Lattice cryptography: from cryptanalysis to New Foundations

Woerden, W.P.J. van

Part III

Basis Reduction

CHAPTER 6

Background on Basis Reduction

This chapter gives an introduction to different types of basis reduction algorithms, and the state-of-the-art of modelling their behaviour.

6.1 Introduction

Every $\mathbb{R}$-linear subspace $V \subset \mathbb{R}^n$ has an orthogonal basis, and many efficient algorithms in linear algebra rely on such a basis. Similarly, an orthogonal basis $\mathbf{B}$ of a lattice would make many problems efficiently solvable, e.g., solving CVP for $\mathbf{t} = \mathbf{Bx}$ would be as simple as rounding the coefficients $\mathbf{c} = \mathbf{B}\lfloor \mathbf{x} \rceil$. Any rank n lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$ has (for $n > 1$) an infinite number of other bases $\mathbf{B} \cdot \mathbf{U}$ for $\mathbf{U} \in \mathcal{GL}_n(\mathbb{Z})$, but for most lattices none of those are orthogonal. Still most lattices have close to orthogonal bases, which still allows us to do some algorithmic tasks efficiently. We call such a basis *well-reduced* or simply *good*. Basis reduction is the act of turning a (bad) basis into a good basis. A common way to measure the orthogonality of a basis is by

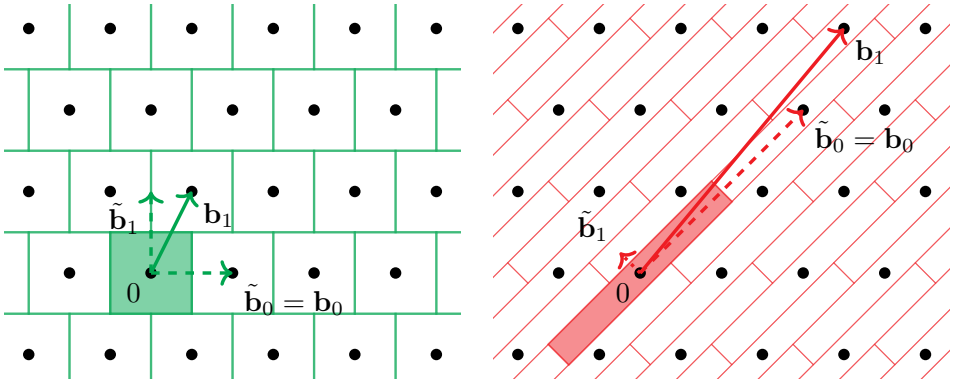

Figure 6.1: Babai's fundamental domain for a good basis (left) versus a bad (right) basis.

its orthogonality defect $\delta(\mathbf{B}) := \frac{\prod_i \|\mathbf{b}_i\|}{\prod_i \|\tilde{\mathbf{b}}_i\|}$, where $\tilde{\mathbf{b}}_i$ is the i-th Gram-Schmidt vector. We have $\delta(\mathbf{B}) \geq 1$ with equality if and only if the basis is orthogonal. Since we can rewrite $\delta(\mathbf{B}) = \frac{\prod_i \|\mathbf{b}_i\|}{\mathrm{vol}(\mathcal{L}(\mathbf{B}))}$, we see that to minimize the orthogonality defect a good basis must consist of short vectors. For a lattice basis orthogonality and shortness are thus directly connected.

To break most lattice based schemes we do not have to compute a basis with the lowest orthogonality defect, or consisting of the shortest vectors; a somewhat orthogonal basis is often sufficient. Once a certain orthogonality is reached we can use the good basis to decrypt a message, recover a secret key, or forge a signature. In part II of this thesis we have shown how to solve exact versions of the shortest and closest vector problem, in moderate dimensions. Running these algorithms on the high dimensional lattices common in cryptography is far beyond feasible, and would also find much shorter vectors than needed to break the scheme. In this chapter we show how to bootstrap these exact algorithms in moderate dimensions, to reduce a large dimensional basis. In particular this allows to compute approximate shortest vectors from lower dimensional exact SVP oracles.

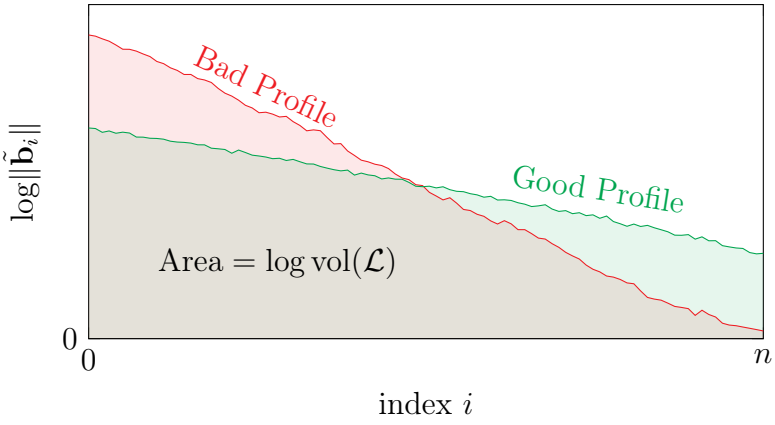

Figure 6.2: A good versus a bad basis log-profile.

6.1.1 Basis profile

It turns out that algorithmically, the profile of a basis, i.e., its GSO norms $(\|\tilde{\mathbf{b}}_i\|)_i$ are much more interesting than the norms of the basis vectors themself. They are directly related, i.e., by Lemma 38 a size-reduced basis consists of short vectors if and only if the GSO norms are short. Still, most cryptanalytic algorithms actually rely on the profile, and not the basis norms.

One such example is Babai's nearest plane algorithm. Recall from Section 2.4.3 that given a basis $\mathbf{B}$ of $\mathcal{L}$, Babai's nearest plane algorithm decodes any target $\mathbf{t} \in \mathrm{span}(\mathcal{L})$ to a close vector $\mathbf{c} \in \mathcal{L}$ such that $\mathbf{t} - \mathbf{c} \in \mathcal{P}(\tilde{\mathbf{B}})$. As a result Babai's algorithm perfectly decodes targets up to distance $\frac{1}{2}\min_i\|\tilde{\mathbf{b}}_i\|$ from the lattice, and has worst-case and average-case squared decoding distance proportional to $\sum_i\|\tilde{\mathbf{b}}_i\|^2$.

Due to the invariant $\prod_i\|\tilde{\mathbf{b}}_i\| = \mathrm{vol}(\mathcal{L})$, both $\min_i\|\tilde{\mathbf{b}}_i\|$ and $\sum_i\|\tilde{\mathbf{b}}_i\|^2$ are optimal (maximized and minimized respectively) when $\|\tilde{\mathbf{b}}_0\| = \ldots = \|\tilde{\mathbf{b}}_{n-1}\| = \mathrm{vol}(\mathcal{L})^{1/n}$, i.e. when the GSO profile is perfectly balanced. Such a basis with a perfectly balanced profile might not exist, e.g. by the Gaussian Heuristic we already have $\|\tilde{\mathbf{b}}_0\| \geq \lambda_1(\mathcal{L}) \approx \sqrt{n/(2\pi e)}\,\mathrm{vol}(\mathcal{L})^{1/n} \gg \mathrm{vol}(\mathcal{L})^{1/n}$. A typical reduced basis profile starts high and decreases quickly (see Figure 6.2). We could (informally) speak of the *slope* of a profile.

For a not so orthogonal bad basis the slope is steep, and the first GSO norm $\|\tilde{\mathbf{b}}_0\|$ is very large, leading to a poor decoding error propor-

tional to $\sum_i \|\tilde{\mathbf{b}}_i\|^2 \geq \|\tilde{\mathbf{b}}_0\|^2$. Similarly, the last GSO norm $\|\tilde{\mathbf{b}}_{n-1}\|$ is very small, leading to a poor bounded distance decoding performance. In contrast, a close to orthogonal good basis has a gradual declining and much more balanced slope, and therefore achieves much better decoding performances.

In the following Sections we consider three ways of obtaining a good basis. The exponential time HKZ algorithm, which achieves a very good slope by minimizing $\|\tilde{\mathbf{b}}_0\|$, then $\|\tilde{\mathbf{b}}_1\|$, and so on. The polynomial-time LLL algorithm, which achieves a mildly good slope, by improving the slope locally. And lastly the BKZ algorithm that combines these ideas and offers a tunable time-quality trade-off between the two.

6.2 HKZ reduction

One approach to balance the typically decreasing basis profile ($\|\tilde{\mathbf{b}}_0\|$, $\ldots, \|\tilde{\mathbf{b}}_{n-1}\|$), is by minimizing the first norm $\|\tilde{\mathbf{b}}_0\|$, then the second $\|\tilde{\mathbf{b}}_1\|$, and so on. The first GSO vector $\tilde{\mathbf{b}}_0$ is just the first basis vector $\mathbf{b}_0$, so by definition it is minimized when $\|\mathbf{b}_0\| = \lambda_1(\mathcal{L})$. For a fixed $\mathbf{b}_0$, the second GSO vector $\tilde{\mathbf{b}}_1$ is exactly the first basis vector in $\mathcal{L}_{[1:n)}$, and is thus minimized when $\|\tilde{\mathbf{b}}_1\| = \lambda_1(\mathcal{L}_{[1:n)})$. We can continue this greedy process until the end. The resulting basis is said to be Hermite-Korkine-Zolotarev reduced.

Definition 88 (Hermite-Korkine-Zolotarev reduced). *A basis* $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$ *is called Hermite-Korkine-Zolotarev (HKZ) reduced if it is size-reduced and*

$$\|\tilde{\mathbf{b}}_\kappa\| = \lambda_1(\mathcal{L}_{[\kappa:n)}) \quad \textit{for all } \kappa = 0, \ldots, n-1.$$

That such a basis exists follows from the fact that any shortest vector of $\mathcal{L}_{[\kappa:n)}$ is primitive in $\mathcal{L}_{[\kappa:n)}$.

An alternative recursive definition would be as follows: a size-reduced basis $[\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$ is HKZ reduced if $\|\mathbf{b}_0\| = \lambda_1(\mathcal{L})$, and $\mathbf{B}_{[1:n)}$ is HKZ reduced (or equals $\{\mathbf{0}\}$). From this Algorithm 7 to obtain an HKZ reduced basis is straightforward. Compute a shortest vector, project away from it, and repeat on the projected lattice.

HKZ reduction is very strong, and could be interpreted as (close to) optimal for many purposes. In particular, an HKZ reduction oracle

Algorithm 7: The HKZ algorithm.

Data: A rank n lattice $\mathcal{L}$ (represented by any basis).

for $\kappa = 0, \ldots, n-1$ **do**
  $\pi_\kappa := \pi_{(\mathbf{b}_0,\ldots,\mathbf{b}_{\kappa-1})^\perp}$;
  $\mathbf{w} \leftarrow$ a shortest vector in $\pi_\kappa(\mathcal{L})$;
  Lift $\mathbf{w}$ to a lattice vector $\mathbf{b}_\kappa \in \mathcal{L}$ such that $\pi_\kappa(\mathbf{b}_\kappa) = \mathbf{w}$;
  Size-reduce $\mathbf{b}_\kappa$ w.r.t. $[\mathbf{b}_0, \ldots, \mathbf{b}_{\kappa-1}]$;
end
return $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$;

would break almost all lattice-based cryptography. However, this good reduction quality comes at a cost. Since one has to solve an exact SVP instance in dimension n, computing an HKZ reduced basis quickly becomes unfeasible.

6.3 The LLL algorithm

The LLL algorithm by Lenstra, Lenstra and Lovász [LLL82] is a basis reduction algorithm that runs in polynomial time and which reaches approximation factors exponential in the dimension. It was the first polynomial time basis reduction algorithm, and it has found countless applications. Furthermore, it is a fundamental building block in more complex basis reduction algorithms.

The general idea of the LLL algorithm is to improve the basis slope, by making sure that consecutive GSO norms $\|\tilde{\mathbf{b}}_\kappa\|, \|\tilde{\mathbf{b}}_{\kappa+1}\|$ do not decrease too quickly. This is achieved by locally reducing all the rank two projected sublattices $\mathcal{L}_{[\kappa:\kappa+2)} = \mathcal{L}([\pi_\kappa(\mathbf{b}_\kappa), \pi_\kappa(\mathbf{b}_{\kappa+1})])$ for $\kappa = 0, \ldots, n-2$.

6.3.1 Lagrange reduction

So let us first focus on the rank two case $\mathcal{L} = \mathcal{L}([\mathbf{b}_0, \mathbf{b}_1])$. We want two basis vectors to be as short and as orthogonal as possible. Following the HKZ reduction algorithm it is natural to let the first basis vector be a shortest vector of the lattice. In this way we obtain a good GSO profile, and by size-reduction a close to orthogonal basis. The resulting

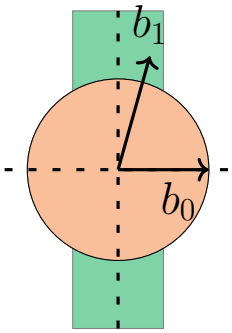

Figure 6.3: Illustration of the Wristwatch Lemma 89. There always exists a basis $[\mathbf{b}_0, \mathbf{b}_1]$ with $\mathbf{b}_0$ a shortest vector, and $\mathbf{b}_1$ in the green strap of width $\frac{1}{2}\|\mathbf{b}_0\|$ (in the direction orthogonal to $\mathbf{b}_0$).

basis is called Lagrange reduced, and visually looks like a 'Wristwatch' basis (see Figure 6.3).

Lemma 89 (Wristwatch Lemma). *Let $\mathcal{L} \subset \mathbb{R}^d$ be a lattice of rank 2, then there exists a $\mathbf{B} = [\mathbf{b}_0, \mathbf{b}_1]$ of $\mathcal{L}$ that satisfies*

1. $\|\mathbf{b}_0\| \leq \|\mathbf{b}_1\|$, *and*
2. $|\langle \mathbf{b}_0, \mathbf{b}_1 \rangle| \leq \frac{1}{2}\|\mathbf{b}_0\|^2$ *(size-reduction).*

In particular this implies that $\|\mathbf{b}_0\| = \lambda_1(\mathcal{L})$ and $\|\tilde{\mathbf{b}}_0\| \leq \sqrt{4/3} \cdot \|\tilde{\mathbf{b}}_1\|$. We call such a basis Lagrange reduced.

There is an efficient algorithm to obtain such a basis.

Lemma 90. *On input a basis of a rank 2 lattice $\mathcal{L}$, Algorithm 8 terminates and returns a Lagrange reduced basis $[\mathbf{b}_0, \mathbf{b}_1]$ of $\mathcal{L}$, i.e. a basis satisfying the condition in Lemma 89.*

Proof. Swapping basis vectors, and subtracting integer multiples of other basis vectors, are all unimodular operations. Therefore the returned basis is still a basis of the input lattice. This also implies that $\mathbf{b}_0, \mathbf{b}_1$ are nonzero throughout the algorithm.

The algorithm terminates in a finite number of steps given that the norm of $\|\mathbf{b}_0\|$ strictly decreases in each iteration, and because by a packing argument there are only a finite number of lattice points in

Algorithm 8: Lagrange reduction

Input : A rank 2 basis $[\mathbf{b}_0, \mathbf{b}_1]$.
Input : A Lagrange reduced basis of the input lattice.

repeat
  swap $\mathbf{b}_0 \leftrightarrow \mathbf{b}_1$
  $k \leftarrow \left\lfloor \frac{\langle \mathbf{b}_0, \mathbf{b}_1 \rangle}{\langle \mathbf{b}_0, \mathbf{b}_0 \rangle} \right\rceil$
  $\mathbf{b}_1 \leftarrow \mathbf{b}_1 - k \cdot \mathbf{b}_0$ `// size-reduction`
until $\|\mathbf{b}_0\| \leq \|\mathbf{b}_1\|$
return $[\mathbf{b}_0, \mathbf{b}_1]$

$\mathcal{L}$ of norm at most some bound. A closer inspection shows that the number of iterations is of order at most $O(\log(\|\mathbf{b}_0\|))$.

Now let us consider the two conditions of Lemma 89. The second condition $\|\mathbf{b}_0\| \leq \|\mathbf{b}_1\|$ is true by the termination condition of the repeat loop. The size-reduction operation just before that makes sure that the first condition $|\langle \mathbf{b}_0, \mathbf{b}_1 \rangle| \leq \frac{1}{2}\|\mathbf{b}_0\|^2$ is also true. □

Proof Lemma 89. The existance of such a basis follows directly from Lemma 90. Let $[\mathbf{b}_0, \mathbf{b}_1]$ be a Lagrange reduced basis. Consider any nonzero lattice vector $\mathbf{v} = x\mathbf{b}_0 + y\mathbf{b}_1 \in \mathcal{L}$ with $(0,0) \neq (x,y) \in \mathbb{Z}^2$. We want to show that $\|\mathbf{v}\| \geq \|\mathbf{b}_0\|$. When $y = 0$ this is trivial, and when $x = 0$ it follows from $\|\mathbf{b}_1\| \geq \|\mathbf{b}_0\|$. For the case $x \neq 0$ and $y \neq 0$ we have

$$\begin{aligned}\|\mathbf{v}\|^2 &= x^2 \cdot \|\mathbf{b}_0\|^2 + y^2 \cdot \|\mathbf{b}_1\|^2 + 2xy\langle \mathbf{b}_0, \mathbf{b}_1 \rangle \\ &\geq (x^2 + y^2 - |xy|)\|\mathbf{b}_0\|^2 \geq \min\{x^2, y^2\} \cdot \|\mathbf{b}_0\|^2 \geq \|\mathbf{b}_0\|^2,\end{aligned}$$

and thus $\|\mathbf{b}_0\| = \lambda_1(\mathcal{L})$. For the GSO norms $\|\tilde{\mathbf{b}}_0\|, \|\tilde{\mathbf{b}}_1\|$ we have

$$\|\mathbf{b}_0\|^2 \leq \|\mathbf{b}_1\|^2 = \|\tilde{\mathbf{b}}_1\|^2 + |\langle \mathbf{b}_0, \mathbf{b}_1 \rangle|^2 / \|\mathbf{b}_0\|^2 \leq \|\tilde{\mathbf{b}}_1\|^2 + \frac{1}{4}\|\mathbf{b}_0\|^2,$$

and thus $\|\tilde{\mathbf{b}}_0\| = \|\mathbf{b}_0\| \leq \sqrt{4/3} \cdot \|\tilde{\mathbf{b}}_1\|$. □

6.3.2 LLL

If all the projected sublattice bases $\mathbf{B}_{[i:i+2)}$ are Lagrange reduced, then we obtain $\|\tilde{\mathbf{b}}_{i+1}\| \geq \sqrt{3/4}\|\tilde{\mathbf{b}}_i\|$ for all $i = 0, \ldots, n-2$. In other

words, the profile cannot decrease too quickly. Simply stated the LLL algorithm proceeds as follows: if any local basis $\mathbf{B}_{[i:i+2)}$ is not Lagrange reduced, then reduce it. Clearly if the algorithm terminates all local bases are Lagrange reduced. To show that it terminates in a polynomial number of iterations we need to slightly relax the local Lagrange condition $\|\pi_i(\mathbf{b}_i)\| \leq \|\pi_i(\mathbf{b}_{i+1})\|$.

Definition 91 (LLL reduced). *A lattice basis* $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_{d-1}]$ *is called* δ*-LLL reduced, for* $\delta \in (\frac{1}{4}, 1]$*, if*

1. *The Lovász Condition* $\delta\|\pi_i(\mathbf{b}_i)\|^2 \leq \|\pi_i(\mathbf{b}_{i+1})\|^2$ *holds for all* $0 \leq i < n-1$*, and*

2. *it is size-reduced.*

We call $\delta \in (\frac{1}{4}, 1]$ the reduction parameter. Note that $\delta = 1$ represents exact Lagrange reduction, but for this value we cannot show that the algorithm terminates in polynomial time. A δ-LLL reduced bases does not decrease too steeply, i.e. combining the two conditions we obtain

$$\|\tilde{\mathbf{b}}_{\kappa+1}\| \geq \sqrt{\delta - \frac{1}{4}}\|\tilde{\mathbf{b}}_{\kappa}\|,$$

for all $\kappa = 0, \ldots, n-2$. This directly has implications for the length of the basis vectors.

Corollary 92 (LLL results). *Let* $\delta \in (\frac{1}{4}, 1]$*. For a* δ*-LLL reduced basis* $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$ *of the lattice* $\mathcal{L}$*, we have*

$$\|\mathbf{b}_0\| \leq (\delta - \tfrac{1}{4})^{-(n-1)/2} \cdot \lambda_1(\mathcal{L})$$
$$\|\mathbf{b}_0\| \leq (\delta - \tfrac{1}{4})^{-(n-1)/4} \cdot \mathrm{vol}(\mathcal{L})^{1/n}$$

Proof. Let $\gamma := \sqrt{1/(\delta - \frac{1}{4})}$. By combining the LLL conditions we have

$$\|\mathbf{b}_0\| = \|\tilde{\mathbf{b}}_0\| \leq \gamma \cdot \|\tilde{\mathbf{b}}_1\| \leq \ldots \gamma^{n-1} \cdot \|\tilde{\mathbf{b}}_{n-1}\|.$$

In particular we have $\|\mathbf{b}_0\| \leq \gamma^{n-1} \cdot \min_i \|\tilde{\mathbf{b}}_i\|$, and the first statement follows from the fact that $\min_i \|\tilde{\mathbf{b}}_i\| \leq \lambda_1(\mathcal{L})$. For the second statement

we have

$$\det(\mathcal{L})^{1/n} = \left(\prod_{i=0}^{n-1} \|\tilde{\mathbf{b}}_i\|\right)^{1/n} \geq \left(\prod_{i=0}^{n-1} \|\tilde{\mathbf{b}}_0\| \cdot \gamma^i\right)^{1/n} = \|\mathbf{b}_0\| \cdot \gamma^{(n-1)/2}.$$

□

For $\delta = 1$ we have $\|\mathbf{b}_0\| \leq (4/3)^{(n-1)/2} \cdot \mathrm{vol}(\mathcal{L})^{1/n}$. Recall that Hermite's bound says precisely that such a vector exists, i.e., $\lambda_1(\mathcal{L}) \leq (4/3)^{(n-1)/2} \cdot \mathrm{vol}(\mathcal{L})^{1/n}$. As such, the LLL algorithm can be seen as an algorithmic instantiation of Hermite's bound: it does not just show existence, it actually computes a vector of (almost) that length.

6.3.3 LLL algorithm

We will now discuss why the LLL algorithm terminates in a polynomial number of iterations. To show that the LLL algorithm runs in polynomial-time one would additionally have to show that the operations, like computing the GSO (using rationals), run in polynomial-time. We will not cover this, but this is precisely where the global size-reduction of the basis comes into play.

Algorithm 9: LLL reduction

Input : A basis $\mathbf{B} = [\mathbf{b}_0, , \ldots, \mathbf{b}_{n-1}]$ of a lattice $\mathcal{L}$ and a reduction parameter $\delta \in (\frac{1}{4}, 1]$.

Output: A δ-LLL reduced basis of the input lattice.

```
Size-reduce B
while ∃index i that does not satisfy Lovász' condition do
    Let U ∈ GL_2(ℤ) be s.t. B_[i:i+2) · U is Lagrange reduced
     // Using Algorithm 8
    [b_i, b_{i+1}] ← [b_i, b_{i+1}] · U                    // Update B
    Size-reduce B
end
return B
```

To argue that the number of iterations is polynomially bounded in the input size we use a *potential* argument.

Definition 93 (Potential). *For a basis* $\mathbf{B}$*, we define the potential* $\mathcal{D}_{\mathbf{B}}$ *as*

$$\mathcal{D}_{\mathbf{B}} = \prod_{i=1}^{n} \mathrm{vol}(\mathcal{L}_{0:i}) = \prod_{i=0}^{n-1} \|\tilde{\mathbf{b}}_i\|^{n-i}.$$

Lemma 94. *For any integer basis* $\mathbf{B} \in \mathbb{Z}^{m \times n}$ *and reduction parameter* $\delta \in (\frac{1}{4}, 1)$*, Algorithm 9 is correct and terminates after at most*

$$O(n^2 \log(\max_i \|\mathbf{b}_i\|))$$

iterations.

Proof. The correctness is clear from the loop condition.

To show that the number of iterations is polynomially bounded we first argue that the potential decreases significantly during each loop. First note that size-reducing a basis does not change the GSO norms, and thus the size-reduction operations do not change the potential. Now let us consider a loop, where $0 \leq i \leq n-2$ is the index at which Lovász' condition is not satisfied. At the start of the loop we thus have

$$\delta \|\pi_i(\mathbf{b}_i)\|^2 > \|\pi_i(\mathbf{b}_{i+1})\|^2.$$

Let $[\mathbf{c}_i, \mathbf{c}_{i+1}] = [\mathbf{b}_i, \mathbf{b}_{i+1}] \cdot \mathbf{U}$ be the new basis vectors, $\tilde{\mathbf{c}}_i, \tilde{\mathbf{c}}_{i+1}$ the new GSO vectors, and let $\mathcal{D}_{\mathbf{B}}$ and $\mathcal{D}_{\mathbf{B}'}$ be the old and new potential respectively. The lattice $\mathcal{L}_{[i:i+2)}$ stays the same (and thus its span), so all GSO norms $\|\tilde{\mathbf{b}}_j\|$ for $j \notin \{i, i+1\}$ remain unchanged in this loop. Furthermore we have the invariant

$$\|\tilde{\mathbf{c}}_i\| \cdot \|\tilde{\mathbf{c}}_{i+1}\| = \mathrm{vol}(\mathcal{L}_{[i:i+2)}) = \|\tilde{\mathbf{b}}_i\| \cdot \|\tilde{\mathbf{b}}_{i+1}\|.$$

By Lagrange reduction, the vector $\tilde{\mathbf{c}}_i$ is a shortest vector in $\mathcal{L}_{[i:i+2)}$, and in particular $\|\tilde{\mathbf{c}}_i\| \leq \|\pi_i(\mathbf{b}_{i+1})\|$. To summarize we have

$$\|\tilde{\mathbf{c}}_i\|^2 \leq \|\pi_i(\mathbf{b}_{i+1})\|^2 < \delta \cdot \|\tilde{\mathbf{b}}_i\|^2,$$

and together with the invariant this gives

$$\|\tilde{\mathbf{c}}_i\|^{n-i} \cdot \|\tilde{\mathbf{c}}_{i+1}\|^{n-(i+1)} < \sqrt{\delta} \cdot \|\tilde{\mathbf{b}}_i\|^{n-i} \cdot \|\tilde{\mathbf{b}}_{i+1}\|^{n-(i+1)},$$

and thus $\mathcal{D}_{\mathbf{B}'} < \sqrt{\delta} \cdot \mathcal{D}_{\mathbf{B}}$. So the potential decreases by a constant factor $\frac{1}{2} < \sqrt{\delta} < 1$ each iteration.

To conclude we now have to upper and lower bound the potential. For the lower bound note that for any integer basis $\mathbf{B}$ we have $\mathcal{D}_{\mathbf{B}} \geq 1$. For the upper bound we have

$$\mathcal{D}_{\mathbf{B}} \leq \prod_{i=0}^{n-1} \|\mathbf{b}_i\|^{n-i} \leq B^{\frac{1}{2}n(n+1)},$$

where $B := \max_i \|\mathbf{b}_i\|$. By the constant factor decrease each iteration, and a lower and upper bound of 1 and $B^{\frac{1}{2}n(n+1)}$, there can be at most $O(n^2 \log B)$ iterations. □

We obtained the lower bound $\mathcal{D}_{\mathbf{B}} \geq 1$ by restricting to integer bases. This is not necessary, alternatively by Hermite's inequality or Minkowski's Theorem we could lower bound each volume $\mathrm{vol}(\mathcal{L}_{0:i})$ in terms of $\lambda_1(\mathcal{L}_{0:i}) \geq \lambda_1(\mathcal{L})$, which only depends on the lattice.

The LLL algorithm does not even need a basis, it can also be modified to work directly on the Gram matrix. The upper triangular Cholesky matrix $\mathbf{C}$ from the decomposition $\mathbf{G} = \mathbf{C}^\top\mathbf{C}$ of a Gram matrix plays the role of (basis and) GSO. The Gram matrix is modified by the unimodular transformations as $\mathbf{G}' = \mathbf{U}^\top\mathbf{G}\mathbf{U}$.

A small modification to the LLL algorithm by Pohst [Poh87] allows to remove the requirement that the input vectors are linearly independent. After this modification the input can consists of any number of vectors, and the output will consists of an LLL reduced basis of the lattice generated by the input vectors. In the next section we will use this property to 'insert' short vectors in a basis. Given a basis $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$ and a short vector $\mathbf{v} \in \mathcal{L}(\mathbf{B})$, we can run the modified LLL reduction algorithm on $[\mathbf{v}, \mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$. Because the first GSO norm can never increase during the LLL reduction algorithm we end up with an LLL reduced basis $\mathbf{B}' = [\mathbf{b}'_0, \ldots, \mathbf{b}'_{n-1}]$ with $\|\mathbf{b}'_0\| \leq \min\{\|\mathbf{b}_0\|, \|\mathbf{v}\|\}$.

A nice property of the LLL algorithm is that it is self-dual: to be more precise, if a basis $\mathbf{B}$ is δ-LLL reduced, then the reversed dual basis $\mathbf{D}$ of $\mathbf{B}$ is after size-reducing also δ-LLL reduced. This means that e.g. Corollary 92 can also be implied to the reverse dual basis $[\mathbf{d}_{n-1}, \ldots, \mathbf{d}_0]$, and thus give a lower bound on $\|\tilde{\mathbf{b}}_{n-1}\| = 1/\|\mathbf{d}_{n-1}\|$.

6.4 BKZ reduction

We have seen the two extreme cases of basis reduction, the HKZ algorithm delivers close to optimal reduction at the cost of solving SVP in the full lattice, and the LLL algorithm that gives exponential approximation factors but runs in polynomial time. We will now discuss a generalisation, that delivers a trade-off between these two extremes.

So how to generalize the LLL and HKZ algorithms? The common theme is that they both rely on an SVP oracle for projected sublattices $\mathcal{L}_{[\kappa:\kappa+\beta)}$, which we call a *block*.

The difference being that HKZ works on blocks $\mathcal{L}_{[\kappa:n)}$ of rank at most n, while the LLL works on blocks $\mathcal{L}_{[\kappa:\kappa+2)}$ of rank 2 (Lagrange reduction). From this perspective the natural generalisation is to work on blocks $\mathcal{L}_{[\kappa:\kappa+\beta)}$ of rank at most β. This is precisely what Block-Korkine-Zolotarev [Sch87] reduction does, and we call the parameter β the *blocksize*.

Definition 95 (BKZ). *A basis $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$ is called BKZ-β reduced if it is size-reduced and*

$$\|\tilde{\mathbf{b}}_\kappa\| = \lambda_1(\mathcal{L}_{[\kappa:\min(\kappa+\beta,n))}) \text{ for all } \kappa = 0, \ldots, n-1.$$

For $\beta = 2$ we recover the definition of 1-LLL, and for $\beta = n$ we recover the definition of HKZ. The condition on $\|\tilde{\mathbf{b}}_\kappa\|$ is often relaxed by a factor $1+\epsilon$ for some $\epsilon > 0$ close to 0, both to allow for numerical imprecisions and potential arguments like LLL. This is similar to the relaxation of the Lovász Condition in LLL. In what follows we will ignore this factor.

Similar to LLL a BKZ reduced basis contains a short basis vector.

Lemma 96 (BKZ approximation [Sch94]). *For a BKZ-δ reduced basis $\mathbf{B} = [\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}]$ of the lattice $\mathcal{L}$, with $2 \leq \beta \leq n$, we have*

$$\|\mathbf{b}_0\| \leq \gamma_\beta^{\frac{n-1}{\beta-1}} \cdot \mathrm{vol}(\mathcal{L})^{1/n},$$

where γ_β is Hermite's constant of rank β.

In addition a Hermite factor bound $\|\mathbf{b}_0\| \leq \sqrt{\gamma_\beta}^{\frac{n-1}{\beta-1}+1} \cdot \mathrm{vol}(\mathcal{L})^{1/n}$ is claimed without proof in [GN08b]. Because $\gamma_\beta \leq O(\beta)$ the bound decreases as the blocksize β increases; a larger blocksize gives a better reduced basis.

6.4.1 BKZ algorithm

The BKZ algorithm (see Algorithm 10) computes a BKZ reduced basis from any other basis. The algorithm greedily attempts to satisfy the BKZ condition at each position by computing a shortest vector in each block $\mathcal{L}_{[\kappa:\min(\kappa+\beta,n))}$, and replacing the basis vector $\mathbf{b}_\kappa$ accordingly. This makes the basis BKZ-β reduced at position κ, but might invalidate the condition at other positions. Applying this once to all positions $\kappa = 0, \ldots, n-2$ is called a *tour*. The BKZ algorithm repeats such tours until the basis remains unchanged and is thus BKZ reduced.

Algorithm 10: The BKZ algorithm.

Input : A lattice basis $\mathbf{B}$, blocksize β.
Output: A BKZ-β reduced basis of the input lattice.

LLL reduce $\mathbf{B}$
while $\mathbf{B}$ *is not BKZ-*β *reduced* **do**
  for $\kappa = 0, \ldots, n-2$ **do** `// A single BKZ-β tour`
    $b \leftarrow \min\{\beta, n-\kappa\}$
    $\mathbf{w} \leftarrow$ a shortest vector in $\mathcal{L}_{[\kappa:\kappa+b)}$
    Lift $\mathbf{w}$ to a full vector $\mathbf{v} \in \mathcal{L}_{[0:\kappa+b)}$ s.t. $\pi_\kappa(\mathbf{v}) = \mathbf{w}$
    Insert $\mathbf{v}$ in $\mathbf{B}$ at position κ using LLL
    LLL reduce $\mathbf{B}_{[0:\kappa+b)}$
  end
end
return B

The BKZ algorithm as stated is not known to terminate in a polynomial number of tours. Terminating variants exist that succeed after about $O(n^2 \log(n)/\beta^2)$ tours [HPS11b; LN20], but the resulting bases are not precisely BKZ-β reduced, and the resulting worst-case bounds on $\|\mathbf{b}_0\|$ are a polynomial factor worse than Lemma 96.

The troubles of proving a polynomial bound on the number of tours, with a potential argument like LLL, seems to be related to the fact that the BKZ algorithm is not self-dual like LLL. There do exists variants of BKZ, like SDBKZ [MW16], that are self-dual, and that have easier provable runtime and quality bounds.

In practice however, not much improvement to the basis profile is attained after say a few dozen tours. The cost of BKZ is thus mainly dominated by the exponential (in β) cost of finding a shortest vector

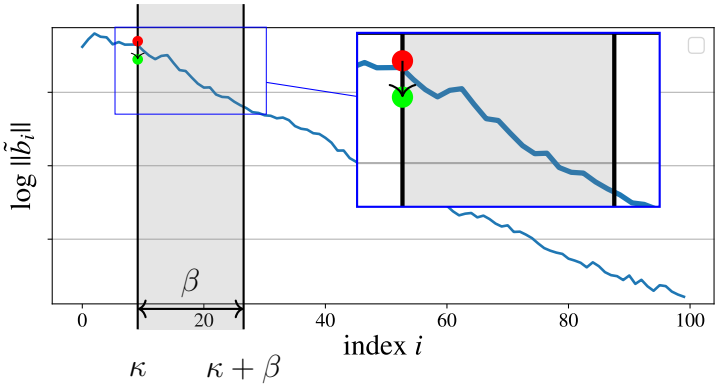

Figure 6.4: The BKZ algorithm: inserting a shortest vector in the projected sublattice $\mathcal{L}_{[\kappa:\kappa+\beta)}$.

in a β-dimensional lattice. Throughout this thesis we will therefore ignore the lattice rank and the number of iterations and cost the BKZ algorithm in terms of the blocksize β.

6.4.2 Progressive BKZ

The cost of a BKZ-β tour quickly grows when increasing β. We thus want to limit the number of tours we have to run at the maximal blocksize β. The better the basis is already reduced, the less tours are needed to reach a (close to) BKZ-β reduced basis. If we first reduce the basis with a lower blocksize $\beta' < \beta$, then afterwards only a few expensive tours are needed. Beyond the number of expensive tours needed this also has benefits for the underlying SVP subroutine. When using (extreme) enumeration a better basis lowers the cost significantly, and when using sieving a better basis improves both the number of dimensions for free and the number of sieving iterations needed during progressive sieving. The idea of *progressive BKZ* (Algorithm 11) is to apply this idea recursively: run only a few tours (say 1 or 2) for increasing $\beta' = 2, 3, \ldots, \beta$.

Algorithm 11: Progressive BKZ.

Input : A lattice basis $\mathbf{B}$, blocksize β, tours $T > 0$.

for $\beta' = 2, 3, \ldots, \beta$ **do**
  Run T BKZ-β' tours.
end
return B

6.4.3 Slide reduction

An alternative generalisation of LLL is the *slide reduction* algorithm. It uses SVP oracles both on primal blocks $\mathcal{L}_{[\kappa:\kappa+\beta)}$, and on dual blocks $\mathcal{L}^*_{[\kappa:\kappa:\beta)}$. This gives more control over the profile and makes slide reduction easier to (provably) analyse than BKZ. For $n \geq 2\beta$ this also leads to slightly better asymptotic bounds on the norm $\|\mathbf{b}_0\|$ of the first basis vector.

Unfortunately, in practice the performance of BKZ seems to be better than slide reduction, and as a result most (concrete) cryptanalysis has been based on the BKZ algorithm. Also, the BKZ algorithm reduces the full bases, which is important for solving certain variants of SVP that are common in cryptography. In contrast, slide reduction focusses mostly on decreasing the norm $\|\mathbf{b}_0\|$ of the first basis vector, e.g. solving approx-SVP. Indeed, recently it was hinted that slide reduction can be competitive with BKZ for approx-SVP [MW16; Wal21]. While it would be interesting to see how the slide reduction algorithm behaves on the lattice problems in this thesis, we keep our focus on the BKZ algorithm which is much better understood from a concrete perspective.

6.5 Behaviour of reduction algorithms

For cryptanalysis it is important to understand the practical behaviour of lattice reduction algorithms. In particular in cryptographic dimensions, in which it is infeasible to obtain experimental evidence. In this section we explain the heuristic models behind the behaviour of HKZ, LLL and BKZ. These models are often build on the Gaussian Heuristic and based on extensive experimental evidence in feasible dimensions.

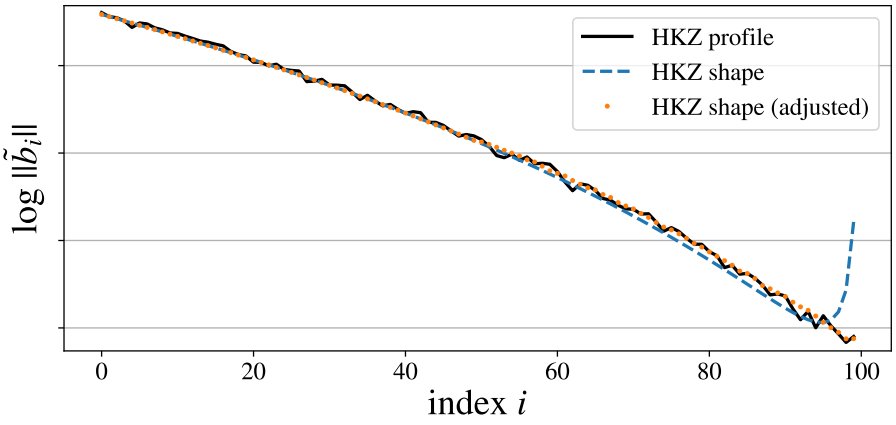

Figure 6.5: A typical HKZ reduced basis of a rank 100 lattice versus the HKZ shape of Definition 97, with and without adjustments for the tail part.

6.5.1 HKZ shape

In Figure 6.5 we see that the typical log-profile of a HKZ reduced random lattice basis is somewhat concave. This shape is correctly predicted by applying the Gaussian Heuristic to the HKZ definition.

Definition 97. *We define the HKZ shape* $(\ell_0, \ldots, \ell_{n-1})$ *of rank* n *by the following sequence*

$$\ell_0 := \mathrm{gh}(n), \qquad \ell_i = \mathrm{gh}(n-i) \cdot \left(\prod_{j<i} \ell_j \right)^{-1/(n-i)}.$$

The above profile shape is for a lattice with volume 1. In general for a HKZ reduced basis $\mathbf{B}$ of rank n the Gaussian Heuristic says that $\log \|\tilde{\mathbf{b}}_i\| \approx \log \ell_i + \frac{1}{n} \log(\det \mathcal{L})$. This estimate is accurate for $i \ll n$, say when $i \leq n - 50$, as can be seen in Figure 6.5.

The tail part of the estimate is inaccurate, because the Gaussian Heuristic gives false predictions in such low dimensions. One way to solve this is to experimentally observe a HKZ profile for a lattice of rank 50, and use that for the last part of the estimate. This adjusted estimate, as shown in Figure 6.5, closely predicts the actual HKZ profile.

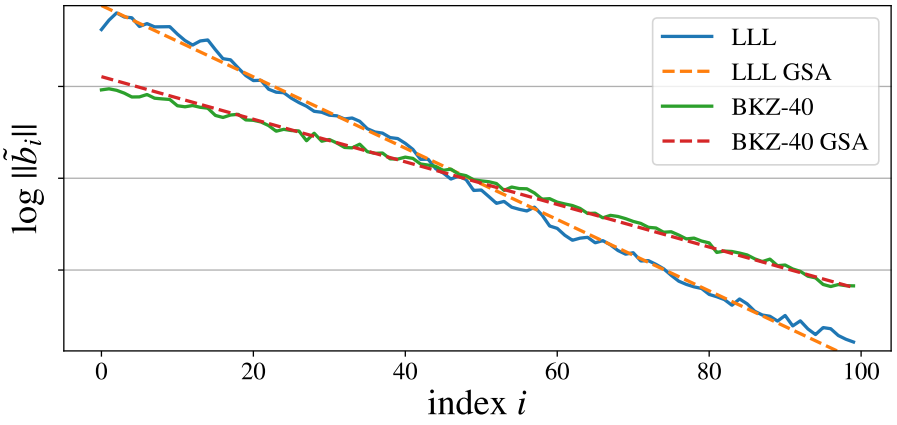

Figure 6.6: Illustration of the Geometric Series Assumption for 100 dimensional LLL and BKZ-40 reduced bases.

6.5.2 Geometric Series Assumption

In Figure 6.6 we see that the typical log-profile of an LLL or BKZ-β reduced basis for $\beta \ll n$ is close to a straight line, i.e. $\|\tilde{\mathbf{b}}_{i+1}\|/\|\tilde{\mathbf{b}}_i\| = \alpha$ for some constant $\alpha > 1$. We can predict the constant by the Gaussian Heuristic, which gives us the Geometric Series Assumption (GSA).

Heuristic 98 (Geometric Series Assumption (GSA)). *Let* $\mathbf{B}$ *be a basis of rank* n *that is BKZ-*β *reduced, then its profile satisfies*

$$\log(\|\tilde{\mathbf{b}}_i\|) = \frac{n-1-2i}{2} \cdot \log(\alpha_\beta) + \frac{\log(\det(\mathbf{B}))}{n},$$

where $\alpha_\beta = \mathrm{gh}(\beta)^{2/(\beta-1)}$.

Justification. Let us zoom in on a BKZ block $\mathcal{L}' = \mathcal{L}_{[\kappa:\kappa+\beta)}$. By the Gaussian Heuristic we have $\lambda = \lambda_1(\mathcal{L}') = \mathrm{gh}(\beta) \cdot \mathrm{vol}(\mathcal{L}')^{1/\beta}$. Under the assumption that the GSO norms decrease exponentially by some factor $\alpha_\beta > 1$ we have $\|\tilde{\mathbf{b}}_{\kappa+i}\| = \lambda/\alpha^i$, and thus $\mathrm{vol}(\mathcal{L}')^{1/\beta} = \lambda/\alpha_\beta^{(\beta-1)/2}$. To conclude the constant $\alpha_\beta > 1$ must satisfy

$$\lambda = \mathrm{gh}(\beta) \cdot \mathrm{vol}(\mathcal{L}') = \mathrm{gh}(\beta) \cdot \lambda/\alpha_\beta^{(\beta-1)/2},$$

from which it follows that $\alpha_\beta = \mathrm{gh}(\beta)^{2/(\beta-1)}$. △

The GSA is reasonably precise for say $\beta \geq 50$ and a not too large blocksize $\beta \ll n$ compared to the lattice rank. For small blocksizes β one can determine the value of α_β experimentally to still get accurate predictions.

However, one should be careful when using the GSA to estimate the start (head) and the end (tail) of the profile. Firstly by definition the last BKZ block $\mathcal{L}_{[n-\beta:n)}$ is HKZ reduced and thus the tail of the profile is slightly concave. Secondly, the Gaussian Heuristic is only an average-case statement, and during BKZ reduction vectors slightly shorter than the prediction can be found. By their shortness the BKZ algorithm is biased towards keeping those vectors and pushing them to the front of the basis. As a result the head of the profile can be a bit lower than predicted by the GSA.

6.5.3 Simulators

While the Geometric Series Assumption gives a good first order estimate of the basis profile after BKZ-reduction, it is known to be inaccurate in small dimensions or when the dimension is only a small multiple of the blocksize due to the head and tail behaviour. Additionally it does not account for the slower convergence when running progressive BKZ with only a few tours. To resolve these problems [CN11] introduced a BKZ simulator based on the Gaussian Heuristic.

This simulator keeps track of the profile $\ell = (\|\tilde{\mathbf{b}}_0\|, \ldots, \|\tilde{\mathbf{b}}_{n-1}\|)$, and runs BKZ tours where instead of computing a shortest vector it just updates

$$\ell_\kappa = \min\left\{\ell_\kappa, \mathrm{gh}(b) \cdot \left(\prod_{i=\kappa}^{\kappa+b-1} \ell_i\right)^{1/b}\right\},$$

for $b = \min\{\beta, n-\kappa\}$ (or experimental values of $\mathrm{gh}(b)$ for $b < 50$), and adjusts the remaining norms $\ell_\kappa, \ldots, \ell_{\kappa+b-1}$ accordingly. Such a simulator predicts correctly both the center (body) and tail part of the profile. This simulator was later refined by a probabilistic variant of the Gaussian Heuristic that can return slightly short vectors with a small probability [YD17; BSW18]. Due to this addition the head behaviour is also captured. In short, these simulators allow for accurate

and efficient predictions of the profile shape for random lattices, even for progressive BKZ with a limited number of tours.