Boneh's Algorithm for Finding Strongly Smooth Numbers in an Interval

by

Palak Batra

A thesis
presented to the University of Waterloo
in fulfillment of the
research paper requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Waterloo, Ontario, Canada, 2024

© Palak Batra 2024

Author's Declaration

I declare hereby that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This research explores and implements Dan Boneh's algorithm for finding strongly smooth numbers within a given interval. It presents Boneh's original algorithm, its refined versions obtained using tighter estimates and parameter changes, as well as examples implemented in Maplesoft. It includes tables and graphs that provide some intuition for the applicability of Boneh's Theorem. Additionally, it supplies background material on smooth and strongly smooth numbers, along with the LLL algorithm, which is a vital component of Boneh's algorithm.

Acknowledgements

I am deeply grateful to my supervisor, Professor Michael Rubinstein, for his invaluable guidance, support, time, and encouragement throughout this research. His insights and patience have been the foundation of this work.

I owe immense thanks to my parents and my brother, who have always believed in me, often more than I believed in myself. Their unwavering support and love have been my greatest source of strength.

Finally, I want to thank my teachers who have shaped me, from childhood to today. Their lessons, guidance, and belief in me have left a lasting impact on my journey.

Dedication

This work is dedicated to my mum and papa, who have always dreamed of me and my brother achieving great things in life. Your relentless hard work, sacrifices, and unwavering belief in me have made everything possible. Thank you for making me capable of striving for my dreams.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The study of smooth numbers is fundamental in computational number theory, with important implications in areas such as cryptography, integer factorization, and lattice-based algorithms. Smooth numbers, defined as integers whose prime factors do not exceed a specified bound, have gained particular interest due to their relevance in algorithmic applications.

This thesis focuses on Boneh's algorithm, which provides a framework for finding strongly smooth numbers. To facilitate understanding of the algorithm and its workings, we implemented it in Maplesoft. The goal was to explore how the theorem operates and to find examples of smooth numbers under the conditions it specifies.

A significant part of this research is the refinement of Boneh's results by employing tighter numerical approximations and selecting improved and more suitable parameters.

The thesis is structured as follows.

Chapter 2: This chapter introduces the fundamental concepts of smooth and strongly smooth numbers, including their definition and properties, and discusses their distribution. These concepts lay the foundation for the subsequent theoretical and practical exploration.

Chapter 3: This chapter describes the Lenstra–Lenstra–Lovász(LLL) algorithm, which is integral to Boneh's theorem. It explains the mathematical foundation of the algorithm, its significance in lattice reduction, an example of the reduction process, and the practical details of its implementation.

Chapter 4: This chapter details Boneh's algorithm. It discusses the major results and theorems that helped in the development of the Maple code, outlines the refinements made to improve accuracy, and presents the results of various examples.

Chapter 5: We demonstrate the applicability of the theorem in detecting smooth numbers through some numerical plots and tables.

Chapter 2

Smooth numbers

Definition

Smooth numbers: A positive integer N is called s-smooth if none of its prime factors are greater than s. In other words, if

$$N = \Pi_1^n p_i$$

then N is s smooth if for all $i = \{1, 2 \ldots, n\}$ $p_i \leq s$.

Example: $N = 252 = 2^2 \times 3^2 \times 7$ is a $7-$smooth number.

Strongly smooth numbers (also called powersmooth numbers): An integer N is called s strongly smooth if N is s smooth and additionally, if p is a prime factor of N then the multiplicity of p as a divisor of N is at most $\frac{\log(s)}{\log(p)}$. In simple words, p^m should not divide N for any m such that $p^m > s$.

Example: $N = 13,392 = 2^4 \times 3^3 \times 31$ is a $31-$ strongly smooth number.

Observe that $N = 720 = 2^4 \times 3^2 \times 5$ is a $5-$smooth number but not a $5-$strongly smooth number, as $2^4 = 16 > 5$.

2.1 Distribution of Smooth numbers

[5] Let $\Psi(x, y)$ denote the number of $y-$ smooth numbers $n \leq x$.

Note that if $y \geq x$, all numbers less than y will have factors $\leq y$, implying that all positive integers $\leq x$ are $y-$ smooth

$$\psi(x, y) = [x] = x - \epsilon. \tag{2.1}$$

Here, ϵ represents the fractional part of x, where $0 \leq \epsilon < 1$.

Now, for $n \leq x$, n can have at most one prime factor $p \geq \sqrt{x}$ (otherwise, $n = p_1 \times p_2 \geq \sqrt{x} \times \sqrt{x} = x$).

Hence, for $\sqrt{x} \leq y \leq x$

$$\begin{aligned} \Psi(x, y) &= [x] - \sum_{y < p \leq x} \sum_{n \leq x, p|n} 1 \\ &= [x] - \sum_{y < p \leq x} [x/p] \\ &= x - x \sum_{y < p \leq x} 1/p + O(\pi(x)). \end{aligned} \tag{2.2}$$

By Chebyshev's elementary bounds for $\pi(x)$ and and Mertens' theorem

$$\sum_{p \leq x} 1/p = \log\log(x) + B + O(1/\log x), \tag{2.3}$$

where B is constant. Thus right hand side of (2.2) becomes

$$= x(1 - \log(\frac{\log(x)}{\log(y)})) - O(\frac{x}{\log(x)}). \tag{2.4}$$

Let $u = \frac{\log(x)}{\log(y)}$ which implies $y = x^{\frac{1}{u}}$, therefore

$$\Psi(x, x^{\frac{1}{u}}) = x(1 - \log(u)) - O(\frac{x}{\log(x)}). \tag{2.5}$$

There exist a function $\rho(u) > 0$ such that when $x \to \infty$ with u bounded,

$$\Psi(x, x^{\frac{1}{u}}) \sim \rho(u)x. \tag{2.6}$$

The function $\rho(u)$ here is called **Dickman function**, and it satisfies the following properties:

1. It can be defined as the unique continuous function on $[0, \infty)$ satisfying the differential–delay equation for $u > 1$:

$$u\rho'(u) = -\rho(u-1). \tag{2.7}$$

2. for $0 \leq u \leq 1$

$$\rho(u) = 1. \tag{2.8}$$

3. For $1 \leq u \leq v$

$$\rho(v) = \rho(u) - \int_u^v \frac{\rho(t-1)}{t}\, dt. \tag{2.9}$$

   This property can be derived from (2.7).

4. (2.7) can be written as $(u\rho(u))' = \rho(u) - \rho(u-1)$, hence integrating this for $u \geq 1$ we get,

$$u\rho(u) = \int_{u-1}^{u} \rho(v)\, dv + C. \tag{2.10}$$

   In particular, for $u = 1 \implies \rho(u) = 1$ in (2.10) gives value of constant of integration $C = 0$.

$$\therefore u\rho(u) = \int_{u-1}^{u} \rho(v)\, dv. \tag{2.11}$$

   for $u \geq 1$

5. $\rho(u)$ is a positive and decreasing function.

   Suppose u_0 is the infimum of the solutions to $\rho(u) = 0$. Then $\rho(u_0) = 0$ and due to the continuity of $\rho(u)$, we have $\rho(u) > 0$ for $0 \leq u < u_0$.

   Hence, substituting $u = u_0$ in (2.11), L.H.S becomes 0 and R.H.S remains positive-a contradiction.

   Thus, $\rho(u) > 0$ for all $u \geq 0$, and hence using (2.7) gives $\rho'(u) < 0$ for all $u > 1$, which means $\rho(u)$ is decreasing.

Another property of the function $\rho(u)$ is as follows:

2.2 Bounds on the Dickman Function

Let function $\rho(u)$ for $u \geq 0$ as defined above satisfying all above stated properties, then

$$\frac{1}{2\Gamma(2u+1)} \leq \rho(u) \leq \frac{1}{\Gamma(u+1)}. \tag{2.12}$$

Proof: At first, we will prove the upper bound using Induction on $0 \leq u \leq U$.

For base case $0 \leq u \leq 1$,

we know $\rho(u) = 1$ by (2.8). Thus, we aim to show $\Gamma(u+1) \leq 1$.

Observe that $\Gamma(1) = 1$ and $\Gamma(2) = 1$, and as $\Gamma(n+1) = \int_0^\infty e^{-x}x^n\,dx$,

$$\implies \Gamma''(n+1) = \int_0^\infty e^{-x}x^n(\log(x))^2\,dx \geq 0 \tag{2.13}$$

therefore surely, $\Gamma(u+1) \leq 1$ for $1 \leq u+1 \leq 2$.

Hence, the base case holds: $\rho(u) \leq \frac{1}{\Gamma(u+1)}$ for $0 \leq u \leq 1$.

Now assume that the result holds **for** $u \leq U$.

Before we prove for $U \leq u \leq U+1$, note that as $\rho(u)$ is decreasing function (in other words $\rho(u) < \rho(u-1)$) and by (2.11),

$$u\rho(u) = \int_{u-1}^{u} \rho(v)\,dv \leq \int_{u-1}^{u} \rho(u-1)\,dv = \rho(u-1),$$
$$\implies u\rho(u) \leq \rho(u-1). \tag{2.14}$$

Now, **for** $U \leq u \leq U+1$,

$$\rho(u) \leq^{(by\ (2.14))} \frac{\rho(u-1)}{u} \leq^{(\text{by induction hypothesis})} \frac{1}{u\Gamma(u)} = \frac{1}{\Gamma(u+1)}.$$

Thus, the upper bound holds for all $u \geq 0$.

Next, we prove the lower bound. It requires a variable substitution: $u = \frac{v}{2}$.

We need to prove:

$$\frac{1}{2\Gamma(v+1)} \leq \rho(v/2)$$

. We will use induction again.

Base case is $0 \leq u \leq 1 \implies 0 \leq v \leq 2$**.**

To get the desired results we aim to show $\Gamma(s) \geq \frac{1}{2}$ for all $s > 0$.

From (2.13) it is clear that $\Gamma(s)$ is a convex function for all $s > 0$ and as

$$\Gamma(1) = \Gamma(2) = 1 \implies \Gamma(s) \geq 1$$

for $0 < s \leq 1$ and $s \geq 2$.

Right now, we are unsure what is happening to the $\Gamma(s)$ value between $1 \leq s \leq 2$, so we will examine it.

$$\Gamma(s) = \int_0^\infty e^{-x} x^{s-1}\, dx = \int_0^1 e^{-x} x^{s-1}\, dx + \int_1^\infty e^{-x} x^{s-1}\, dx. \tag{2.15}$$

observe that for $0 \leq x \leq 1$ and $1 \leq s \leq 2$,

$$x^{s-2} \geq 1,$$

$$\implies x^{s-1} \geq x.$$

Also,for $x > 1$ and $1 \leq s \leq 2$,

$$x^{s-1} \geq 1.$$

Thus, using these observations in (2.15) gives

$$\begin{aligned} \Gamma(s) = \int_0^\infty e^{-x} x^{s-1}\, dx \geq \int_0^1 e^{-x} x\, dx + \int_1^\infty e^{-x}\, dx. \qquad (2.16) \\ = \frac{-1}{e} - \frac{1}{e} + 1 + \frac{1}{e} \\ = 1 - \frac{1}{e} \\ > \frac{1}{2} \end{aligned}$$

Hence,$\Gamma(s) > \frac{1}{2}$ for all $s > 0$.

$$\implies 2\Gamma(s) > 1$$

as a result, we proved the base case $0 \leq v \leq 2$.

$$1 < \frac{1}{2\Gamma(v+1)} \leq \rho(v/2) = 1.$$

Assume the result holds **for** $v \leq V$.

Before we prove the result for $V \leq v \leq V+1$, note that as $\rho(v)$ is a decreasing function (in other words $\rho(v) < \rho(v-1)$) and by (2.11),

$$u\rho(u) = \int_{u-1}^{u} \rho(t)\,dt \geq \int_{u-1}^{u} \frac{\rho(u-1/2)}{2}\,dt = \frac{\rho(u-1/2)}{2},$$
$$\implies u\rho(u) \geq \frac{\rho(u-1/2)}{2}. \tag{2.17}$$

Now, **for** $V \leq v \leq V+1$,

$$\rho(v/2) \geq^{\text{by (2.17)}} \frac{\rho((v-1)/2)}{v} \geq^{\text{by induction hypothesis}} \frac{1}{2v\Gamma(v)} \geq \frac{1}{2\Gamma(v+1)}.$$

Thus, the lower bound also holds for all $v \geq 0$.

2.3 An Asymptotic Estimate for Ψ

Let $\Psi(x,y)$ be the number of positive integers $\leq x$ having prime factors $\leq y$, and let $\rho(u)$ denote the function described above. Then for any $U \geq 0$ we have:

$$\Psi(x, x^{1/u}) = \rho(u)x + O(\frac{x}{\log(x)}) \tag{2.18}$$

uniformly for $0 \leq u \leq U$ and for all $x \geq 2$.

Proof: We will prove this using the Principle of Mathematical Induction. Assume U have integral values and we induct on those values.

Base case: $0 \leq u \leq 1$

By(2.8) we have $\rho(u) = 1$, so the RHS of the statement implies:

$$= x + O(\frac{x}{\log(x)})$$

and by (2.1), the LHS of the statement is:

$$= x + O(1).$$

As $O(1) \leq O(\frac{x}{\log(x)})$ for all $x \geq 2$, and the Big-O notation represents the order of bounding, hence LHS = RHS Thus, the base case holds.

Also, comparing (2.5) and (2.18) for $1 \leq u \leq 2$,

$$\rho(u) = 1 - \log(u) \tag{2.19}$$

Assume that for integer $U \geq 2$ and for all $0 \leq u \leq U$ result holds.

Now, for proving the result for $U \leq u \leq U + 1$,

Let $P(n)$ be the largest prime factor of n, so

$$\Psi(x, y) = 1 + \sum_{p \leq y} card\{n \leq x : P(n) = p\}. \tag{2.20}$$

Here, we add 1 to account for $P(1)$.

$$\implies \Psi(x, y) = 1 + \sum_{p \leq y} \Psi(\frac{x}{p}, p), \tag{2.21}$$

If $y \leq z$

$$\Psi(x, y) - \Psi(x, z) = \sum_{p \leq y} \Psi(\frac{x}{p}, p) - \sum_{p \leq z} \Psi(\frac{x}{p}, p) \tag{2.22}$$

$$\implies \Psi(x, y) = \Psi(x, z) - \sum_{y \leq p \leq z} \Psi(\frac{x}{p}, p). \tag{2.23}$$

Suppose $z = x^{1/U}$ and $y = x^{1/u}$ (keep in mind $U \leq u \leq U + 1$).

Define $u_p = (\frac{\log(x)}{\log(p)} - 1)$, thus $p = (\frac{x}{p})^{\frac{1}{u_p}}$.

Note that for $p \geq y = x^{1/u}$

$$\implies \log(p) \geq \frac{\log(x)}{u},$$

$$\implies u \geq \frac{\log(x)}{\log(p)},$$

$$\implies U \geq u - 1 \geq u_p.$$

Hence, R.H.S of (2.23) using hypothesis implies

$$= \Psi(x, x^{1/U}) - \sum_{y \le p \le z} \Psi(\frac{x}{p}, (\frac{x}{p})^{\frac{1}{u_p}}), \tag{2.24}$$

$$= \rho(U)x + O(\frac{x}{\log(x)}) - x \sum_{y \le p \le z} \frac{\rho(\frac{\log(x)}{\log(p)} - 1)}{p} + O(x \sum_{y \le p \le z} \frac{1}{p \log(\frac{x}{p})}). \tag{2.25}$$

Assuming $s(w) = \sum_{p \le w} \frac{1}{p}$ and using Mertens' Estimates we get:

$$s(w) = \log(\log(w)) + c + r(w). \tag{2.26}$$

Using summation by parts

$$\sum_{y \le p \le z} \frac{\rho(\frac{\log(x)}{\log(p)} - 1)}{p} = \sum_{p \le z} \frac{\rho(\frac{\log(x)}{\log(p)} - 1)}{p} - \sum_{p \le y} \frac{\rho(\frac{\log(x)}{\log(p)} - 1)}{p}. \tag{2.27}$$

$d(\rho(w))$ is representing differentiation of function $\rho(w)$ wrt w.

$$\begin{aligned} = s(z)\rho(\frac{\log(x)}{\log(z)} - 1) - \int_1^z s(w) d(\rho(\frac{\log(x)}{\log(w)} - 1))\, dw - s(y)\rho(\frac{\log(x)}{\log(y)} - 1) \\ + \int_1^y s(w) d(\rho(\frac{\log(x)}{\log(w)} - 1))\, dw \end{aligned} \tag{2.28}$$

$$\begin{aligned} = (\log(\log(z) + c + r(z))\rho(\frac{\log(x)}{\log(z)} - 1) - (\log(\log(y) + c + r(y))\rho(\frac{\log(x)}{\log(y)} - 1) \\ - \int_y^z (\log(\log(w) + c) d(\rho(\frac{\log(x)}{\log(w)} - 1))\, dw - \int_y^z (r(w)) d(\rho(\frac{\log(x)}{\log(w)} - 1))\, dw \end{aligned} \tag{2.29}$$

$$\begin{aligned} = (\log(\log(z) + c + r(z))\rho(\frac{\log(x)}{\log(z)} - 1) - (\log(\log(y) + c + r(y))\rho(\frac{\log(x)}{\log(y)} - 1) \\ -((\log(\log(w) + c+)\rho(\frac{\log(x)}{\log(w)} - 1))|_y^z - \int_y^z (\frac{1}{w \log(w)})\rho(\frac{\log(x)}{\log(w)} - 1)\, dw \\ - \int_y^z (r(w)) d(\rho(\frac{\log(x)}{\log(w)} - 1))\, dw \end{aligned} \tag{2.30}$$

$$= (r(z))\rho(\frac{\log(x)}{\log(z)} - 1) - (r(y))\rho(\frac{\log(x)}{\log(y)} - 1) - \int_y^z (\frac{1}{w\log(w)})\rho(\frac{\log(x)}{\log(w)} - 1)\, dw \\ - \int_y^z (r(w))d(\rho(\frac{\log(x)}{\log(w)} - 1))\, dw. \tag{2.31}$$

Take $t = \frac{\log(x)}{\log(w)}$,

Observe: $d(\log(\log(w))) = \frac{dw}{w\log(w)} = \frac{-dt}{t}$ Hence the 3rd term in above equation becomes:

$$\int_U^u \rho(t-1)\frac{dt}{t}. \tag{2.32}$$

Again using Mertens' estimates, we know $r(w) << \frac{1}{\log(w)}$ and as $\rho(u)$ is decreasing and hence bounded by $1, \because \rho(0) = 1$.

Moreover, as $z = x^{1/U}$ and $y = x^{1/u} \implies \frac{1}{\log(z)}, \frac{1}{\log(y)} << \frac{1}{\log(x)}$

Then,

$$(r(z))\rho(\frac{\log(x)}{\log(z)} - 1) - (r(y))\rho(\frac{\log(x)}{\log(y)} - 1) - \int_y^z (r(w))d(\rho(\frac{\log(x)}{\log(w)} - 1))\, dw \\ << \frac{1}{\log(x)}(1 + \int_y^z |d(\rho(\frac{\log(x)}{\log(w)} - 1))|\, dw \\ << \frac{1}{\log(x)}. \tag{2.33}$$

Now, observe that the 4th term of (2.25)is:

$$O(x \sum_{y \le p \le z} \frac{1}{p\log(\frac{x}{p})}).$$

here as $y \le p \le z \implies x^{1/u} \le p \le x^{1/U} \implies \frac{\log(x)}{u} \le p \le \frac{\log(x)}{U}$.

$$\begin{aligned}\therefore \frac{1}{\log(\frac{x}{p})} &= \frac{1}{\log(x)-\log(p)} \\ &\leq \frac{1}{\log(x)-\frac{\log(x)}{U}} \\ &\leq \frac{1}{\log(x)-\frac{\log(x)}{2}} \\ &<< \frac{1}{\log(x)}.\end{aligned} \tag{2.34}$$

Using Mertens' Estimates, $s(w) = \sum_{p\leq w} \frac{1}{p}$ such that

$$s(w) = \log(\log(w)) + c + r(w),$$

also as $z = x^{1/U}$ and $y = x^{1/u}$,

$$\implies \log(\log(z)) = \log(\log(x)) - \log(U).$$

and

$$\log(\log(y)) = \log(\log(x)) - \log(u).$$

Therefore,

$$\log(\log(z)) - \log(\log(y)) = \log(U) - \log(u) = O(1).$$

Hence the term

$$\begin{aligned}O(x\sum_{y\leq p\leq z}\frac{1}{p\log(\frac{x}{p})}) &<< O(\frac{x}{\log(x)}\sum_{y\leq p\leq z}\frac{1}{p}) \\ &<< O(\frac{x}{\log(x)}).\end{aligned} \tag{2.35}$$

Employing (2.31) to (2.33)and (2.35)in (2.25) gives:

$$\Psi(x, x^{1/u}) = x(\rho(U) - \int_U^u \rho(t-1)\frac{dt}{t}) + O(\frac{x}{\log(x)}), \tag{2.36}$$

At last using (2.9) we get,

$$\Psi(x, x^{1/u}) = \rho(u)x + O(\frac{x}{\log(x)}).$$

Hence result also holds for $U \leq u \leq U+1$.

Thus, the result is true for all $u \geq 0$ by induction.

2.4 An Asymptotic estimate for ρ

Let $\rho(u)$ be the function defined above, adhering to all stated properties.Then Bruijn (in [3],1.8) proved:

$$\rho(u) = exp[-u\log(u) + \log(\log(u)) - 1 - \frac{1}{\log(u)} + \frac{\log(\log(u))}{\log(u)} + O(\frac{(\log(\log(u)))^2}{(\log(u))^2})]. \tag{2.37}$$

Chapter 3

LLL- Algorithm

A lattice is formed by all linear combinations with integer coefficients of a subgroup of any basis in R^n, as formulated in the definition below.

Lattice

Let n be a positive integer. A subset L of the n-dimensional real vector space $\mathbb{R}^n$ is called a lattice if there exists a basis $\{b_1, b_2, \ldots, b_n\}$ of $\mathbb{R}^n$ such that:

$$L = \sum_{i=1}^{n} \mathbb{Z} b_i = \{\sum r_i b_i : r_i \in \mathbb{Z} (1 \leq i \leq n)\}.$$

Here, $b_i (\ 1 \leq i \leq n)$ is referred to as a basis for the lattice L, and hence n is called the rank of L.

Determinant of Lattice

The determinant of a matrix formed by taking the basis vectors of a lattice as columns is called the determinant of a given lattice. Represented as:

$$d(L) = |\det(b_1, b_2, \ldots, b_n)|.$$

Note that this real number does not depend on the choice of the basis; it is fixed for a given lattice.

Gram-Schmidt orthogonalization process

It is a method of constructing an orthogonal basis from a set of vectors in an inner product space. Vectors b_1^* $(1 \leq i \leq n)$ and real numbers $\mu_{i,j}(1 \leq j < i \leq n)$ are inductively defined as:-

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*,$$

$$\mu_{i,j} = \frac{< b_i, b_j^* >}{< b_j^*, b_j^* >},$$

here $<,>$ denotes the ordinary inner product over $\mathbb{R}^n$. This process implies that $\{b_1^*, b_2^*, \ldots, b_n^*\}$ is an orthogonal basis of $\mathbb{R}^n$.

Reduced basis of a Lattice

A basis $\{b_l, b_2, \ldots, b_n\}$ for a lattice L is reduced if

$$|\mu_{i,j}| \leq \frac{1}{2} \tag{3.1}$$

for $1 \leq j < i \leq n$, and

$$|b_i^* + \mu_{i,i-1} b_{i-1}^*|^2 \geq \frac{3}{4} |b_{i-1}^*|^2 \tag{3.2}$$

for $1 < i \leq n$.

The constant $\frac{3}{4}$ is arbitrarily chosen and can be replaced by any fixed real number δ with $\frac{1}{4} < \delta < 1$.

3.1 Properties of Lattice Reduction

If $\{b_1, b_2, \ldots, b_n\}$ be a reduced basis for a lattice L in $\mathbb{R}^n$ and $\{b_1^*, b_2^*, \ldots, b_n^*\}$ is the corresponding orthogonal basis, then [4]

1. $|b_j|^2 \leq 2^{i-1} |b_i^*|^2$ for $1 \leq j \leq i \leq n$,
2. $d(L) \leq \Pi_{i=1}^n |b_i| \leq 2^{\frac{n(n-1)}{4}} d(L)$,

3. $|b_1| \leq 2^{\frac{(n-1)}{4}} d(L)^{\frac{1}{n}}$.

Proof: Consider $\{b_1, b_2, \ldots, b_n\}$ as a reduced basis for a lattice L, therefore all b_i satisfy equations (3.1) and (3.2).

By (3.2), it implies:

$$|b_i^* + \mu_{i,i-1} b_{i-1}^*|^2 \geq \frac{3}{4} |b_{i-1}^*|^2,$$

Applying triangular inequality gives:

$$|b_i^*|^2 \geq (\frac{3}{4} - \mu_{i,i-1}^2) |b_{i-1}^*|^2,$$

Using (3.1):

$$|b_i^*|^2 \geq \frac{1}{2} |b_{i-1}^*|^2.$$

Using induction for $1 < i \leq n$ and taking the above as the base case, we prove:

$$2^{i-j} |b_i^*|^2 \geq |b_j^*|^2. \tag{3.3}$$

Now, from the way b_i^* is defined:

$$|b_i|^2 \leq |b_i^*|^2 + \sum_{j=1}^{i-1} \mu_{i,j}^2 |b_j^*|^2,$$

Using (3.1):

$$\leq |b_i^*|^2 + \sum_{j=1}^{i-1} \frac{1}{4} |b_j^*|^2,$$

Using (3.3):

$$\leq |b_i^*|^2 + \sum_{j=1}^{i-1} \frac{1}{4} 2^{i-j} |b_i^*|^2,$$

Using sum of Geometric Progression series:

$$= (1 + \frac{1}{4}(2^i - 2)) |b_i^*|^2,$$

$$= 2^{i-1} |b_i^*|^2,$$

This implies:

$$|b_i|^2 \leq 2^{i-1}|b_i^*|^2. \tag{3.4}$$

Using (3.3), we get:

$$|b_j|^2 \leq 2^{j-1}|b_j^*|^2 \leq 2^{j-1}2^{i-j}|b_i^*|^2 = 2^{i-1}|b_i^*|^2.$$

Thus, this proves part (1) of the proposition.

By definition of the determinant of a lattice:

$$d(L) = |det(b_1, b_2, \ldots, b_n)| = |det(b_1^*, b_2^*, \ldots, b_n^*)|.$$

As b_i^* are pairwise orthogonal, the determinant becomes:

$$d(L) = \Pi_{i=1}^n |b_i^*|.$$

From the definition of b_i^* and (3.4),

$$b_i^* \leq b_i \leq 2^{\frac{i-1}{2}}|b_i^*|.$$

Thus

$$d(L) \leq \Pi_{i=1}^n |b_i| \leq \Pi_{i=1}^n 2^{\frac{i-1}{2}}|b_i^*| = 2^{\frac{n(n-1)}{4}}\Pi_{i=1}^n |b_i^*| = 2^{\frac{n(n-1)}{4}} d(L),$$

$$\implies d(L) \leq \Pi_{i=1}^n |b_i| \leq 2^{\frac{n(n-1)}{4}} d(L).$$

This proves part (2) of the proposition.

For part(3), setting $j = 1$ in the expression (1), we get:

$$|b_1|^2 \leq 2^{i-1}|b_i^*|^2$$

for $1 \leq i \leq n$.

Taking the product over $i = 1, 2...n$, we get:

$$|b_1|^n \leq 2^{\frac{n(n-1)}{4}}\Pi_{i=1}^n |b_i^*|^n,$$

$$\implies |b_1| \leq 2^{\frac{(n-1)}{4}} d(L)^{\frac{1}{n}}.$$

$$- - - - - - - - - - - - - - -$$

3.2 LLL algorithm for finding reduced basis

To transform a given basis $b_1, b_2, \ldots, b_n$ for a lattice L into a reduced one, we first compute b_i^* and μ_{ij} using the Gram-Schmidt orthogonalization process. While working with algorithm [4], the basis elements $b_1, b_2, \ldots, b_n$ will be changed many times, maintaining the resultant as a basis of L. The algorithm starts with an index $i = 2$ and proceeds from there.

Step1: Check for

$$|\mu_{i,i-1}| \leq \frac{1}{2},$$

if this condition is satisfied then jump to step 2.

Else

Let r represent the nearest integer to $\mu_{i,i-1}$, and update:

$$b_i = b_i - rb_{i-1}.$$

Then recompute all $\mu_{i,j}$ accordingly using this updated value of b_i for all $1 \leq j < i$. Note that all other $\mu_{i,j}$ and all b_i^* remain unchanged.

Step2:

Check for condition

$$|b_i^* + \mu_{i,i-1}b_{i-1}^*|^2 \geq \frac{3}{4}|b_{i-1}^*|^2.$$

If this is satisfied, then increment i to $i + 1$.

Else Swap b_i with b_{i-1} and recompute all corresponding $\mu_{i,j}$ and b_i^*,b_{i-1}^* .

Repeat steps 1 and 2 until $i = n$. By the $i = n$ step, you have a reduced basis.

3.3 Illustrative Example of Lattice Reduction

Consider a basis b_1, b_2, b_3 for $\mathbb{R}^3$ given by the columns of the matrix: $\begin{pmatrix} 1 & -1 & 3 \\ 1 & 0 & 5 \\ 1 & 2 & 6 \end{pmatrix}$.

Find the reduced basis using the LLL algorithm. [2]

Solution:

Using Gram-Schmidt orthogonalization,

$$b_1^* = b_1.$$

Start the reduction process with $i = 2$.

$$\mu_{2,1} = \frac{< b_2, b_1^* >}{< b_1^*, b_1^* >} = \frac{1}{3},$$

Then

$$b_2^* = b_2 - \mu_{2,1} b_1^* = b_2 - \frac{1}{3} b_1^* = \begin{pmatrix} \frac{-4}{3} \\ \frac{-1}{3} \\ \frac{5}{3} \end{pmatrix}.$$

Following the reduction steps, clearly:

$$\mu_{2,1} < \frac{1}{2},$$

also checking

$$|b_2^* + \mu_{2,1} b_1^*|^2 \geq \frac{3}{4} |b_1^*|^2,$$

gives:

$$5 > \frac{9}{4},$$

which is true. Hence, we increment i to $i = 3$.

$$\mu_{3,1} = \frac{< b_3, b_1^* >}{< b_1^*, b_1^* >} = \frac{14}{3},$$

$$\mu_{3,2} = \frac{< b_3, b_2^* >}{< b_2^*, b_2^* >} = \frac{13}{14},$$

Then,

$$b_3^* = b_3 - \mu_{3,1} b_1^* - \mu_{3,2} b_2^* = b_3 - \frac{14}{3} b_1^* - \frac{13}{14} b_2^* = \begin{pmatrix} \frac{-6}{14} \\ \frac{9}{14} \\ \frac{-3}{14} \end{pmatrix}.$$

As

$$\mu_{3,2} = \frac{13}{14} > \frac{1}{2}.$$

Therefore following step 1 of reduction process gives,

$$b_3 = b_3 - r b_2,$$

here $r = 1$ that is round value of $\mu_{3,2}$.

$$b_3 = b_3 - rb_2 = \begin{pmatrix} 3 \\ 5 \\ 6 \end{pmatrix} - \begin{pmatrix} -1 \\ 0 \\ 2 \end{pmatrix} = \begin{pmatrix} 4 \\ 5 \\ 4 \end{pmatrix}.$$

So now

$$\mu_{3,2} = \frac{-1}{14} < \frac{1}{2}.$$

Checking

$$|b_3^* + \mu_{3,2} b_2^*|^2 \geq \frac{3}{4}|b_2^*|^2,$$

Gives

$$\frac{5}{9} \geq \frac{14}{4},$$

but that's not true, so we do swapping of b_2 and b_3 step and start working with matrix

$$\begin{pmatrix} 1 & 4 & -1 \\ 1 & 5 & 0 \\ 1 & 4 & 2 \end{pmatrix}.$$

Again using Gram-Schmidt orthogonalization

$$b_1^* = b_1,$$

remains unchanged.

$$\mu_{2,1} = \frac{< b_2, b_1^* >}{< b_1^*, b_1^* >} = \frac{13}{3}.$$

Then,

$$b_2^* = b_2 - \mu_{2,1} b_1^* = b_2 - \frac{13}{3} b_1^* = \begin{pmatrix} \frac{-1}{3} \\ \frac{-2}{3} \\ \frac{1}{3} \end{pmatrix}.$$

As

$$\mu_{2,1} = \frac{13}{3} > \frac{1}{2},$$

so by reduction step 1

$$b_2 = b_2 - rb_1,$$

here $r = 4$ that is round value of $\mu_{2,1}$.

$$b_2 = b_2 - rb_1 = \begin{pmatrix} 4 \\ 5 \\ 4 \end{pmatrix} - 4 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}.$$

Now

$$\mu_{2,1} = \frac{1}{3} < \frac{1}{2}.$$

Checking

$$|b_2^* + \mu_{2,1} b_1^*|^2 \geq \frac{3}{4}|b_1^*|^2,$$

thus gives,

$$1 \geq \frac{9}{4},$$

That's not true and hence again swapping b_2 and b_1 we get matrix

$$\begin{pmatrix} 0 & 1 & -1 \\ 1 & 1 & 0 \\ 0 & 1 & 2 \end{pmatrix}.$$

Again by Gram-Schmidt orthogonalization,

$$b_1^* = b_1,$$

remains unchanged.

$$\mu_{2,1} = \frac{< b_2, b_1^* >}{< b_1^*, b_1^* >} = 1 > \frac{1}{2},$$

so by reduction step 1,

$$b_2 = b_2 - rb_1,$$

here $r = 1$ that is round value of $\mu_{2,1}$.

$$b_2 = b_2 - rb_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - 1 \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}.$$

Now evaluating

$$\mu_{2,1} = 0,$$

$$\mu_{3,1} = 0,$$

$$b_2^* = b_2,$$
$$\mu_{3,2} = \frac{1}{2},$$
$$b_3^* = b_3 - \mu_{3,1}b_1^* - \mu_{3,2}b_2^* = b_3 - \frac{1}{2}b_2^* = \begin{pmatrix} \frac{-3}{2} \\ \frac{-1}{2} \\ \frac{3}{2} \end{pmatrix}.$$

Checking

$$|b_3^* + \mu_{3,2}b_2^*|^2 \geq \frac{3}{4}|b_2^*|^2,$$

gives

$$5 > \frac{3}{2},$$

that's true.

Hence, increment the value of i to $i = 3 = n$. This terminates the process, and we have the reduced basis matrix

$$\begin{pmatrix} 0 & 1 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 2 \end{pmatrix}.$$

– – – – – – – – – – –

3.4 Generalizations of properties of Lattice Reduction

If $\{b_1, b_2, \ldots, b_n\}$ is a δ-reduced basis for a lattice L and $\lambda_1(L)$ represents the shortest non-zero vector in lattice L then,

1. $||b_1|| \leq \frac{2}{\sqrt{4\delta-1}}^{(n-1)} \lambda_1(L)$,
2. $||b_1|| \leq \frac{2}{\sqrt{4\delta-1}}^{\frac{(n-1)}{2}} d(L)^{\frac{1}{n}}$.

   where $d(L)$ denotes the determinant of lattice L.

3.5 Computational Complexity of the LLL Algorithm

As per the Classic LLL algorithm discussed above, if $L \subset \mathbb{Z}^n$ is a lattice with basis $b_1, b_2, \ldots, b_n$, and let $B \in \mathbb{R}$, $B > 2$, be such that $|b_i^2| \leq B$ for $1 < i < n$. Then the number of arithmetic operations required by the basis reduction algorithm is $O(n^4 \log(B))$, and the integers involved in these operations each have binary length $O(n \log(B))$.

Thus, the number of bit operations required by the basis reduction algorithm is $O(n^6(\log(B))^3)$, with the implied constant in the O-notation being proportionate to $\frac{1}{\log(\frac{1}{\sqrt{\delta}})}$. This complexity can be reduced to $O(n^{5+\epsilon}(\log(B))^{2+\epsilon})$, for every $\epsilon \geq 0$, using fast multiplication techniques [4].

The LLL algorithm has undergone significant improvements over time, with the advancement [6], the running time of advanced algorithms has been reduced to $O(n^\omega(C + n)^{1+\epsilon})$ where $C > \log(\|B\|\|B - 1\|)$ for an integer lattice B of dimension n and $O(n^\omega)$ denotes the running time of matrix multiplication, size reduction, and QR factorization for some $\omega \in (2, 3]$.

In particular, if B is upper triangular and size-reduced, then the running time is $O(n^\omega(\log \|B\|_{max} + n)^{1+\epsilon})$.

Chapter 4

Boneh's Algorithm

In this section, we discuss Boneh's algorithm for finding smooth numbers. This algorithm was published in [1].

4.1 A Key Lemma for Algorithm

Lemma 1 *Let S and B be positive integers and let $w(x) \in \mathbb{Z}[x]$ be a polynomial of degree $d-1$. Let the norm of a polynomial $w(x) = \sum_{i=0}^{d-1} c_i x_i$ be $\|w(x)\|^2 = \sum_{i=0}^{d-1} c_i^2$. Suppose that*

1. *$w(x_0) = 0 \mod S$ for some integer x_0 where $|x_0| < B$, and*
2. *$\|w(xB)\| < \frac{S}{\sqrt{d}}$.*

Then $w(x_0) = 0$ holds on integers.

Proof: let $w(x) = a_0 + \cdots + a_{d-1}x^{d-1}$ then

$$|w(x_0)| = |\sum a_i x_0^i|$$

using $|x_0| < B$ and taking summation out of mode

$$\leq \sum |a_i B^i|$$

$$\leq \sqrt{\sum_{0}^{d-1} 1}\sqrt{\sum_{0}^{d-1} |c_i B^i|^2}$$

(assuming 2.)

$$= \sqrt{d}\|w(x.B)\| < S$$

since $|w(x_0)| \equiv 0$ mod S and $|w(x_0)| < S$ implies $w(x_0) = 0$.

Definition

Let $B, < p_1, \ldots, p_n >$ and $< r_1, \ldots, r_n >$, The values $0 < p_1 < p_2 < \cdots < p_n$ are relatively prime integers (not necessarily primes), and $B, r_1, \ldots, r_n \in \mathbb{Z}$. Set $P = \Pi_{i=1}^{n} p_i$. Let R be an integer such that $R = r_i$ (Mod p_i) for all $i = 1, \ldots, n$. For an integer m, we define **the amplitude of** m as

$$amp(m) = gcd(P, m - R)$$

4.2 Core Theorem of Boneh's Algorithm

Let $B, < p_1, \ldots, p_n > and < r_1, \ldots, r_n >$, where $0 < p_1 < p_2 < \cdots < p_n$ are relatively prime integers (not necessarily primes), and $B, r_1, \ldots, r_n \in \mathbb{Z}$. Define $P = \Pi_{i=1}^{n} p_i$. Then, for any $d \geq 1 + \sqrt{\frac{\log_2(P)}{\log_2(B)}}$, there is an algorithm that outputs all positive integers $m < B$ for which

$$amp(m) > P^{\epsilon}$$

where

$$\epsilon = \sqrt{\frac{\log_2(4B)}{\log_2(P)}} + \frac{5}{4d}.$$

The running time of the algorithm is dominated by the time it takes to run the LLL on a lattice of dimension d. When $d \geq 20 \log_2(P)$, the algorithm produces all positive $m < B$ for which $amp(m) > P^{\sqrt{\frac{\log_2(4B)}{\log_2(p)}}}$.

Proof: Let $R \in [0, P]$ be an integer such that by the Chinese Remainder Theorem

$$R = r_i \mod p_i, \quad \text{for all } i = 1, \ldots, n.$$

Let a and a' be two integers, determined in further discussion. Let us define two groups of polynomials in $\mathbb{Z}[x]$:

$$g_i(x) = P^{a-i}(x-R)^i$$

for $i = 0, \ldots, a-1$

$$h_i(x) = (x-R)^a x^i$$

for $i = 0, \ldots, a'-1$

Let $0 < m < B$ be an integer for which $amp(m)$ satisfies the bound of 4.2. Set $M = amp(m)$.We know that $m - R = 0 \mod M$ (since $amp(m)$ divides $m - R$). By the way g_i and h_i are defined, we have

$$g_i(m) = 0 \mod M^a$$

for all $i = 0, \ldots, a-1$

$$h_i(m) = 0 \mod M^a$$

for all $i = 0, \ldots, a'-1$.

Further, if w(x) is considered any integer linear combination of these polynomials g_i and h_i then also

$$w(x) \equiv 0 \mod M^a.$$

Now employing 1,if

$$\|w(xB)\| < \frac{M^a}{\sqrt{deg(w)+1}}, \tag{4.2.1}$$

then m will be a root over integers for $w(x)$. Hence, to find a polynomial $w(x)$ over integers such that it has all m as its roots, it is enough to construct a linear combination of polynomials $g_i(xB)$ and $h_i(xB)$ that satisfies the above bound given by 4.2.1.

Let us construct a lattice having dimension $d = a + a'$.The lattice is spanned by the coefficient vectors of the polynomials $g_i(xB)$ for $i = 0, ..., a$ and $h_i(xB)$ for $i = 0, ..., a'$. For example: taking $a = 4$ and $a' = 3$, the resulting lattice L of dimension 7 is spanned by the rows of the matrix:

	1	x	x^2	x^3	x^4	x^5	x^6
$g_0(xB)$	P^4						
$g_1(xB)$	$-RP^3$	P^3B					
$g_2(xB)$	R^2P^2	$-2RP^2B$	P^2B^2				
$g_3(xB)$	$-R^3P$	$3R^2PB$	$-3RPB^2$	PB^3			
$h_0(xB)$	R^4	$-4R^3B$	$6R^2B^2$	$-4RB^3$	B^4		
$h_1(xB)$		R^4B	$-4R^3B^2$	$6R^2B^3$	$-4RB^4$	B^5	
$h_2(xB)$			R^4B^2	$-4R^3B^3$	$6R^2B^4$	$-4RB^5$	B^6

Now using the LLL Algorithm we can get a short vector which can be assumed as the coefficient vector of the polynomial $w(xB)$.

(as LLL algorithm for given lattice spanned by $< u_1, ..., u_d >$, produces a vector v satisfying: $\|v\| \leq 2^{d/2} det(L)^{1/d}$) using $\delta = \frac{3}{4}$ specifically)

therefore $w(xB)$ satisfies

$$\|w(xB)\| \leq 2^{d/2} det(L)^{1/d}. \tag{4.2.2}$$

now if

$$2^{d/2} det(L)^{1/d} < \frac{M^a}{\sqrt{d}}, \tag{4.2.3}$$

It guaranteed that LLL will produce a polynomial $w(xB)$ such that

$$\|w(xB)\| < \frac{M^a}{\sqrt{d}}. \tag{4.2.4}$$

satisfying 4.2.1. Rearranging 4.2.3 we get

$$M > det(L)^{1/ad} \gamma(d)^{1/a}. \tag{4.2.5}$$

where $\gamma(d) = 2^{\frac{d}{2}}\sqrt{d}$.

Note that the determinant of L is simply the product of the elements on the diagonal, as is clear from the above example that the basis of L is a triangular matrix.

$$det(L) = P^{\frac{a(a+1)}{2}} B^{\frac{d(d-1)}{2}}.$$

4.2.5 then implies,

$$M > P^{\frac{(a+1)}{2d}} B^{\frac{(d-1)}{2a}} \gamma(d)^{1/a} = P^{\frac{1}{2d}} P^{\frac{a}{2d} + \frac{1}{a} \cdot (\frac{\log_2(B)}{\log_2(P)} \frac{d-1}{2} + \frac{\log_2(\gamma(d))}{\log_2(P)})}. \tag{4.2.6}$$

For simplicity just replace $C = \frac{1}{2d}$ and $D = [\frac{\log_2(B)}{\log_2(P)} \frac{d-1}{2} + \frac{\log_2(\gamma(d))}{\log_2(P)}]$.

Then we want an integer $a \in [1, ..., d]$ that minimizes the RHS of 4.2.6 that is:

$$P^{C + aC + \frac{D}{a}}. \tag{4.2.7}$$

Taking the derivative of this exponent and replacing values for D,C

$$a_{opt} = \sqrt{\frac{D}{C}} = d\sqrt{\frac{(1 - \frac{1}{d}) \log_2(B)}{\log_2(P)} + \frac{2 \log_2(\gamma(d))}{d \log_2(P)}}, \tag{4.2.8}$$

$$= d\sqrt{\frac{(\frac{d-1}{d})\log_2(B)}{\log_2(P)} + \frac{2\log_2(\gamma(d))}{d\log_2(P)}} \tag{4.2.9}$$

By assumption we have:

$$d \geq 1 + \sqrt{\frac{\log_2(P)}{\log_2(B)}},$$

then also

$$d > \sqrt{\frac{\log_2(P)}{\log_2(B)}}. \tag{4.2.10}$$

Using this bound for d in 4.2.8 gives,

$$a_{opt} \geq d\sqrt{\frac{\sqrt{\frac{\log_2(P)}{\log_2(B)}}\log_2(B)}{d\log_2(P)} + \frac{2\log_2(\gamma(d))}{d\log_2(P)}}, \tag{4.2.11}$$

taking d inside the square root and canceling

$$\geq \sqrt{d.\sqrt{\frac{\log_2(B)}{\log_2(P)}} + \frac{2d\log_2(\gamma(d))}{\log_2(P)}}, \tag{4.2.12}$$

using 4.2.10

$$\sqrt{\sqrt{\frac{\log_2(P)}{\log_2(B)}}\sqrt{\frac{\log_2(B)}{\log_2(P)}} + \frac{2d\log_2(\gamma(d))}{\log_2(P)}}, \tag{4.2.13}$$

$$= \sqrt{1 + \frac{2d\log_2(\gamma(d))}{\log_2(P)}}. \tag{4.2.14}$$

Hence, $a_{opt} > 1$

Let $[a_{opt}]$ be the closest integer to a_{opt}.

$$a_{opt} - \frac{1}{2} \leq [a_{opt}] \leq a_{opt} + \frac{1}{2}$$

$$\frac{1}{[a_{opt}]} \leq \frac{1}{a_{opt} - \frac{1}{2}} \leq \frac{1}{a_{opt}} + \frac{1}{a_{opt}^2}$$

Note that the second inequality in the above equation holds as:

$$\frac{1}{X-\frac{1}{2}} \leq \frac{1}{X} + \frac{1}{X^2}$$

$$\Leftrightarrow X^2 \leq (X - \frac{1}{2}) * (X+1)$$

$$\Leftrightarrow X^2 \leq X^2 + \frac{X}{2} - \frac{1}{2}$$

$$\Leftrightarrow 0 \leq \frac{X}{2} - \frac{1}{2}$$

$$\Leftrightarrow X \geq 1$$

as $a_{opt} > 1$, so the above inequality holds.

Therefore exponent of P in 4.2.7 becomes,

$$C + [a_{opt}]C + \frac{D}{[a_{opt}]} \leq C + [a_{opt} + \frac{1}{2}]C + [\frac{1}{a_{opt}} + \frac{1}{a_{opt}^2}]D, \tag{4.2.15}$$

substituting value of a_{opt}

$$= C + [\sqrt{\frac{D}{C}} + \frac{1}{2}]C + [\sqrt{\frac{C}{D}} + \frac{C}{D}]D, \tag{4.2.16}$$

$$= 2\sqrt{CD} + \frac{5}{2}C, \tag{4.2.17}$$

substituting values for C and D gives

$$= \sqrt{(1 - \frac{1}{d})\frac{\log_2(B)}{\log_2(P)} + \frac{2\log_2(\gamma(d))}{d\log_2(P)}} + \frac{5}{4d} = \epsilon. \tag{4.2.18}$$

So, 4.2.5 is satisfied if $M > P^{\epsilon}$.

Note that $\frac{2\log_2(\gamma(d))}{d} < 1.75$ for all $d > 0$.

Furthermore, we can simplify the bound by dropping the $(1 - \frac{1}{d})$ factor in)4.2.18 and we get

$$M > P^{\epsilon},$$

where

$$\epsilon = \sqrt{\frac{1.75 + \log_2(B)}{\log_2(P)}} + \frac{5}{4d}. \tag{4.2.19}$$

To sum up, once we got $w(xB)$ such that it satisfies when $M > P^\epsilon$, then $w(x)$ will have property: if $0 < m < B$ and $amp(m) > M$ then m is a root of $w(x)$ over the integers.

Thus, we evaluate $w(x)$ and then use some root-finding algorithm in $\mathbb{R}$ to find all integer roots of $w(x)$ and output the roots m that satisfy $amp(m) > P^\epsilon$.

Now observe that

$$\sqrt{1.75 + \log_2(B)} = \sqrt{\log_2(4) + \log_2(B) - \frac{1}{4}} < \sqrt{\log_2(4B)} - \frac{\frac{1}{8}}{\sqrt{\log_2(4B)}}. \tag{4.2.20}$$

Observe that if $d > 20\log_2(P)$ then definitely $d > 10\sqrt{\log_2(P)\log_2(4B)}$ as $P > B \geq 1$.

Hence,4.2.19 can be manipulated as,

$$\epsilon = \sqrt{\frac{1.75 + \log_2(B)}{\log_2(P)}} + \frac{5}{4d} < \sqrt{\frac{\log_2(4B)}{\log_2(P)}} - \frac{\frac{1}{8}}{\sqrt{\log_2(P)\log_2(4B)}} + \frac{5}{4d}, \tag{4.2.21}$$

using $d > 10\sqrt{\log_2(P)\log_2(4B)}$ implies

$$\epsilon < \sqrt{\frac{\log_2(4B)}{\log_2(P)}}.$$

This completes the proof.

– – – – ––

4.3 Refinements to Boneh's Theorem

Theorem 1 (Version 1, more accurate approximations) *Let $B, < p_1, \ldots, p_n >$ and $< r_1, \ldots, r_n >$, where $0 < p_1 < p_2 < \cdots < p_n$ are relatively prime integers (not necessarily prime), and $B, r_1, \ldots, r_n \in \mathbb{Z}$. Define $P = \Pi_{i=1}^n p_i$. Then, for any $d \geq 1 + \sqrt{\frac{\log_2(P)}{\log_2(B)}}$, there is an algorithm that outputs all positive integers $m < B$ for which*

$$amp(m) > P^\epsilon$$

where

$$\epsilon = \sqrt{\frac{\log_2(3B)}{\log_2(P)}} + \frac{5}{4d}.$$

The running time of the algorithm is dominated by the time it takes to run the LLL on a lattice of dimension d. When $d \geq 20\log_2(P)$*, the algorithm produces all positive* $m < B$ *for which* $amp(m) > P^{\sqrt{\frac{\log_2(3B)}{\log_2(p)}}}$.

Proof: Let $R \in [0, P]$ be an integer such that by the Chinese Remainder Theorem

$$R = r_i \mod p_i, \quad \text{for all } i = 1, \ldots, n.$$

Let a and a' be two integers, determined in further discussion. Let us define two groups of polynomials in $\mathbb{Z}[x]$:

$$g_i(x) = P^{a-i}(x - R)^i$$

for $i = 0, \ldots, a-1$

$$h_i(x) = (x - R)^a x^i$$

for $i = 0, \ldots, a' - 1$

Let $0 < m < B$ be an integer for which $amp(m)$ satisfies the bound of 1. Set $M = amp(m)$.We know that $m - R = 0 \mod M$ (since $amp(m)$ divides $m - R$). By the way g_i and h_i are defined, it follows that:

$$g_i(m) = 0 \mod M^a$$

for all $i = 0, \ldots, a-1$

$$h_i(m) = 0 \mod M^a$$

for all $i = 0, \ldots, a' - 1$.

Further, if w(x) is considered any integer linear combination of these polynomials g_i and h_i then also

$$w(x) \equiv 0 \mod M^a.$$

Now employing 1,if

$$\|w(xB)\| < \frac{M^a}{\sqrt{deg(w) + 1}}, \tag{4.3.1}$$

then m will be a root over integers for $w(x)$. Hence, to find a polynomial $w(x)$ over integers such that it has all m as its roots, it is enough to construct a linear combination of polynomials $g_i(xB)$ and $h_i(xB)$ that satisfy the above bound given by 4.3.1.

Let us construct a lattice having dimension $d = a + a'$.The lattice is spanned by the coefficient vectors of the polynomials $g_i(xB)$ for $i = 0, ..., a$ and $h_i(xB)$ for $i = 0, ..., a'$. For example: taking $a = 4$ and $a' = 3$, the resulting lattice L of dimension 7 is spanned by the rows of the matrix:

	1	x	x^2	x^3	x^4	x^5	x^6
$g_0(xB)$	P^4						
$g_1(xB)$	$-RP^3$	P^3B					
$g_2(xB)$	R^2P^2	$-2RP^2B$	P^2B^2				
$g_3(xB)$	$-R^3P$	$3R^2PB$	$-3RPB^2$	PB^3			
$h_0(xB)$	R^4	$-4R^3B$	$6R^2B^2$	$-4RB^3$	B^4		
$h_1(xB)$		R^4B	$-4R^3B^2$	$6R^2B^3$	$-4RB^4$	B^5	
$h_2(xB)$			R^4B^2	$-4R^3B^3$	$6R^2B^4$	$-4RB^5$	B^6

Now, using the LLL algorithm we can get a short vector that can be assumed as the coefficient vector of the polynomial $w(xB)$.

(because the LLL algorithm, for a given lattice spanned by $< u_1, ..., u_d >$, produces a vector v satisfying: $\|v\| \leq 2^{d/2} det(L)^{1/d}$ using $\delta = \frac{3}{4}$ specifically.)

therefore $w(xB)$ satisfies

$$\|w(xB)\| \leq 2^{d/2} det(L)^{1/d} \tag{4.3.2}$$

now if

$$2^{d/2}.det(L)^{1/d} < \frac{M^a}{\sqrt{d}} \tag{4.3.3}$$

its guaranteed that LLL will produce a polynomial $w(xB)$ such that

$$\|w(xB)\| < \frac{M^a}{\sqrt{d}} \tag{4.3.4}$$

satisfying 4.3.1. Rearranging 4.3.3 we get

$$M > det(L)^{1/ad} \gamma(d)^{1/a} \tag{4.3.5}$$

where $\gamma(d) = 2^{\frac{d}{2}}\sqrt{d}$ Note that the determinant of L is simply the product of the elements on the diagonal as it's clear from the above example that the basis of L is a triangular matrix.

$$det(L) = P^{\frac{a(a+1)}{2}} B^{\frac{d(d-1)}{2}}$$

4.3.5 then implies

$$M > P^{\frac{a(a+1)}{2}} B^{\frac{d(d-1)}{2}} \gamma(d)^{1/a} = P^{\frac{1}{2d}} P^{\frac{a}{2d}+\frac{1}{a}\cdot(\frac{\log_2(B)}{\log_2(P)}\frac{d-1}{2}+\frac{\log_2(\gamma(d))}{\log_2(P)})} \tag{4.3.6}$$

For simplicity just replace $C = \frac{1}{2d}$ and $D = \frac{\log_2(B)}{\log_2(P)}\frac{d-1}{2} + \frac{\log_2(\gamma(d))}{\log_2(P)}]$ Then we want an integer $a \in [1, ..., d]$ that minimizes R.H.S of 4.3.6 that is

$$P^{C+aC+\frac{D}{a}} \tag{4.3.7}$$

taking the derivative of this exponent and replacing values for D and C

$$a_{opt} = \sqrt{\frac{D}{C}} = d\sqrt{\frac{(1-\frac{1}{d})\log_2(B)}{\log_2(P)} + \frac{2\log_2(\gamma(d))}{d\log_2(P)}} \tag{4.3.8}$$

$$= d\sqrt{\frac{(\frac{d-1}{d})\log_2(B)}{\log_2(P)} + \frac{2\log_2(\gamma(d))}{d\log_2(P)}} \tag{4.3.9}$$

By assumption we have:

$$d \geq 1 + \sqrt{\frac{\log_2(P)}{\log_2(B)}}$$

then also

$$d > \sqrt{\frac{\log_2(P)}{\log_2(B)}} \tag{4.3.10}$$

using this bound for d in 4.3.8 gives

$$a_{opt} \geq d\sqrt{\frac{\sqrt{\frac{\log_2(P)}{\log_2(B)}}\log_2(B)}{d\log_2(P)} + \frac{2\log_2(\gamma(d))}{d\log_2(P)}} \tag{4.3.11}$$

taking d inside the square root and canceling

$$\geq \sqrt{d\sqrt{\frac{\log_2(B)}{\log_2(P)}} + \frac{2d\log_2(\gamma(d))}{\log_2(P)}} \tag{4.3.12}$$

using 4.3.10

$$> \sqrt{\sqrt{\frac{\log_2(P)}{\log_2(B)}}\sqrt{\frac{\log_2(B)}{\log_2(P)}} + \frac{2d\log_2(\gamma(d))}{\log_2(P)}} \tag{4.3.13}$$

$$= \sqrt{1 + \frac{2d \log_2(\gamma(d))}{\log_2(P)}} \tag{4.3.14}$$

Hence, $a_{opt} > 1$

Let $[a_{opt}]$ be the closest integer to a_{opt}.

$$a_{opt} - \frac{1}{2} \leq [a_{opt}] \leq a_{opt} + \frac{1}{2}$$

$$\frac{1}{[a_{opt}]} \leq \frac{1}{a_{opt} - \frac{1}{2}} \leq \frac{1}{a_{opt}} + \frac{1}{a_{opt}^2}$$

Note that the second inequality in the above equation holds as:

$$\frac{1}{X - \frac{1}{2}} \leq \frac{1}{X} + \frac{1}{X^2}$$

$$\Leftrightarrow X^2 \leq (X - \frac{1}{2}) * (X + 1)$$

$$\Leftrightarrow X^2 \leq X^2 + \frac{X}{2} - \frac{1}{2}$$

$$\Leftrightarrow 0 \leq \frac{X}{2} - \frac{1}{2}$$

$$\Leftrightarrow X \geq 1$$

as $a_{opt} > 1$, so the above inequality holds.

Therefore, exponent of P in 4.3.7 becomes,

$$C + [a_{opt}]C + \frac{D}{[a_{opt}]} \leq C + [a_{opt} + \frac{1}{2}]C + [\frac{1}{a_{opt}} + \frac{1}{a_{opt}^2}]D \tag{4.3.15}$$

substituting value of a_{opt}

$$= C + [\sqrt{\frac{D}{C}} + \frac{1}{2}]C + [\sqrt{\frac{C}{D}} + \frac{C}{D}]D \tag{4.3.16}$$

$$= 2\sqrt{CD} + \frac{5}{2}C \tag{4.3.17}$$

substituting values for C and D gives

$$= \sqrt{(1 - \frac{1}{d}).\frac{\log_2(B)}{\log_2(P)} + \frac{2\log_2(\gamma(d))}{d\log_2(P)}} + \frac{5}{4d} = \epsilon \tag{4.3.18}$$

So, 4.3.5 is satisfied if $M > P^{\epsilon}$ Note that $\frac{2\log_2\gamma(d)}{d} < 1.55$ for all $d > 0$. because

$$\frac{2\log_2(\gamma(d))}{d} = \frac{2\log_2(2^{\frac{d}{2}}\sqrt{d})}{d} \tag{4.3.19}$$

after applying the properties of the log function

$$= \frac{2(\frac{d}{2}\log_2(2) + \frac{1}{2}\log_2(d))}{d} \tag{4.3.20}$$

cancellation results in

$$1 + \frac{\log_e(d)}{(\log_e(2))d} \tag{4.3.21}$$

its derivative is

$$\frac{1}{\log_e(2)}(\frac{1}{d^2} - \frac{\log_e(d)}{d^2}) \tag{4.3.22}$$

equating it to 0 for finding the maximum value point of expression in 4.3.19. Maximum value occurs at point $d =$ e that is 1.55 implies

$$\frac{2\log_2(\gamma(d))}{d} < 1.55 \tag{4.3.23}$$

for all $d > 0$. Furthermore, we can simplify the bound by dropping the $(1 - \frac{1}{d})$ factor in 4.3.18 and we get

$$M > P^{\epsilon}$$

where

$$\epsilon = \sqrt{\frac{1.55 + \log_2(B)}{\log_2(P)}} + \frac{5}{4d} \tag{4.3.24}$$

To sum up, once we got $w(xB)$ such that it satisfies 4.3.1 when $M > P^{\epsilon}$ then $w(x)$ will have property: if $0 < m < B$ and $amp(m) > M$ then m is a root of $w(x)$ over the integers.

Thus, we evaluate $w(x)$ and then use some root-finding algorithm in $\mathbb{R}$ to find all integer roots of $w(x)$ and output the roots m that satisfy $amp(m) > P^{\epsilon}$.

Now observe that

$$\sqrt{1.55 + \log_2(B)} < \sqrt{\log_2(3) + \log_2(B)} = \sqrt{\log_2(3B)} \tag{4.3.25}$$

Hence,4.3.24 can be manipulated as

$$\epsilon = \sqrt{\frac{1.55 + \log_2(B)}{\log_2(P)}} + \frac{5}{4d} < \sqrt{\frac{\log_2(3B)}{\log_2(P)}} + \frac{5}{4d} \tag{4.3.26}$$

implies

$$\epsilon < \sqrt{\frac{\log_2(3B)}{\log_2(P)}} \tag{4.3.27}$$

This completes the proof.

– – – – ––

Theorem 2 (Version 2, Changing parameter(δ) value) *Let $B, < p_1, \ldots, p_n >$ and $< r_1, \ldots, r_n >$, where $0 < p_1 < p_2 < \cdots < p_n$ are relatively prime integers (not necessarily primes), and $B, r_1, \ldots, r_n \in \mathbb{Z}$. Define $P = \Pi_{i=1}^{n} p_i$. Then, for any $d \geq 1 + \sqrt{\frac{\log_2(P)}{\log_2(B)}}$, there is an algorithm that outputs all positive integers $m < B$ for which*

$$amp(m) > P^{\epsilon}$$

where

$$\epsilon = \sqrt{\frac{\log_2(2B)}{\log_2(P)}} + \frac{5}{4d}.$$

The running time of the algorithm is dominated by the time it takes to run the LLL on a lattice of dimension d. When $d \geq 20 \log_2(P)$, the algorithm produces all positive $m < B$ for which $amp(m) > P^{\sqrt{\frac{\log_2(2B)}{\log_2(p)}}}$.

Proof: Let $R \in [0, P]$ be an integer such that by the Chinese Remainder Theorem

$$R = r_i \mod p_i, \quad \text{for all } i = 1, \ldots, n.$$

Let a and a' be two integers, determined in further discussion. Let us define two groups of polynomials in $\mathbb{Z}[x]$:

$$g_i(x) = P^{a-i}(x - R)^i$$

for $i = 0, \ldots, a-1$

$$h_i(x) = (x-R)^a x^i$$

for $i = 0, \ldots, a'-1$

Let $0 < m < B$ be an integer for which $amp(m)$ satisfies the bound of 1. Set $M = amp(m)$.We know that $m - R = 0 \mod M$ (since $amp(m)$ divides $m - R$). By the way g_i and h_i are defined, it follows that:

$$g_i(m) = 0 \mod M^a$$

for all $i = 0, \ldots, a-1$

$$h_i(m) = 0 \mod M^a$$

for all $i = 0, \ldots, a'-1$.

Further, if w(x) is considered any integer linear combination of these polynomials g_i and h_i then also

$$w(x) \equiv 0 \mod M^a.$$

Now employing 1,if

$$\|w(xB)\| < \frac{M^a}{\sqrt{deg(w)+1}}, \tag{4.3.28}$$

then m will be a root over integers for $w(x)$. Hence, to find a polynomial $w(x)$ over integers such that it has all m as its roots, it is enough to construct a linear combination of polynomials $g_i(xB)$ and $h_i(xB)$ that satisfies the above bound given by (4.3.13).

Let us construct a lattice having dimension $d = a + a'$.The lattice is spanned by the coefficient vectors of the polynomials $g_i(xB)$ for $i = 0, ..., a$ and $h_i(xB)$ for $i = 0, ..., a'$. For example: taking $a = 4$ and $a' = 3$, the resulting lattice L of dimension 7 is spanned by the rows of the matrix:

	1	x	x^2	x^3	x^4	x^5	x^6
$g_0(xB)$	P^4						
$g_1(xB)$	$-RP^3$	P^3B					
$g_2(xB)$	R^2P^2	$-2RP^2B$	P^2B^2				
$g_3(xB)$	$-R^3P$	$3R^2PB$	$-3RPB^2$	PB^3			
$h_0(xB)$	R^4	$-4R^3B$	$6R^2B^2$	$-4RB^3$	B^4		
$h_1(xB)$		R^4B	$-4R^3B^2$	$6R^2B^3$	$-4RB^4$	B^5	
$h_2(xB)$			R^4B^2	$-4R^3B^3$	$6R^2B^4$	$-4RB^5$	B^6

Now, using the LLL algorithm we can get a short vector that can be assumed as the coefficient vector of the polynomial $w(xB)$.

The LLL algorithm involves a parameter $\frac{1}{4} < \delta < 1$, and it produces a short vector of length $\leq (\frac{2}{\sqrt{(4\delta-1)}})^{\frac{d-1}{2}} det(L)^{\frac{1}{d}}$.While we cannot take $\delta = 1$ in the LLL algorithm, for the purpose of our estimates below we use $\delta = 1$, resulting to a bound of $(\frac{4}{3})^{\frac{d-1}{4}} det(L)^{\frac{1}{d}}$ for the shortest vector produced. The constant 4/3 persists through the equations up to 4.3.45, where we use an inequality (rounding up), which is equivalent to taking a δ slightly less than 1.

Therefore, $w(xB)$ satisfies

$$\|w(xB)\| \leq (\frac{4}{3})^{\frac{d}{4}}.det(L)^{1/d} \tag{4.3.29}$$

now if

$$(\frac{4}{3})^{\frac{d}{4}} det(L)^{1/d} < \frac{M^a}{\sqrt{d}} \tag{4.3.30}$$

It guaranteed that LLL will produce a polynomial $w(xB)$ such that

$$\|w(xB)\| < \frac{M^a}{\sqrt{d}} \tag{4.3.31}$$

satisfying 4.3.28. Rearranging 4.3.30 we get

$$M > det(L)^{1/ad} \gamma(d)^{1/a} \tag{4.3.32}$$

where $\gamma(d) = (\frac{4}{3})^{\frac{d}{4}} \sqrt{d}$

Note that the determinant of L is simply the product of the elements on the diagonal, as is clear from the above example that the basis of L is a triangular matrix.

$$det(L) = P^{\frac{a(a+1)}{2}} B^{\frac{d(d-1)}{2}}$$

4.3.32 then implies

$$M > P^{\frac{a(a+1)}{2}} B^{\frac{d(d-1)}{2}} \gamma(d)^{1/a} = P^{\frac{1}{2d}} P^{\frac{a}{2d} + \frac{1}{a}.(\frac{\log_2(B)}{\log_2(P)} \frac{d-1}{2} + \frac{\log_2(\gamma(d))}{\log_2(P)})} \tag{4.3.33}$$

For simplicity just replace $C = \frac{1}{2d}$ and $D = \frac{\log_2(B)}{\log_2(P)}.\frac{d-1}{2} + \frac{\log_2(\gamma(d))}{\log_2(P)}]$ Then we want an integer $a \in [1, ..., d]$ that minimizes R.H.S of (4.3.18) that is

$$P^{C+aC+\frac{D}{a}} \tag{4.3.34}$$

taking the derivative of this exponent and replacing values for D and C

$$a_{opt} = \sqrt{\frac{D}{C}} = d\sqrt{\frac{(1-\frac{1}{d})\log_2(B)}{\log_2(P)} + \frac{2\log_2(\gamma(d))}{d\log_2(P)}} \tag{4.3.35}$$

$$= d\sqrt{\frac{(\frac{d-1}{d})\log_2(B)}{\log_2(P)} + \frac{2\log_2(\gamma(d))}{d\log_2(P)}} \tag{4.3.36}$$

By assumption we have:

$$d \geq 1 + \sqrt{\frac{\log_2(P)}{\log_2(B)}}$$

then also

$$d > \sqrt{\frac{\log_2(P)}{\log_2(B)}} \tag{4.3.37}$$

using this bound for d in 4.3.36 gives

$$a_{opt} \geq d\sqrt{\frac{\sqrt{\frac{\log_2(P)}{\log_2(B)}}\log_2(B)}{d\log_2(P)} + \frac{2\log_2(\gamma(d))}{d\log_2(P)}} \tag{4.3.38}$$

taking d inside the square root and canceling

$$\geq \sqrt{d\sqrt{\frac{\log_2(B)}{\log_2(P)}} + \frac{2d\log_2(\gamma(d))}{\log_2(P)}} \tag{4.3.39}$$

using 4.3.37

$$> \sqrt{\sqrt{\frac{\log_2(P)}{\log_2(B)}}\sqrt{\frac{\log_2(B)}{\log_2(P)}} + \frac{2d\log_2(\gamma(d))}{\log_2(P)}} \tag{4.3.40}$$

$$= \sqrt{1 + \frac{2d\log_2(\gamma(d))}{\log_2(P)}} \tag{4.3.41}$$

Hence, $a_{opt} > 1$

Let $[a_{opt}]$ be the closest integer to a_{opt}.

$$a_{opt} - \frac{1}{2} \leq [a_{opt}] \leq a_{opt} + \frac{1}{2} \tag{4.3.42}$$

$$\frac{1}{[a_{opt}]} \leq \frac{1}{a_{opt} - \frac{1}{2}} \leq \frac{1}{a_{opt}} + \frac{1}{a_{opt}^2}$$

Note that the second inequality in the above equation holds as:

$$\frac{1}{X - \frac{1}{2}} \leq \frac{1}{X} + \frac{1}{X^2}$$

$$\Leftrightarrow X^2 \leq (X - \frac{1}{2}) * (X + 1)$$

$$\Leftrightarrow X^2 \leq X^2 + \frac{X}{2} - \frac{1}{2}$$

$$\Leftrightarrow 0 \leq \frac{X}{2} - \frac{1}{2}$$

$$\Leftrightarrow X \geq 1$$

as $a_{opt} > 1$, so the above inequality holds.

Therefore, exponent of P in 4.3.34 becomes,

$$C + [a_{opt}]C + \frac{D}{[a_{opt}]} \leq C + [a_{opt} + \frac{1}{2}]C + [\frac{1}{a_{opt}} + \frac{1}{a_{opt}^2}]D$$

substituting value of a_{opt}

$$= C + [\sqrt{\frac{D}{C}} + \frac{1}{2}]C + [\sqrt{\frac{C}{D}} + \frac{C}{D}]D$$

$$= 2\sqrt{CD} + \frac{5}{2}C$$

substituting values for C and D gives

$$= \sqrt{(1 - \frac{1}{d})\frac{\log_2(B)}{\log_2(P)} + \frac{2\log_2(\gamma(d))}{d\log_2(P)}} + \frac{5}{4d} = \epsilon \tag{4.3.43}$$

So, (4.3.32) is satisfied if $M > P^{\epsilon}$ Note that $\frac{2\log_2\gamma(d)}{d} < 0.75$ for all $d > 0$. because

$$\frac{2\log_2(\gamma(d))}{d} = \frac{2\log_2((\frac{4}{3})^{\frac{d}{4}}\sqrt{d})}{d} \tag{4.3.44}$$

after applying the properties of the log function,

$$= \frac{2(\frac{d}{4}\log_2(\frac{4}{3}) + \frac{1}{2}\log_2(d))}{d}$$

cancellation gives,

$$\frac{1}{2}\frac{\log_e(\frac{4}{3})}{\log_e(2)} + \frac{\log_e(d)}{(\log_e(2))d}.$$

Its derivative is

$$\frac{1}{\log_e(2)}(\frac{1}{d^2} - \frac{\log_e(d)}{d^2}),$$

equating it to 0 for finding maximum value point of expression in (4.3.44). Maximum value occurs at point $d = \mathrm{e}$ and the value is approximately 0.73826, implies

$$\frac{2\log_2(\gamma(d))}{d} < 0.73826 \tag{4.3.45}$$

for all $d > 0$. Furthermore, we can simplify the bound by dropping the $(1 - \frac{1}{d})$ factor in (4.3.43) and we get

$$M > P^{\epsilon}$$

where

$$\epsilon = \sqrt{\frac{0.73826 + \log_2(B)}{\log_2(P)}} + \frac{5}{4d} \tag{4.3.46}$$

To sum up, once we got $w(xB)$ such that it satisfies (4.3.28) when $M > P^{\epsilon}$ then $w(x)$ will have property: if $0 < m < B$ and $amp(m) > M$ then m is a root of $w(x)$ over the integers.

Thus, we evaluate $w(x)$ and then use some root-finding algorithm in $\mathbb{R}$ to find all integer roots of $w(x)$ and output the roots m that satisfy $amp(m) > P^{\epsilon}$.

Now observe that

$$\sqrt{0.73826 + \log_2(B)} < \sqrt{\log_2(2) + \log_2(B) - \frac{1}{4}} < \sqrt{\log_2(2B)} - \frac{\frac{1}{8}}{\sqrt{\log_2(2B)}} \tag{4.3.47}$$

Observe that if $d > 20\log_2(P)$, then it follows that $d > 10\sqrt{\log_2(P)\log_2(2B)}$ as $P > B \geq 1$

Hence,(4.3.46) can be manipulated as

$$\epsilon = \sqrt{\frac{0.73826 + \log_2(B)}{\log_2(P)}} + \frac{5}{4d} < \sqrt{\frac{\log_2(2B)}{\log_2(P)}} - \frac{\frac{1}{8}}{\sqrt{\log_2(P)\log_2(2B)}} + \frac{5}{4d} \tag{4.3.48}$$

using $d > 10\sqrt{\log_2(P)\log_2(2B)}$ implies

$$\epsilon < \sqrt{\frac{\log_2(2B)}{\log_2(P)}}$$

This completes the proof.

– – – – ––

Smooth numbers in a given interval

This section links the above 4.2 to find all strongly smooth numbers in a given interval.

Suppose s is a given fixed positive number. Let $q_1, q_2, q_3 \ldots, q_n$ be all primes less than s.Note that n is the number of such primes and assume that they are sorted in ascending order, which means $q_i < q_j$ for $i < j$. Define $S = \Pi_{i=1}^{n} q_i^{\alpha_i}$ where $\alpha_i > 0, q_i^{\alpha_i} \leq s$. Let $I = [U, V]$ be a given interval such that $V < 2U$ and denote $|I|$ as the length of the interval given as $|I| = V - U$.

We consider the previous theorem (4.2) with $P = S$,$B = |I|$ interval size, $< p_1, p_2, \ldots, p_n >=< q_1^{\alpha_1}, q_2^{\alpha_2}, \ldots, q_n^{\alpha_n} >$ and $< r_1, r_2, \ldots, r_n >=< -U, -U, \ldots, -U >$.

4.4 Boneh's second theorem

Let interval $I = [U, V]$, s, $T > 0$ be integers. For any integer $d \geq 1 + \sqrt{\frac{log_2(S)}{log_2(|I|)}}$ there is a polynomial-time algorithm that outputs all integers $N \in I$ whenever

$$\gcd(N, S) > T$$

, where

$$T > S^{\epsilon} \text{ and } \epsilon = \sqrt{\frac{log_2(4|I|)}{log_2(S)}} + \frac{5}{4d}.$$

Solving for $|I|$ and simplifying, gives:

$$|I| \leq \frac{1}{4} T^{\frac{log(T)}{log(S)} - \frac{2.5}{d}}.$$

In addition, if $V < 2T$ and $|I|$ is sufficiently small so that $T > S^{\epsilon}$ then the algorithm will output all s strongly smooth integers in $[U, V]$.

Proof: Lets take $|I|, < q_1^{\alpha_1}, q_2^{\alpha_2}, \ldots, q_n^{\alpha_n} >, < -U, -U, \ldots, -U >$ where q_i's are prime numbers and $\alpha_i \in \mathbb{N}$ such that $\alpha_i = \lfloor \frac{\log(s)}{\log(q_i)} \rfloor$, hence $q_i^{\alpha_i}$ are co-prime for all $i \in \{1, 2, ..n\}$.

Now using 4.2 gives all $0 < m < |I|$ such that $amp(m) > S^{\epsilon}$.

It means we have all $N = U + m$,

$$\text{such that } U < U + m < U + |I| = V \text{ and } gcd(U + m, S) > S^{\epsilon}.$$

This proves one part of the theorem for any $T > S^{\epsilon}$.

We have $T > S^{\epsilon}$ where $\epsilon = \sqrt{\frac{\log_2(4|I|)}{\log_2(S)}} + \frac{5}{4d}$,

$$\implies T > S^{\sqrt{\frac{\log_2(4|I|)}{\log_2(S)}} + \frac{5}{4d}}, \tag{4.4.1}$$

Taking log_2 on both sides,

$$\implies \log_2(T) > [\sqrt{\frac{\log_2(4|I|)}{\log_2(S)}} + \frac{5}{4d}] \log_2(S), \tag{4.4.2}$$

$$\implies [\frac{\log_2(T)}{\log_2(S)} - \frac{5}{4d}]^2 > \frac{\log_2(4|I|)}{\log_2(S)}, \tag{4.4.3}$$

$$\implies \log_2(S)[(\frac{\log_2(T)}{\log_2(S)})^2 - (\frac{5}{2d})(\frac{\log_2(T)}{\log_2(S)}) + \frac{25}{16d^2}] > \log_2(4|I|). \tag{4.4.4}$$

Now raising everything to a power of 2 on both sides of inequality,

$$\implies 2^{\frac{(\log_2(T))^2}{\log_2(S)} - (\frac{5}{2d})(\log_2(T)) + \frac{25}{16d^2}(\log(S))} > 2^{\log_2(4|I|)}, \tag{4.4.5}$$

$$\implies T^{[\frac{\log(T)}{\log(S)}]-[\frac{5}{2d}]}S^{[\frac{25}{16d^2}]} > 4|I|, \tag{4.4.6}$$

$$\implies \frac{T^{[\frac{\log(T)}{\log(S)}]-[\frac{5}{2d}]}S^{[\frac{25}{16d^2}]}}{4} > |I|. \tag{4.4.7}$$

As $S > 0$ and $a > 0$ therefore $S^a > 1$ hence above bound on $|I|$ is still satisfied if we take:

$$\frac{T^{[\frac{\log(T)}{\log(S)}]-[\frac{5}{2d}]}}{4} > |I|. \tag{4.4.8}$$

Now, suppose $V < 2T \implies N < 2T$ as $N \in I$. As we have $gcd(N, S) > T$,

$$\frac{V}{2} < T < gcd(N, S) \leq N < V \implies \frac{V}{2} < gcd(N, S) < V. \tag{4.4.9}$$

This means $gcd(N, S) = N$ because otherwise, it misses at least one prime factor of N (say 2, smallest prime). So,

$$gcd(N, S) \leq \frac{N}{2}.$$

But that contradict 4.4.9 as

$$\frac{N}{2} < \frac{V}{2}.$$

Hence, we conclude $gcd(N, S) = N$, which means $N|S$ and therefore N is s strongly smooth number.

4.5 Example

Input values:

$T = 2^{1500}$

$s = 1500$

$d = 5$

we got an upper bound for $|I| \leq 4.878585220 \times 10^{84}$.It is necessary to verify that $d = 5$ is consistent with this value of $|I|$ according to the bounds stated in the theorem, which shows that any $d > 4$ can work. Setting:

$U = 1329262360453049482949407708107073915852803708202770302802119617223514698032537464579451479686480209288375176729012726520349781065744636058537033093584609335001316258605646535202883097362326164959757831630795462126608999682396706088139699320575754012571045906755822135902219695559885967914095362328284156052328801261119047101982641412898540777662035297752243142687701954489505800722016435880472061836342831263551621506324488152900703550911556724781803895 92$

$|I| = 10^{80}$ (within the bound)

Running the algorithm outputs:-

$a_{opt} = 2$

$a' = d - a_{opt} = 3$

And these polynomials from the LLL algorithm:

$-7\times(10142605542130662305190583924546381181747839183582396361248140460915501435167047275431501624075177674271040488789569x^2-8140654528867401364265563125808244704574229722990600873628326575481322094866891400792813075292888492961828201260056064367606965434686117877744179568840050022877578139573499814476313177010905675 04x+8871625362757040455313811812587452065636089916440423471078763817688162702926118751022249726913599377101158712306131243401424068126002667260599868729019209888639938880022433220768861185971820604433406428626025462395299955265207891636395009221573940304419691832542524944727616)\times(x-9345676903215646577898775334354656767223456778)^2$

$2\times(9300654868036883474619269746731296899557798258870172864678578566495044555172570831849091297861689043560147172760395 7x^2+9110479120743614189263540067713219906349578896462592793035395451826963662239939323938961519096839277877053670895045731994324803795976353349801389056603680325639893334767987605451391338829547458288x-7445742645801409068686688577373505365716951165590966634199686609385169079375171460342801665562360370085229992070862752888328331083470974701250726247173512093834348664500582492359076819894392091817596498176698796057505200015336484440161727019325530486165282743102713398160795 52)\times(x-9345676903215646577898775334354656767223456778)^2$

$$5\times(4161844578784983702442857747757251302392466004779448025180631998740413560637946818713738596360195281821494478218893 3x^2+6961302683863157665081204280922654870587446825368736368429904920190180360242443966993964927044852738796693358492930609421490817240219791479368233032554076018186017124150463184174034814241697057392x+1931468468914614427307303963196354803448097265015238768784514221392071066243481612373493739193547490237114091143676084355248221265969392564183742050491063135213785647310330256409590626398193474671119980560712953280453297702830809412422386604838355724870345402894310276931641152) \times(x - 9345676903215646577898775334354656767223456778)^2$$

$$-(\mathrm{x} - 9345676903215646577898775334354656767223456778)\times(1695104560165156450525789742576096919801304666946439456389950619523813734273133758591736903564594284712635369639169943854694943x^3+1360338509984978394902111868041783760325770604713918764875038742707636049313138873907122864503978743254860982216193243861163819314037321937473717605330221851525328135220945664886898062916195459857582542483429886845198778544031269657778030913x^2+9703870028696834491462820052125009887683346868856450194050718791828958610013754670125611523738176344628854398975109310626932336974460335689317298270691096924179566262930249002598160998037783257404793903242026432513321204310386840842236566559550175736453271162295656660462588410044696310088610555250389509724215008777564070769933128345468858239056751809613x+5191635731253421224567696133396170845311756568482987996889821076226085948213681233858637340483294978546818095854588711159282119232186710253173555400995756760353482145002585690988772509338032565147290026791586053733856229329643712218133412415824131054994277059422342648985287293765398903225883598549145592044324793469662960396057837337249073803101404710336506317224093465861585685165893461832371807430252482967239981092755360701958023079020626105306109109652576839648 0)$$

$$(1347921079913430479793416707345984348052327220959155359778947246808083044489467286587940740931532570624808950392235977835093289166952045094406971093973897940098711258561273190855852570791898949728726964962092525065920696459950857063642198306217406489464014909111968779816683613044438529100877228225168336194345215249548080476922545711355852217324608359796721771015200164801646996646624328457514336998104276739753971632097782443800761854185810305813132934 3 \mathrm{x}^4+1442293687094660638086387785005128996949635467889085856627381168094491539236670597701938477 7$$

$$
\begin{aligned}
&319372115627550506565954712037098681335211865683567815356984691610862996594130412829004047403190699643919576152604884062010572810510516698826043869354535831765219690338053843456008864331094941566848458061694126926960332442101563035110639529657589213924295170910473166931070233227791969878009368309029738689281344641392256565854014670838974297224602712753269390641098104782860934264852047388636168381776673084792619655547162776959109484550189196371190381571678236550868705400275x^3+154327364623358461724602166010788848827051259705097476587334016467579551171642716803164309797329460748582422711422652720906781022724931069137171712304009680672847082898131072523761335067091110608071499397830581729572002105219377547527906602730951436531351291160046452660124293235824553153077538764110291500569717814529971349146453224897627458345733719640209233485639829413662106070137236459930139632317251024247075242930809727395534623900608011323755975873737047158459707571857755593197961566425402377085355980445750326175138734331222965148061548751785387452334054773881536459705033606641676620124041671854856110055656756117062185990258247886797420479945945131325195887171318698729192+165132356084616762377282886514066490063472155984319366912997247207292230483669570794224819427836744817988917054325164847328893194376094496275643986865479542983071698220932036181042038063844214982175274695684833774761138458240731031478744203502241577838919472998385485476341156355022367145530454100522924083859301399690926367912758065713380477176235670438934553032532640527995082174717050836457606239972657236107533374107321612708310988746836384527738529431911794552499772100688016937472635184515600899832022731619029738798018209777133954132481286025898496340084798101958668160326859433878124437193738972759738531370146652347864935091115262125569266006946396876260812938983076373950160228003433575137729976640687085576442301894528524173161594302142121898975528445443942921568391440789112680540123 2x+176693842291721285000947273536192607779268761200350710506346403549804223355918045410842019551770198898298394441303469778192843218598415996988108307946067762605126691805026799161180434145549153432087528259495518524110841610438185969010556592496893810245675532045909780815967078983683993464180808921780361873343468970405628914526200273704177164914317525070669676942022695997434456748955566995685336745694752980093821365737136665038692589140632177966205507794126694333495970580351486811088407675524058559709696360187712674058683168002719499598283219137495267594496321551664912472410810540716329194666674977508266145532890889268299382285687664370818590319684134863189262983814547334071508258449358576556631374157168408983623773277020266067444920035702643977300410690813147513071495503458093273835233574448938931523632693671760741552981953750713283599932322139397976155753005163484456098231923482538623681878795360 0)
$$

The first polynomial, as well as most of the others, has a linear factor corresponding to:

$m = 934567690321564657789877533435465676722345678$

$gcd(U+m, S) = 13292623604530494829494077081070739158528037082027703028021196$
$172235146980325374645794514796864802092883751767290127265203497810657446360585$
$703309358460933500131625860564653520288309736232616495975783163079546212660899$
$682396706088139699320575754012571045906755822135902219695559885967914095362328$
$284156052328801261119047101982641412898540777662035297752243142687701954489505$
$807220164358804720618363428312636450782753566446186796913042551032924540384637$
$0 (\approx 2^{1512})$

$S^{\epsilon} = 1.557851971 \times 10^{277} (\approx 2^{921})$

Theorem conditions hold as $gcd(U + m, S) > T > S^{\epsilon}$

factors of $(U + m)$ = 2, 3, 5, 7, 11, 13, 17, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 149, 151, 157, 163, 167, 173, 179, 181, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 263, 269, 271, 277, 281, 293, 307, 313, 317, 331, 337, 347, 349, 353, 359, 373, 379, 383, 389, 397, 409, 419, 421, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 631, 641, 643, 647, 653, 659, 673, 677, 683, 691, 701, 709, 719, 733, 739, 743, 751, 757, 761, 769, 787, 797, 809, 811, 823, 827, 839, 853, 857, 859, 863, 881, 883, 887, 907, 911, 919, 929, 937, 941, 953, 977, 983, 991, 997, 1123, 1129, 1151, 1153, 1163, 1171, 1181, 1187, 1193, 1201, 1213, 1217, 1223, 1229, 1231, 1237, 1249, 1259, 1277, 1279, 1283, 1289, 1291, 1297, 1301, 1303, 1307, 1319, 1321, 1327, 1493, 1499.

Disclaimer: It is not a natural example. We manually choose an interval $[U, V]$ that includes a 1500-strongly smooth number that we precomputed. The reason behind this is that the probability of finding a smooth number is very small and that of strongly smooth numbers is even smaller. Thus, most of the time, the algorithm will return empty-handed, unless one works with an interval in which one knows that a strongly smooth number lives.

Chapter 5

Observations

5.1 Graphs and Tables

We varied the smoothness bound s, and interval length $|I|$, to get some numerical intuition on how large N needs to be (for specific s, $|I|$ and d) in order for Boneh's algorithm to output N if it is s-strongly smooth. There is some flexibility in the choice of d, so we also experimented with using several different values of d.

Since we are working here with specific choices of s, I, and d, we can refine ϵ in the statement of the theorem and compute a more precise bound directly from equations 4.2.1 -4.2.18.

For N's that are strongly s-smooth, it follows that $N = \gcd(N, S)$. Thus, for such N, we have $N \geq S^{\epsilon}$ and, by computing ϵ, we can determine how large N needs to be. This is what our plots depict.

Here $d_0 = \lceil (1 + \sqrt{\frac{\log_2(S)}{\log_2(|I|)}}) \rceil$ and $S = \Pi_1^n p_i^{a_i}$ for all primes $p_i \leq s$ where $a_i = \lfloor \frac{\log_2(s)}{\log_2(p_i)} \rfloor$

Results for $s = 10$

$I = 10^r$	$N = 10^k$ for $d = d_0$	$N = 10^k$ for $d = 100d_0$	$N = 10^k$ for $d = 7d_0$
10^1	10^3	10^3	10^3
10^2	10^4	10^3	10^3
10^3	10^4	10^4	10^4
10^5	10^5	10^5	10^5
10^{10}	10^6	10^6	10^6
10^{15}	10^7	10^8	10^8
10^{20}	10^7	10^9	10^9
10^{25}	10^8	10^{10}	10^{10}
10^{30}	10^9	10^{11}	10^{10}
10^{35}	10^9	10^{11}	10^{11}
10^{40}	10^{10}	10^{12}	10^{12}
10^{45}	10^{10}	10^{13}	10^{13}
10^{50}	10^{11}	10^{14}	10^{13}

Table 5.1: Results for $s = 10$

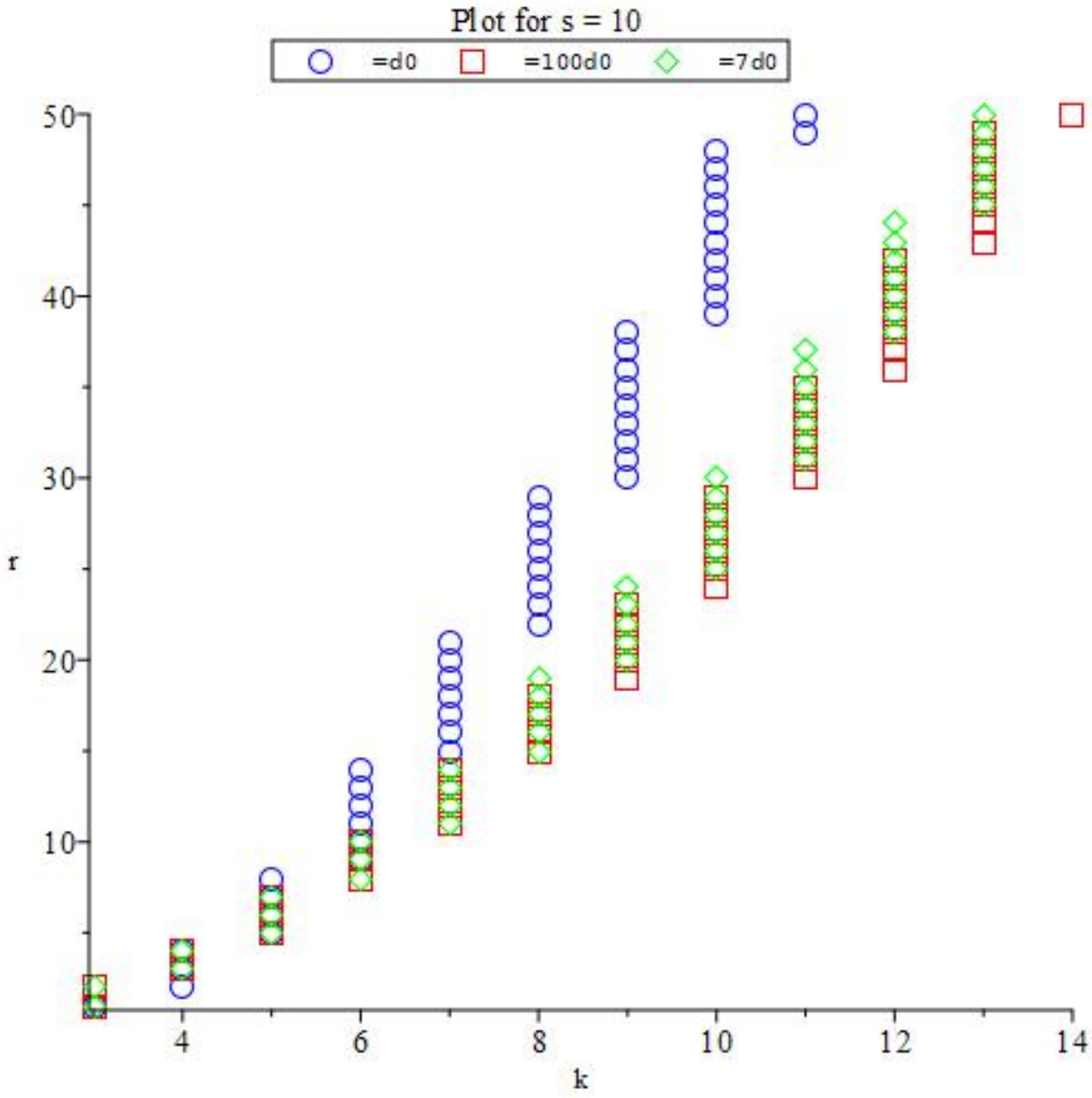

Figure 5.1: Graph for $s = 10$

Results for $s = 100$

$I = 10^r$	$N = 10^k$ for $d = d_0$	$N = 10^k$ for $d = 100d_0$	$N = 10^k$ for $d = 7d_0$
10^1	10^{11}	10^8	10^8
10^2	10^{14}	10^{10}	10^{11}
10^3	10^{16}	10^{12}	10^{13}
10^5	10^{19}	10^{15}	10^{16}
10^{10}	10^{24}	10^{21}	10^{21}
10^{15}	10^{29}	10^{26}	10^{26}
10^{20}	10^{31}	10^{29}	10^{30}
10^{25}	10^{34}	10^{33}	10^{33}
10^{30}	10^{36}	10^{36}	10^{36}
10^{35}	10^{39}	10^{38}	10^{39}
10^{40}	10^{41}	10^{41}	10^{41}
10^{45}	10^{44}	10^{44}	10^{43}
10^{50}	10^{44}	10^{46}	10^{46}

Table 5.2: Results for $s = 100$

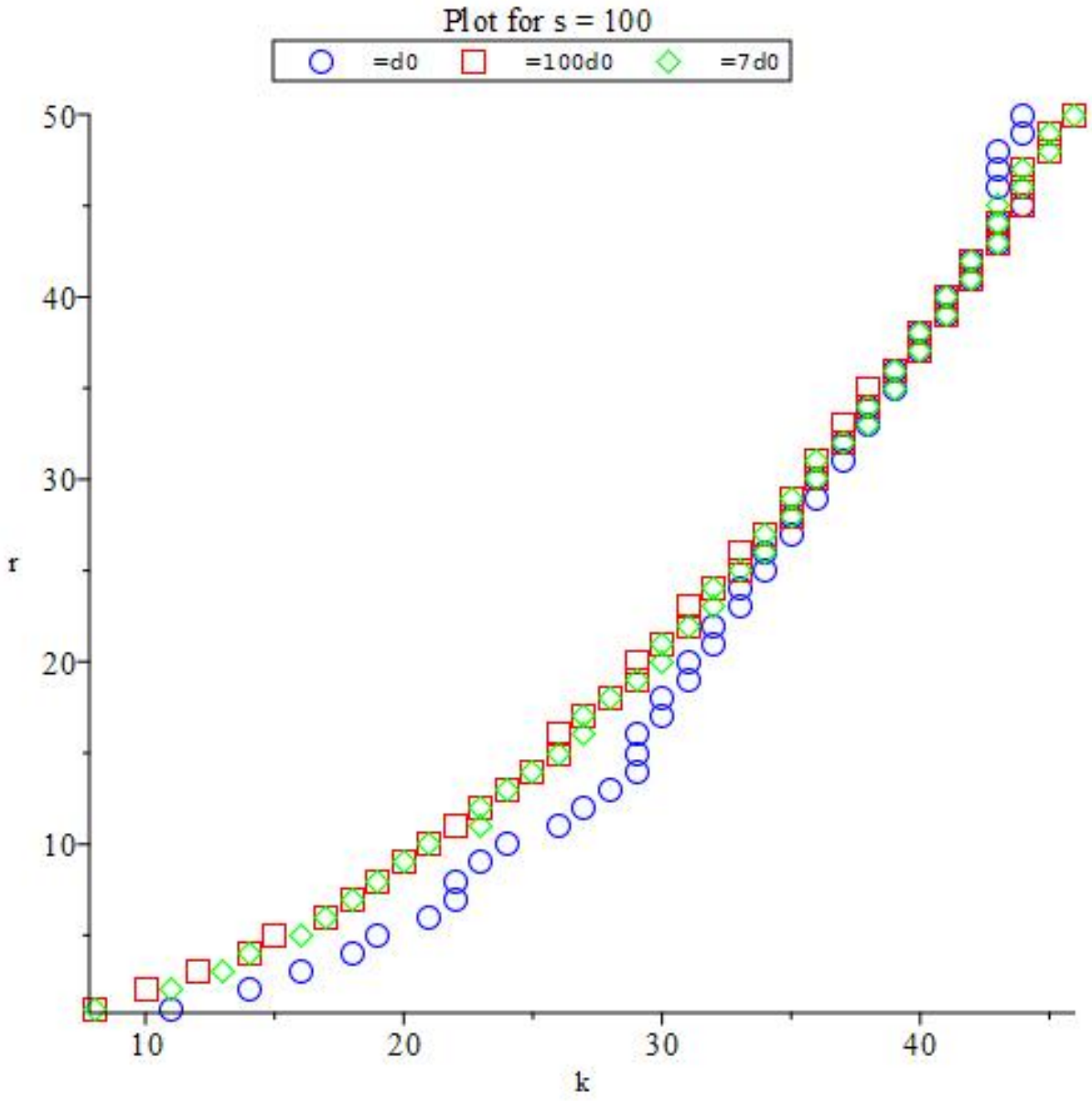

Figure 5.2: Graph for $s = 100$

Results for $s = 1000$

$I = 10^r$	$N = 10^k$ for $d = d_0$	$N = 10^k$ for $d = 100d_0$	$N = 10^k$ for $d = 7d_0$
10^1	10^{35}	10^{24}	10^{26}
10^2	10^{46}	10^{32}	10^{34}
10^3	10^{54}	10^{38}	10^{41}
10^5	10^{67}	10^{49}	10^{51}
10^{10}	10^{91}	10^{68}	10^{71}
10^{15}	10^{109}	10^{82}	10^{86}
10^{20}	10^{124}	10^{95}	10^{98}
10^{25}	10^{136}	10^{105}	10^{109}
10^{30}	10^{148}	10^{115}	10^{120}
10^{35}	10^{158}	10^{124}	10^{129}
10^{40}	10^{168}	10^{133}	10^{137}
10^{45}	10^{178}	10^{141}	10^{145}
10^{50}	10^{185}	10^{148}	10^{153}

Table 5.3: Results for $s = 1000$

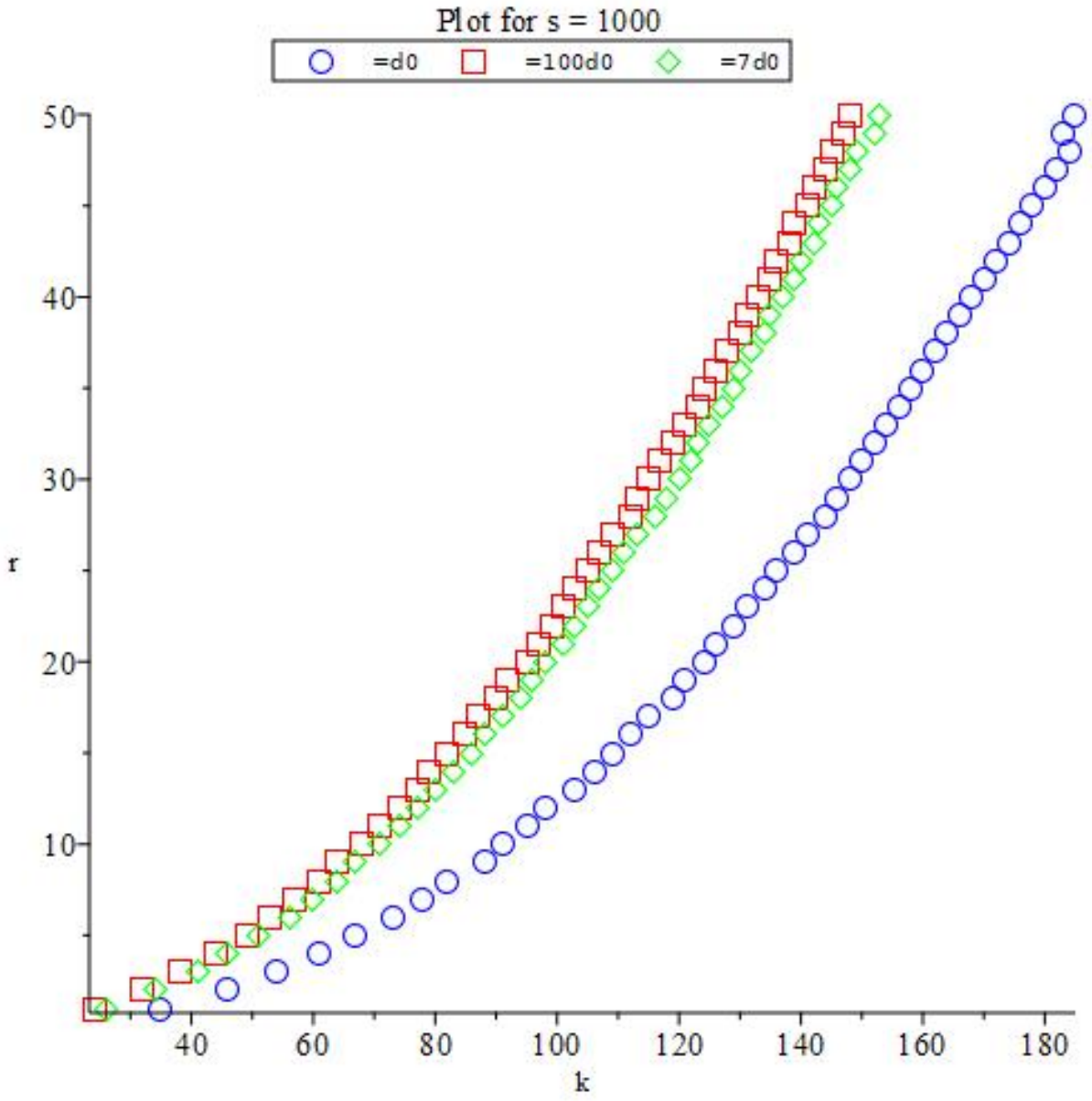

Figure 5.3: Graph for $s = 1000$

Results for $s = 10000$

$I = 10^r$	$N = 10^k$ for $d = d_0$	$N = 10^k$ for $d = 100d_0$	$N = 10^k$ for $d = 7d_0$
10^1	10^{109}	10^{76}	10^{80}
10^2	10^{146}	10^{101}	10^{107}
10^3	10^{175}	10^{121}	10^{128}
10^5	10^{221}	10^{153}	10^{162}
10^{10}	10^{307}	10^{213}	10^{226}
10^{15}	10^{368}	10^{260}	10^{274}
10^{20}	10^{425}	10^{299}	10^{316}
10^{25}	10^{468}	10^{334}	10^{351}
10^{30}	10^{509}	10^{365}	10^{384}
10^{35}	10^{548}	10^{394}	10^{414}
10^{40}	10^{585}	10^{421}	10^{443}
10^{45}	10^{623}	10^{446}	10^{470}
10^{50}	10^{648}	10^{470}	10^{494}

Table 5.4: Results for $s = 10000$

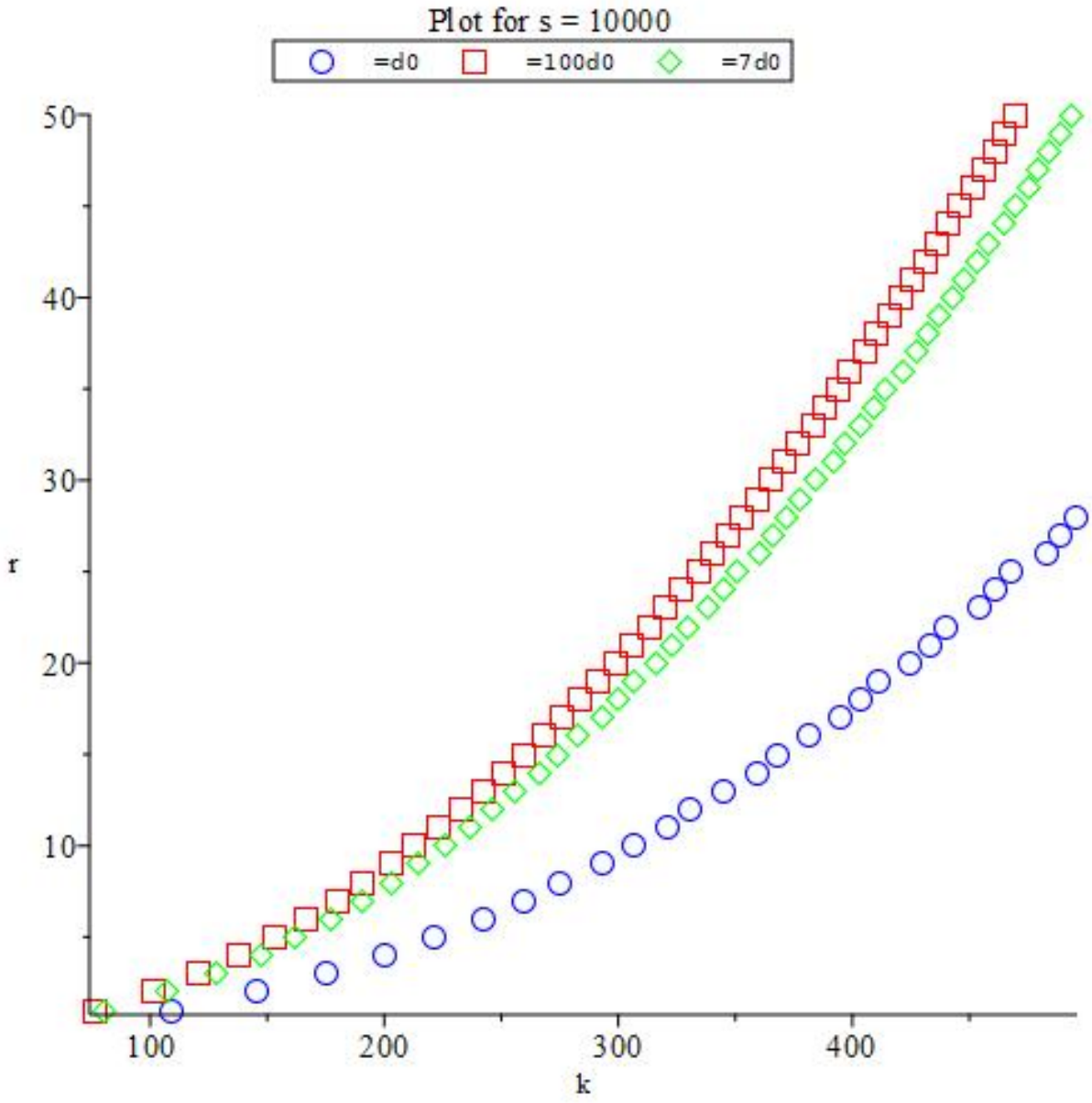

Figure 5.4: Graph for $s = 10000$

Results for $s = 100000$

$I = 10^r$	$N = 10^k$ for $d = d_0$	$N = 10^k$ for $d = 100d_0$	$N = 10^k$ for $d = 7d_0$
10^1	10^{345}	10^{239}	10^{253}
10^2	10^{464}	10^{318}	10^{338}

Table 5.5: Results for $s = 100000$

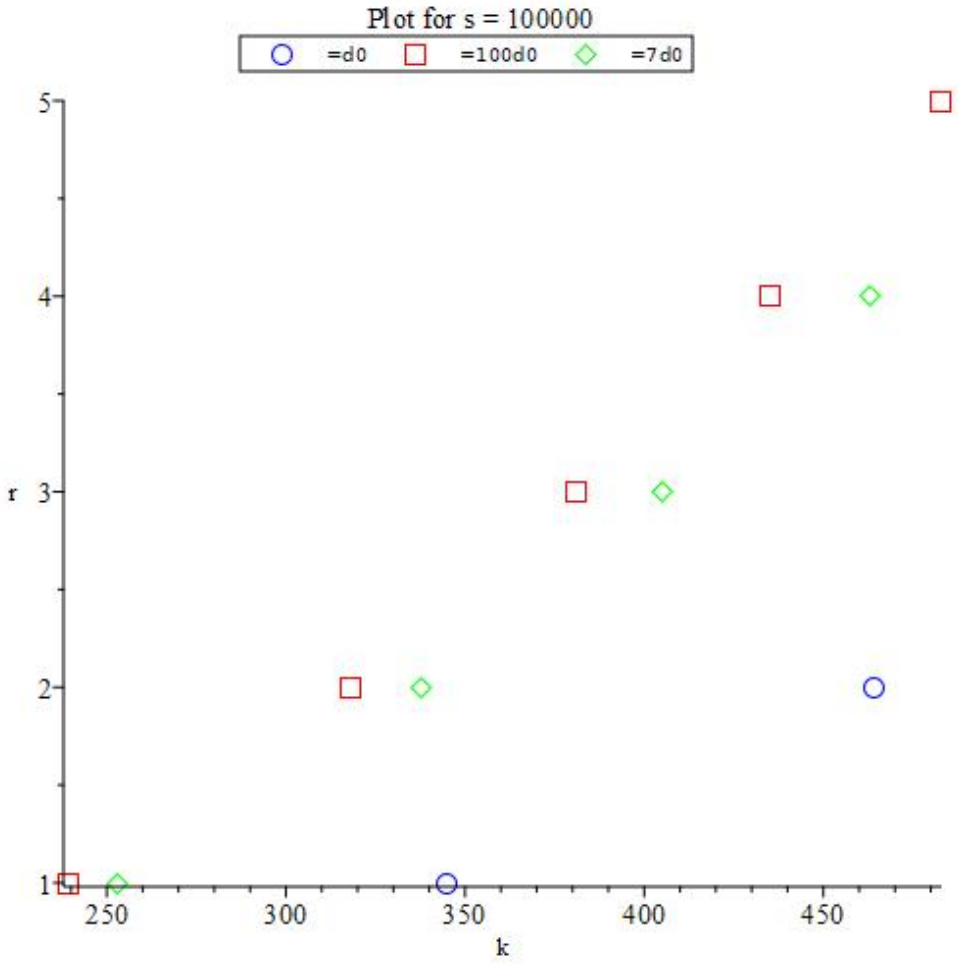

Figure 5.5: Graph for $s = 100000$

Example

Based on the table for $s = 1000$, we ran another example.

Choosing parameters from the above tables:

$s = 1000$

$d = d_0 = 4$

$|I| = 10^{50}$

$U = 722279208856126761637957277201561515588421090431095815193215043605763$
$030831604728869388156227509394599381215217784408534360849522408757901625770 1$
$427688031019135094978102215348640529852458411 6$

Running the algorithm outputs:-

$a_{opt} = 1$

$a' = d - a_{opt} = 3.$

The LLL algorithm then yields the following polynomials:

$2\times(x-123456789123456789123456789123456789123456789)\times(417859967195003268583043145272x^2+126136136089483638110288587554366520551722379199900047452588591760321341991394047x-136636644554314032593382659283535588541205835064716061548683148754191967083480538969619262390468383113404057863308975535913702305 80)$

$-(x-123456789123456789123456789123456789123456789)\times(752983253131279096322913548729x^2+191085611576767988012626968179573899548077786401335248944734235985396119944317174x+384831088940697685621993373617829164726062902972892441461572536938670731710533218565613269606645187381239429111970742239066470627 60)$

$(x-123456789123456789123456789123456789123456789)\times(530011814732450335939880083389 6x^2-3682285584634845164665240109237344098932533084639014213087152832602466375198386 71x-491373889700363506594291348811987556201927826796822717206319638536836804701177093490131918183188489957217606259687239244498918412)$

$208305199569176357188699016901 2x^3-1747480117269404531430490592883053137957749216243934330843188011841070127148952 15x^2+891700467362438559866974692365491839832757185399168434811993977279679392046355000023945510610492674018925647287896756962$

$$949681357189260x+722279208855025896872379387114526106732980878562507907756786727$$
$$310623292734420299082006742298709091734614605396103026236849229036651123899254042$$
$$291048674190296600185525760869427392990989859 52$$

The first polynomial has a linear factor corresponding to:

$m = 123456789123456789123456789123456789123456789$

$gcd(U+m, S) = 722279208856126761637957277201561515588421090431095815193215043$
$605763030831604728869388156227509394599381215217784408534360849522408757901625 77$
$014400337099314807738904478942609862087648040905 \approx 7.223 \times 10^{190}$

$S^{\epsilon} = 1.307209324 \times 10^{184}$

Factors of $(U + m)$ = 5, 13, 17, 29, 37, 41, 53, 61, 73, 89, 97, 101, 109, 113, 137, 149, 157, 173, 181, 193, 197, 229, 233, 241, 257, 269, 277, 281, 293, 313, 317, 337, 349, 353, 373, 389, 397, 401, 409, 421, 433, 449, 457, 461, 509, 521, 541, 557, 569, 577, 593, 601, 613, 617, 641, 653, 661, 673, 677, 701, 709, 733, 757, 761, 769, 773, 797, 809, 821, 829, 853, 857, 877, 881, 929, 937, 941.

Disclaimer: Our purpose in this example was to serve as a sanity check for our tables. However, it is not a natural example. We manually choose an interval $[U, V]$ that includes a 1000-strongly smooth number that we precomputed. The reason behind this is that the probability of finding a smooth number is very small and that of strongly smooth numbers is even smaller. Thus, most of the time, the algorithm will return empty-handed, unless one works with an interval in which one knows that a strongly smooth number lives.

References

[1] Dan Boneh. Finding smooth integers in short intervals using crt decoding. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 265–272, 2000.

[2] Wieb Bosma. 4. LLL. https://www.math.ru.nl/~bosma/onderwijs/voorjaar07/compalg7.pdf, 2010. retrieved 28 February 2010.

[3] Nicolaas G de Bruijn. The asymptotic behaviour of a function occuring in the theory of primes. *Journal of the Indian Mathematical Society. New Series*, 15:25–32, 1951.

[4] Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische annalen*, 261:515–534, 1982.

[5] Hugh L Montgomery and Robert C Vaughan. *Multiplicative number theory I: Classical theory*. Number 97. Cambridge university press, 2007.

[6] Keegan Ryan and Nadia Heninger. Fast practical lattice reduction through iterated compression. In *Annual International Cryptology Conference*, pages 3–36. Springer, 2023.

APPENDICES

Appendix A

Maplesoft codes

A.1 Maplesoft code for Boneh's Algorithm

```
with*LinearAlgebra;
with(IntegerRelations);
with(MTM);

# Parameters for smoothness and interval setup
s := 1000;  # Smoothness bound
q := 2;    # Starting prime for the sequence
S := 1;    # Product of selected primes raised to powers
U := 2^600;  # Start of the interval
interval_length := 2^148;  # Length of the interval
V := U + interval_length;  # End of the interval
R := -U;  # Negative shift used in CRT decoding

# Construct the list of primes and their powers
p := [];  # List to store the powers of primes
while q < s do
 # Determine the highest power of q within the smoothness bound
    a := floor(evalf(log(s)/log(q)));
    S := S * q^a;  # Update the product S
    p := [op(p), q^a];  # Append the power of q to the list
    q := nextprime(q);  # Move to the next prime
```

```
end do;

# Calculation of parameters related to the lattice dimension
d := ceil(evalf(1 + sqrt(log2(S)/log2(interval_length))))+
45;  # Dimension parameter
A := 2^(d/2) * sqrt(d);  # Constant used in smoothness analysis
C := 1 / (2 * d);  # Constant derived from theorem
J := log(s) * (d - 1) / (2 * log(S)) + log(A) / log(S);
# Parameter used in optimization
aopt := sqrt(J / C);  # Optimal a value
Aopt := round(aopt);  # Round to nearest integer
Aprime := d - Aopt;  # Remaining dimensions for the lattice

# Interval and lattice matrix setup
B := interval_length;  # Interval length
L := Matrix(Aopt + Aprime, Aopt + Aprime, 0);
# Initialize lattice matrix with zeros

# Constructing the lattice matrix L using polynomials
for i from 0 to Aopt - 1 do
    g := expand(S^(Aopt - i) * (B * x - R)^i);
    # Polynomial expansion for basis vector
    for j from 0 to i do
        c := coeff(g, x, j);  # Extract the coefficient of x^j
        L[i + 1, j + 1] := c;  # Populate matrix L
    end do;
end do;

# Adding additional basis vectors to matrix L
for l from 0 to Aprime - 1 do
    h := expand((x * B)^l * (B * x - R)^Aopt);
    # Polynomial expansion for remaining basis vectors
    for k from 0 to l + Aopt do
        e := coeff(h, x, k);  # Extract the coefficient of x^k
        L[Aopt + l + 1, k + 1] := e;  # Populate matrix L
    end do;
end do;
```

```
print(L);  # Output the constructed lattice matrix

# Apply the LLL algorithm for lattice basis reduction
L2 := LLL(L);

# Process and output each reduced polynomial
for i to d do
    w := 0;   # Initialize polynomial w
    for j from 0 to d - 1 do
        w := w + L2[i, j + 1] * x^j / B^j;
        # Construct polynomial w using reduced lattice vectors
    end do;
    print(i, w);  # Output polynomial w
    print(i, factor(w));  # Output the factored form of polynomial w
end do;

# Calculation of the optimal smoothness measure(epsilon)
s_optimal := evalf(C * (1 + Aopt) + J / Aopt);
# Evaluate the optimal smoothness parameter
S_eps := S^s_optimal;  # Compute the adjusted smoothness bound
```

A.2 Maplesoft code used for plots

```
with(plots);
#looping over s values
for J to 5 do
    printf("================================================================================
    # to seperate the outputs
    s := 10^J;
    print(s);
    q := 2;
    S := 1;
    p := [];
    while q < s do
        a := floor(evalf(log(s)/log(q)));
```

```
    S := S*q^a;
    p := [op(p), q^a];
    q := nextprime(q);
end do;# Calculating S
pairs_1 := [];
#looping over interval length.
for r to 50 do
    interval_length := 10^r;
    d := ceil(evalf(1 + sqrt(log2(evalf(S))/log2(interval_length))));
    gamma_d := evalf(2^(d/2)*sqrt(d));
    C := 1/(2*d);
    logS := log(evalf(S));
    DD := evalf(log(interval_length)*(d - 1)/(2*logS) + log(gamma_d)/logS);
    aopt := sqrt(DD/C);
    Aopt := round(aopt);
    Aprime := d - Aopt;
    eps_optimal := evalf(C*(1 + Aopt) + DD/Aopt);
    S_eps := S^eps_optimal;
    #looping over N values
    for k to 500 do
        N := 10^k;
        # checking the theorem condition
        if S_eps < N then
            pairs_1 := [op(pairs_1), [k, r]];
            break;
        end if;
    end do;
end do;
#plotting results for d=d0
plot1 := pointplot(pairs_1, symbol = circle, symbolsize = 15, color = blue, labels
 # repeating process for different d value
pairs_2 := [];
# looping on the interval length
for r to 50 do
    interval_length := 10^r;
    d := 100*ceil(evalf(1 + sqrt(log2(evalf(S))/log2(interval_length))));
    gamma_d := evalf(2^(d/2)*sqrt(d));
    C := 1/(2*d);
```

```
    logS := log(evalf(S));
    DD := evalf(log(interval_length)*(d - 1)/(2*logS) + log(gamma_d)/logS);
    aopt := sqrt(DD/C);
    Aopt := round(aopt);
    Aprime := d - Aopt;
    eps_optimal := evalf(C*(1 + Aopt) + DD/Aopt);
    S_eps := S^eps_optimal;
    #looping over N values
    for k to 500 do
        N := 10^k;
        if S_eps < N then
            pairs_2 := [op(pairs_2), [k, r]];
            break;
        end if;
    end do;
end do;
#plotting for d=100d_0
plot2 := pointplot(pairs_2, symbol = box, symbolsize = 15, color = red, labels = ['
 # repeating process for another  different d value
pairs_3 := [];
#looping over interval length
for r to 50 do
    interval_length := 10^r;
    d := 7*ceil(evalf(1 + sqrt(log2(evalf(S))/log2(interval_length))));
    gamma_d := evalf(2^(d/2)*sqrt(d));
    C := 1/(2*d);
    logS := log(evalf(S));
    DD := evalf(log(interval_length)*(d - 1)/(2*logS) + log(gamma_d)/logS);
    aopt := sqrt(DD/C);
    Aopt := round(aopt);
    Aprime := d - Aopt;
    eps_optimal := evalf(C*(1 + Aopt) + DD/Aopt);
    S_eps := S^eps_optimal;
     #looping over N
    for k to 500 do
        N := 10^k;
        if S_eps < N then
            pairs_3 := [op(pairs_3), [k, r]];
```

```
                break;
            end if;
        end do;
    end do;
    #plotting for d=7d_0
    plot3 := pointplot(pairs_3, symbol = diamond, symbolsize = 15, color = green, label
    caption := cat("Plot for s = ", s);
    #plotting for all d values together
    p := display([plot1, plot2, plot3], legend = ["=d0", "=100d0", "=7d0"], legendstyle
end do;
```

A.3 Maplesoft Code for Table values case($d = d_0$)

```
with*LinearAlgebra;
with(IntegerRelations);
with(MTM);
#looping over s values
for J to 5 do
    printf("================================================================================
    #just to seperate output
    s := 10^J;
    print(s);
    q := 2;
    S := 1;
    p := [];
    while q < s do
        a := floor(evalf(log(s)/log(q)));
        S := S*q^a;
        p := [op(p), q^a];
        q := nextprime(q);
    end do; #calculating S
    #looping over interval length
    for r to 50 do
        interval_length := 10^r;
        d := ceil(evalf(1 + sqrt(log2(evalf(S))/log2(interval_length))));
        gamma_d := evalf(2^(d/2)*sqrt(d));
```

```
        C := 1/(2*d);
        logS := log(evalf(S));
        DD := evalf(log2(interval_length)*(d - 1)/(2*log2(S)) + log2(gamma_d)/log2(S));
        aopt := sqrt(DD/C);
        Aopt := round(aopt);
        Aprime := d - Aopt;
        eps_optimal := evalf(C*(1 + Aopt) + DD/Aopt);
        S_eps := S^eps_optimal;
        #looping over N
        for k to 700 do
            N := 10^k;
            #checking for theorem condition
            if S_eps < N then
                printf("s: %a interval 10^r: %a N 10^k:%a\n", s, r, k);
                break;
            end if;
        end do;
    end do;
end do;
```