13, yields that for any counter $t$, $D(t)$ is greater than 1 and satisfies $D(t) \leq 3/4D(t - 1)$, form which we deduce that the number of counters must be finite. \hfill $\square$

Refining the argumentation used to prove that the LLL-Algorithm terminates, will allow us to determine its complexity.

**Lemma A.15.** The LLL-Algorithm with input $v_1, \ldots, v_n$ in $\mathbb{Z}^n$ uses $O(n^4 \log C)$ operations in $\mathbb{Z}$ where $C$ bounds the norm of the input vectors from above.

**Proof.** We first determine the maximally possible number of passes through the while loop, and then compute the cost of each step separately.

The proof of Proposition A.14 shows that the number of exchanges is at most $\log_{4/3} D(0)$. By Lemma A.12 and Remark A.4, $D(0)$ satisfies

$$D(0) = ||v_1^*||^{2(n-1)}||v_2^*||^{2(n-2)} \cdots ||v_{n-1}^*||^2 \leq ||v_1||^{2(n-1)}||v_2||^{2(n-2)} \cdots ||v_{n-1}||^2 \leq C^{n(n-1)};$$
implying that at most $n(n-1)\log_{4/3} C$ exchanges are made in the course of the algorithm. As the value of $i$ is increased by one in Step 4 and decreased by one in Step 5, a bound on the number of passes through the while loop may be obtained by adding twice the number of exchanges to the difference between the initial and final value of $i$, resulting in $n-1+2n(n-1)\log_{4/3} C$, which simplifies to $O(n^2 \log C)$ passes.

A somewhat technical calculation, which we leave to the reader, shows that the Gram-Schmidt orthogonalization in Step 2 requires $O(n^3)$ operations in $\mathbb{Z}$. Replacing a basis vector $b_i$ by $b_i - [\mu_{ij}]b_j$ and subsequently adjusting the Gram-Schmidt coefficients takes $O(n)$ operations in $\mathbb{Z}$, and thus one execution of Step 3 necessitates $O(n^2)$ operations. Computing the norms of the vectors considered in Step 4, also takes $O(n)$ operations in $\mathbb{Z}$. Finally, the necessary modifications to the Gram-Schmidt basis after exchanging two basis vectors require $O(n^2)$ operations.

Therefore, one iteration of the while loop comes at the cost of $O(n^2)$ operations. Given that the number of passes through the while loop is at most $O(n^2 \log C)$, we conclude that the total number of operations needed is $O(n^3) + O(n^4 \log C) = O(n^4 \log C)$. \hfill \Box

The above lemma does not provide us with an estimate of the cost of the LLL-Algorithm as it makes no mention of the size of the integers on which the algorithm performs its various operations. In a next step, we therefore bound the size of all integers on which the algorithm operates in relation to $C$. To this end, we first establish some general facts about the Gram-Schmidt orthogonalization of a set of linearly independent vectors with integer entries.

**Lemma A.16.** Let $b_1, \ldots, b_n$ in $\mathbb{Z}^n$ be linearly independent vectors over the field of real numbers. Using the same notation as above, consider $d_i^*, \mu_{ij}$ for all indices $1 \leq j < i \leq n$. Setting $d_0 = 1$, the following statements hold:

(i) All entries of the vector $d_{i-1}b_i^*$ are integers.

(ii) The product $d_j\mu_{ij}$ is an integer.

(iii) The absolute value of $\mu_{ij}$ is less than $\sqrt{d_{j-1}\|b_i\|}$.

**Proof.** (i) By construction of the Gram-Schmidt basis, there exist real numbers $\nu_1, \ldots, \nu_i$ so that $b_i^* = b_i - \sum_{k=1}^{i-1} \nu_{ik} b_k$. We infer that, for indices all $j < i - 1$,

$$\langle b_j, b_i \rangle = \langle b_j, b_i^* + \sum_{k=1}^{i-1} \nu_{ik} b_k \rangle = \sum_{k=1}^{i-1} \nu_{ik} \langle b_j, b_k \rangle,$$
which yields
\[
\begin{pmatrix}
\langle b_1, b_i \rangle \\
\vdots \\
\langle b_{i-1}, b_i \rangle
\end{pmatrix}
= 
\begin{pmatrix}
\langle b_1, b_1 \rangle & \ldots & \langle b_1, b_{i-1} \rangle \\
\vdots & \ddots & \vdots \\
\langle b_{i-1}, b_1 \rangle & \ldots & \langle b_{i-1}, b_{i-1} \rangle
\end{pmatrix}
\begin{pmatrix}
\nu_{i1} \\
\vdots \\
\nu_{i(i-1)}
\end{pmatrix}
= 
\begin{pmatrix}
b_1 \\
\vdots \\
b_{i-1}
\end{pmatrix}
\begin{pmatrix}
b_1 \\
\vdots \\
b_{i-1}
\end{pmatrix}^t
\begin{pmatrix}
\nu_{i1} \\
\vdots \\
\nu_{i(i-1)}
\end{pmatrix}.
\]

Applying Cramer’s rule to compute the inverse ([2], 205), we thus see that \(d_{i-1}\nu_{ik}\) is integral for all indices \(k \leq i - 1\), from which we conclude that \(d_{i-1}b_i^* = d_{i-1}b_i - \sum_{k=1}^{i-1} d_{i-1}\nu_{ik}b_k\) lies in \(\mathbb{Z}^n\).

(ii) According to Lemma A.12, \(d_j/\|b_j^*\|^2 = d_{j-1}\) for all \(1 \leq j \leq n\), and hence
\[
d_j\mu_{ij} = d_j \frac{\langle b_i, b_j^* \rangle}{\|b_j^*\|^2} = d_{j-1} \langle b_i, b_j^* \rangle = \langle b_i, d_{j-1}b_j^* \rangle,
\]
which is an integer by statement (i).

(iii) Recalling the Cauchy-Schwarz inequality ([2], 275), we conclude that
\[
|\mu_{ij}| = \frac{|\langle b_i, b_j^* \rangle|}{\|b_j^*\|^2} \leq \frac{\|b_i\| \cdot \|b_j^*\|}{\|b_j^*\|^2} = \frac{\|b_i\|}{\|b_j^*\|} \leq d_{j-1}^{1/2} \|b_i\|
\]
where we used another consequence of Lemma A.12, namely that \(\|b_i^*\|^2 = d_i/d_i - 1 \geq 1/d_i - 1\).

\[\square\]

Lemma A.17. Let \(C\) be an upper bound on the norm of the vectors \(v_1, \ldots, v_n\) in \(\mathbb{Z}^n\). At any stage of the LLL-Algorithm with input \(v_1, \ldots, v_n\) and for any index \(1 \leq i \leq n\), we have
\[
\|b_i^*\| \leq C \text{ and thus } d_i \leq C^{2i}.
\]

Proof. The Gram-Schmidt basis induced by \(v_1, \ldots, v_n\) consists of vectors whose norm is less than \(C\) as for any index \(i\), \(\|v_i^*\|\) is less than \(\|v_i\|\), according to Remark A.4. Given that the Gram-Schmidt basis is only modified in the exchange step of the algorithm, it is therefore sufficient to verify that
\[
\max\{\|\tilde{b}_1^*\|, \ldots, \|\tilde{b}_n^*\|\} \leq \max\{\|b_1^*\|, \ldots, \|b_n^*\|\}
\]
where the basis \(\tilde{b}_1, \ldots, \tilde{b}_n\) is obtained from \(b_1, \ldots, b_n\) by exchanging the \(i\)-th and the \((i - 1)\)-th element.
According to Lemma A.11, $\tilde{b}_j^*$ is equal to $b_j^*$ for all $j \neq i-1, i$, and we have seen during the proof of Lemma A.13 that $||\tilde{b}_{i-1}^*||$ is less than $\sqrt{3/4}||b_{i-1}^*||$. Lemma A.11 further implies that $b_{i-1}^* = \tilde{b}_i^* + \mu_{i,i-1} \tilde{b}_{i-1}^*$, form which we deduce that

$$||b_{i-1}^*||^2 = ||\tilde{b}_i^* + \mu_{i,i-1} \tilde{b}_{i-1}^*||^2 = ||\tilde{b}_i^*||^2 + ||\mu_{i,i-1} \tilde{b}_{i-1}^*||^2 \geq ||\tilde{b}_i^*||^2.$$

Therefore, inequality $\star$ is satisfied, and hence, $||b_i^*|| \leq C$ for all $i$. Together with Lemma A.12, this entails that $d_i = \prod_{k=1}^i ||b_k^*||^2 \leq C^{2i}$.

**Lemma A.18.** Let $v_1, \ldots, v_n$ in $\mathbb{Z}^n$ be input vectors into the LLL-Algorithm and consider a number $C$ so that $C \geq \max\{||v_1||, \ldots, ||v_n||\}$. For $1 \leq k \leq n$, we have that

$$||b_k|| \leq \sqrt{n}C \quad (\ast)$$

at any stage in the algorithm, except possibly during the replacement step when $k = i$, in which case the length of $b_i$ satisfies

$$||b_i|| \leq n(2C)^n.$$

**Proof.** By definition, we initially have that $C$ is greater than the norm of $b_k$ for all $k$. Given that $\max\{||b_1||, \ldots, ||b_n||\}$ remains unchanged in Steps 1, 2, 4 and 5 of the algorithm, we may focus on Step 3. However, unless $k = i$, the replacement step does not affect the vector $b_k$. Therefore, the first part of the lemma follows form the facts that $b_i$ satisfies inequality $\star$ immediately after Step 3. Indeed, we have that

$$||b_i||^2 = ||b_i^*||^2 + \sum_{j=1}^{i-1} \mu_{ij}^2 ||b_j^*||^2 \leq C^2 + \sum_{j=1}^{i-1} \mu_{ij}^2 C^2 \quad (\ast\ast)$$

by Lemma A.16, and setting $m_i = \max\{|\mu_{i1}|, \ldots, |\mu_{i-1}|, 1\}$

$$\leq m_i^2 nC^2 = nC^2$$

as at the end of the replacement step $|\mu_{ij}| \leq 1/2$ for all indices $j < i$.

Together with Lemma A.16 (iii) and A.17, this implies that at the beginning of the replacement step

$$m_i \leq \max\{\sqrt{d_{j-1}} : 1 \leq j \leq i-1\} ||b_i|| \leq C^{i-2} \sqrt{n}C \leq \sqrt{n}C^{n-1}$$

since $\sqrt{d_0} ||b_1||$ is certainly greater than 1. We now consider the replacement of $b_i$ by $\tilde{b}_i = b_i - [\mu_{ij}] b_j$ for some fixed index $j < i$. For all $1 \leq k < j$, we have that

$$|\tilde{\mu}_{ik}| = |\mu_{ik} - [\mu_{ij}]\mu_{jk}| \leq |\mu_{ik}| + ||[\mu_{ij}]|||\mu_{jk}|$$

$$\leq m_i + \left(m_i + \frac{1}{2}\right) \cdot \frac{1}{2} = \frac{3}{2} m_i + \frac{1}{4} \leq 2m_i$$
where we used the fact that $m_i \geq 1$ and $|\mu_{jk}| \leq 1/2$, by construction. Hence,

$$\hat{m}_i = \max\{|\hat{\mu}_{i1}|, \ldots, |\hat{\mu}_{i, i-1}|, 1\} \leq 2m_i.$$  

Given that Step 3 consists of at most $n - 1$ replacements of this type, the fact that immediately before Step 3 $m_i$ is less than $\sqrt{n}C_n^{-1}$ thus allows us to infer that during the execution of Step 3 the value of $m_i$ never exceeds $\sqrt{n}(2C)^{n-1}$.

By inequality $\star\star$, we therefore conclude that $||b_i|| \leq m_i\sqrt{n}C \leq n(2C)^n$ during the replacement step. 

**Proposition A.19.** The LLL-Algorithm computes a reduced basis of a lattice $\Lambda$ in $\mathbb{Z}^n$ generated by $n$ vectors $v_1, \ldots, v_n$ of norm less than $C$ in $O(n^6(\log C)^3)$ bit operations.

**Proof.** The three preceding lemmas will enable us to show that all integers which occur in the algorithm are of size $O(n \log C)$. By Lemma A.15, we may therefore infer that the LLL-Algorithm requires $O((n \log C)^2 \cdot n^4 \log C) = O(n^6(\log C)^3)$ bit operations, given that operations on integers of size $O(m)$ take $O(m^2)$ bit operations ([5], 66).

It remains to bound the size of any integer that occurs in the execution of the algorithm: By Lemma A.16 (i) and (ii) the denominator of any rational number which appears is less than $d_n - 1$, which is of size $O(n \log C)$, according to Lemma A.17. Let us now turn to the numerators: For all indices $i$,

$$||b_i|| \leq n(2C)^n$$

by Lemma A.18

and

$$||d_{i-1}b_i^*|| \leq d_{i-1}||b_i^*|| \leq C^{2(i-1)}C$$

by Lemma A.17.

Hence, $||b_i||$ and $||d_{i-1}b_i^*||$ are both of size $O(n \log C)$; in particular, all entries of $b_i$ and $d_{i-1}b_i^*$ are of size $O(n \log C)$. Finally, for $1 \leq j < i \leq n$, statements (ii) and (iii) of Lemma A.16 entail that the numerator of $\mu_{ij}$ is bounded by

$$|d_j\mu_{ij}| \leq d_j\sqrt{d_{i-1}||b_i||},$$

which, according to Lemmas A.17 and A.18, is less than

$$C^{2j}C^{j-1}n(2C)^n \leq n2^nC^{4n}.$$  

Therefore, the numerator of $\mu_{ij}$ is also of size $O(n \log C)$. 

In conclusion, given a basis $v_1, \ldots, v_n$ in $\mathbb{Z}^n$ of a lattice $\Lambda$ of full rank, we can compute a nonzero lattice element $v$ in $\Lambda$ whose norm satisfies

$$||v|| \leq 2^{(n-1)/2}\sqrt{n(\det \Lambda)^{1/n}},$$
using $O(n^6(\log C)^3)$ bit operations where $C$ is an upper bound on the norm of the basis elements $v_1, \ldots, v_n$. The principal difficulty in generalizing these findings to lattices of arbitrary rank lies in the fact that Minkowski’s Theorem A.8 solely holds for lattices of full rank. Proposition A.20 overcomes this problem.

**Proposition A.20.** If the vectors $v_1, \ldots, v_n$ form a reduced basis of a lattice $\Lambda$ in $\mathbb{R}^m$, the first basis element satisfies
\[
||v_1|| \leq 2^{(n-1)/2} \sqrt{n}(\det \Lambda)^{1/n}.
\]

**Proof.** We may assume without loss of generality that $m \geq n + 1$. The basic idea underlying this proof is to complement $v_1, \ldots, v_n$ to a basis of $\mathbb{R}^m$ by adding $m - n$ suitable vectors, say $w_1, \ldots, w_{m-n}$, and to apply Minkowski’s Theorem A.8 to the lattice $\tilde{\Lambda}$ generated by $v_1, \ldots, v_n, w_1, \ldots, w_{m-n}$.

For $i = 1, \ldots, m - n$, we recursively choose the vectors $w_i$ in $\mathbb{R}^m$ so that $w_i$ lies in the orthogonal complement to the subspace of $\mathbb{R}^m$ spanned by $v_1, \ldots, v_n$ and $w_1, \ldots, w_{i-1}$. Consider the $m$-dimensional cube $C$, which is compatible with the induced Gram-Schmidt basis, given by
\[
C = \left\{ \sum_{i=1}^{n} \alpha_i \frac{v_i^*}{||v_i^*||} + \sum_{j=1}^{m-n} \beta_j \frac{w_j}{||w_j||} \in \mathbb{R}^m : ||\alpha_i||, ||\beta_j|| \leq (\det \tilde{\Lambda})^{1/m} \right\}.
\]

As the collection $v_1^*/||v_1^*||, \ldots, v_n^*/||v_n^*||, w_1/||w_1||, \ldots, w_{m-n}/||w_{m-n}||$ is an orthonormal basis of $\mathbb{R}^m$, the volume of $C$ is equal to $2^m \det \tilde{\Lambda}$. By Minkowski’s Theorem A.8, there exists a nonzero vector $v$ in $\mathbb{R}^m$ in the intersection of the cube $C$ and the lattice $\tilde{\Lambda}$.

We now claim that the vector $v$ is an element of the lattice $\Lambda$, if $||w_j||$ is strictly greater than $(\det \tilde{\Lambda})^{1/m}$ for all indices $j$. Indeed, let us assume to the contrary that $v$ does not lie in $\Lambda$. Being an element of $\tilde{\Lambda} \setminus \Lambda$, it can be written as
\[
v = \sum_{i=1}^{n} \lambda_i v_i + \sum_{j=1}^{m-n} \nu_j w_j
\]
for some $(\lambda_1, \ldots, \lambda_n) \in \mathbb{Z}^n$ and $(\nu_1, \ldots, \nu_{m-n}) \in \mathbb{Z}^{m-n} \setminus \{0\}$, i.e. there exists an index $k$ such that $\nu_k \neq 0$. Hence, we infer that
\[
\nu_k ||w_k|| \geq ||w_k|| > (\det \tilde{\Lambda})^{1/m},
\]
which forms a contradiction to the fact that $v$ lies in the cube $C$. Therefore, the intersection of $C$ and $\Lambda \setminus \{0\}$ is not empty, provided that the norm of all $w_j$ is strictly greater than $(\det \tilde{\Lambda})^{1/m}$. 

In order to ensure that the basis vectors satisfy the above condition, we normalize the \( w_j \) so that 
\[
||w_j|| = \varepsilon^{m/(m-n)}(\det \Lambda)^{1/n}
\]
for a fixed \( \varepsilon > 1 \). Indeed, by Lemma A.12,
\[
\det \tilde{\Lambda} = \det \left( \begin{pmatrix} v_1 & \cdots & v_1 \\ \vdots & \ddots & \vdots \\ w_{m-n} & \cdots & w_{m-n} \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ w_1 \end{pmatrix} \right) = \prod_{i=1}^{n} ||v_i^*|| \prod_{i=j}^{m-n} ||w_j^*||
\]

\[
= \det \Lambda \prod_{i=j}^{m-n} ||w_j|| = \det \Lambda \left( \varepsilon^{m/(m-n)}(\det \Lambda)^{1/n} \right)^{m-n} = \varepsilon^m (\det \Lambda)^{m/n},
\]

and hence \((\det \tilde{\Lambda})^{1/m} = \varepsilon (\det \Lambda)^{1/n} < \varepsilon^{m/(m-n)}(\det \Lambda)^{1/n} = ||w_i||\) since \( \varepsilon \) is strictly greater than 1. Letting \( v \) be a nonzero vector in the intersection of \( \mathcal{C} \) and \( \Lambda \), we thus infer that
\[
||v|| \leq \sqrt{n}(\det \tilde{\Lambda})^{1/m} \leq \sqrt{n}(\varepsilon^m (\det \Lambda)^{m/n})^{1/m} = \varepsilon \sqrt{n}(\det \Lambda)^{1/n},
\]

where we used the fact that the norm of any element of the form \( v = \sum_{i=1}^n \alpha_i v_i^* / ||v_i^*|| \) in \( \mathcal{C} \) is less than \( \sqrt{n}(\det \tilde{\Lambda})^{1/m} \), by the triangle inequality.

As the proofs of Proposition A.6 and Lemma A.7 may be applied word for word to lattices of arbitrary rank, we conclude that for all \( \varepsilon > 1 \),
\[
2^{-(n-1)/2}||v_1|| \leq \min\{||w|| : w \in \Lambda \setminus \{0\}\} \leq ||v|| \leq \varepsilon \sqrt{n}(\det \Lambda)^{1/n},
\]

which proves the proposition since the vector \( v_1 \) is independent of \( \varepsilon \). \( \Box \)

It is straightforward to verify that the LLL-Algorithm A.10 works equally well when taking \( n \) linearly independent vectors \( v_1, \ldots, v_n \) in \( \mathbb{Z}^m \) as input for some natural number \( m > n \). In fact, non of the proofs leading up to the statement that the LLL-Algorithm terminates make use of the fact that only lattices of full rank are considered. An analogous argument may be applied to determine the complexity of this generalized version of the LLL-Algorithm; however, the result differs slightly. Let the real number \( C \) bound the norm of the input vectors \( v_1, \ldots, v_n \) in \( \mathbb{Z}^m \) form above.

**Proposition A.21.** Given \( n \) linearly independent vectors \( v_1, \ldots, v_n \) in \( \mathbb{Z}^m \), the LLL-Algorithm computes a reduced basis of the lattice \( \Lambda(v_1, \ldots, v_n) \), using \( O(mn^3 \log C) \) operations on integers of size \( O(n \log C) \). In particular, the LLL-Algorithm requires \( O(mn^5 (\log C)^3) \) bit operations.
Proof. We have already seen that the generalized LLL-Algorithm is correct, hence, it suffices to compute its complexity. Moreover, the proof of the fact that any integer which occurs in the execution of the algorithm is of size $O(n \log C)$ remains valid.

In fact, only Lemma A.15 needs to be adapted to the new circumstances. Given that there are still $O(n^2 \log C)$ passes through the while loop, which encompasses Steps 3, 4 and 5, it is enough to reevaluate the number of operations in $\mathbb{Z}$ each step takes: Step 2 requires $O(mn^2)$ integer operations. One replacement in Step 3 takes $O(m)$ operations, and thus the entire Step necessitates $O(mn)$ operations. Computing the norms to be compared in Step 4 comes at a cost of $O(m)$ operations, and finally, Step 5 also requires $O(mn)$ operations. Therefore, each pass through the while loop takes $O(mn)$ operations in $\mathbb{Z}$, which entails that the algorithm uses $O(mn^2) + O(n^2 \log C)O(mn) = O(mn^3 \log C)$ operations. \hfill $\square$