# The Convergence of Slide-type Reductions

Michael Walter***

IST Austria
michael.walter@ist.ac.at

**Abstract.** In this work, we apply the dynamical systems analysis of Hanrot *et al.* (CRYPTO'11) to a class of lattice block reduction algorithms that includes (natural variants of) slide reduction and block-Rankin reduction. This implies sharper bounds on the polynomial running times (in the query model) for these algorithms and opens the door to faster practical variants of slide reduction. We give heuristic arguments showing that such variants can indeed speed up slide reduction significantly in practice. This is confirmed by experimental evidence, which also shows that our variants are competitive with state-of-the-art reduction algorithms.

## 1 Introduction

Lattice block reduction is a key tool in cryptanalysis, so understanding its potential and its limitations is essential for the security of many cryptosystems. The basic idea of lattice block reduction is to use an oracle that solves the shortest vector problem (SVP) on lattices with low dimension to find short vectors in lattices with larger dimension. Most work in lattice block reduction has focused on BKZ [Sch87,SE94] – the first generalization of the celebrated LLL [LLL82] algorithm, see e.g. [GN08b,HPS11,CN11,Wal15,ADH+19,AWHT16,ABF+20,LN20] to list just a few. Other reduction algorithms are known, like slide reduction [GN08a,ALNS20] and SDBKZ [MW16], which allow proving better bounds on the output quality, but in practice BKZ is still the go-to choice for finding short vectors. Block reduction algorithms are usually judged by the shortness of the vectors they are able to find within a given amount of time. The length of the vector found can be quantified in two different ways: by its ratio with either 1) the shortest non-zero vector of the lattice (the approximation factor) or 2) the ($n$-th root of the) volume/determinant of the lattice (the Hermite factor).

*Slide Reduction.* The focus of this work is slide reduction and, to some degree, its generalization to block-Rankin reduction [LN14]. When it was introduced,

* Supported by the European Research Council, ERC consolidator grant (682815 - TOCNeT).

slide reduction provided the best-known bounds on the approximation and Hermite factor and was easily proved to terminate in a polynomial number of calls to the SVP oracle. Other algorithms achieving the same Hermite factor and terminating in a (smaller) polynomial number of SVP calls are known at this point [MW16,Neu17], but to date, slide reduction still achieves the best bound on the approximation factor. The basic idea of slide reduction is simple: given a basis $\mathbf{B}$ for an $n$-dimensional lattice, a block size $d$ that divides[1] $n$ and an oracle that solves SVP in dimension $d$, apply the SVP oracle to the $n/d$ disjoint (projected) blocks $\left(\mathbf{B}_{[id+1,(i+1)d]}\right)_i$ of the basis. Then apply the oracle to the *dual* of the blocks shifted by 1, i.e. to $\left(\widehat{\mathbf{B}}_{[id+2,(i+1)d+1]}\right)_i$. This results in "primal" and "dual" blocks that overlap by one index (and $d-1$ indices). This process is repeated until no more progress is made. The generalization to block-Rankin reduction works similarly, but it solves a more general problem and uses a more general tool. It approximates the densest sublattice problem (DSP) [DM13], which is itself a generalization of SVP, by relying on an oracle that solves the $k$-DSP in dimension $d$. (SVP corresponds to 1-DSP.) In this variant, the dual blocks are shifted by $k$ resulting in overlaps of size $k$. The analysis of this algorithm is a straightforward adaptation of the one for slide reduction. Unfortunately, initial experimental evaluations of slide reduction [GN08a,GN08b] found it to be not competitive in practice with BKZ and so far there has been no research into practical variants of slide reduction and block-Rankin reduction to the best of our knowledge. This is despite the fact that it offers some trivial parallelization, since the disjoint blocks can be processed independently. This is not true for other reduction algorithms and could give slide reduction a considerable advantage in practice, especially because modern SVP solvers are hard to distribute.

*Dynamical Systems Analyses.* Inspired by the analysis of LLL, [GN08a,LN14] used an analysis based on a potential function to bound the number of oracle calls in slide reduction and block-Rankin reduction. Such an analysis does not work well for BKZ and for a long time it was open if the number of oracle calls in BKZ may be polynomially bounded. This changed when [HPS11] proposed an analysis based on dynamical systems to study BKZ and showed that one can put a polynomial bound on the number of oracle calls while preserving its output quality. Interestingly, this bound was much stronger than the one proven for slide reduction (and block-Rankin reduction) using the potential function. It was conjectured in [HPS11] that applying their approach to slide reduction may give much better bounds than the ones proven in [GN08a,LN14].

A similar analysis was later used to study another reduction algorithm, SDBKZ [MW16], where the analysis was simpler and cleaner. Unfortunately, [MW16] left a gap, where for certain parameterizations of the algorithm the bound on the number of oracle calls was not polynomial. The gap was closed

[1] The restriction that $d \mid n$ is lifted in [ALNS20] by combining it with the algorithm of [MW16].

later by [Neu17], using a seemingly different analysis: "simple (and sharper) induction arguments on a bound on the bit sizes". On closer inspection, it turns out that the analysis of [Neu17] can also be seen as an instance of the typical dynamical systems analysis, but with a small tweak. We make this tweak explicit in Section 5, which allows us to apply a similar tweak in our analysis of Slide-type reductions (see below).

### 1.1 Results

In this work, we consider a class of reduction algorithms that capture natural variants of slide reduction and block-Rankin reduction. We apply the dynamical systems analysis to the algorithms in this class and show that they converge quickly. This implies sharper polynomial-time running time bounds in the query model for slide reduction (when used to find short vectors in terms of the Hermite factor) and block-Rankin reduction.

**Theorem 1 (Informal).** *Let $\mathbf{B} \in \mathbb{R}^{m \times n}$ be an LLL-reduced lattice basis with $\det(\mathcal{L}(\mathbf{B})) = 1$ and $\epsilon > 0$ an arbitrary constant. After $O\left(\frac{n^3}{dk(d-k)} \ln\left(\frac{n}{\epsilon}\right)\right)$ calls to the $(k,d)$-DSP oracle, the output basis of block-Rankin reduction satisfies*

$$\det\left(\mathcal{L}\left(\mathbf{B}_{[1,k]}\right)\right)^{1/k} \lesssim (1+\epsilon)\gamma_{k,d}^{\frac{n-k}{2k(d-k)}}.$$

The best previous bound on the number of oracle queries proven in [LN14] is $O\left(\frac{n^3 \log \max_i \|\mathbf{b}_i\|}{\epsilon d^2}\right)$. For degenerate cases, $\max_i \|\mathbf{b}_i\|$ can be arbitrarily large (within the restriction that its logarithm is polynomial in the input size) even for LLL-reduced bases of lattices with determinant 1. (We focus on lattices with determinant 1 in this work for convenience. This is w.l.o.g. since one can always scale the lattice accordingly.) Theorem 1 confirms the conjecture of [HPS11]. Not only does it give a much stronger bound for slide reduction in case $k = 1$, it also gives a bound for block-Rankin reduction that depends on the overlap $k$ and improves for increasing $k$. This can be viewed as formalizing the intuition that a larger overlap leads to faster propagation of information within the basis. Of course, solving the DSP for larger $k$ is also harder and thus the complexity of the oracle itself will be larger and so will the overall running time.

In light of this it is natural to replace the DSP oracle by an oracle that approximates the DSP instead of solving it exactly. This suggests a variant, where the DSP problem is approximated using an HKZ oracle. We call this variant HKZ-slide reduction. It is inspired by recent observations in [ADH+19] that modern SVP solvers do not only find the shortest vector but approximately HKZ reduce the head of the basis essentially for free. Compared with slide reduction, increasing the overlap in HKZ-slide reduction decreases the running time at the cost of slightly increasing the length of the shortest vector found. We give heuristic arguments (Section 4.1) and experimental evidence (Section 4.2) that demonstrate that this trade-off can be very favorable in practice. A well chosen overlap yields a variant of slide reduction that we consider competitive with the

state-of-the-art in lattice block reduction [ADH+19]. When interpreting this result, it should be kept in mind that we did not explore all options to fine-tune the oracle to our algorithm and that BKZ has received considerable research effort to arrive at the performance level it is at now. This is not the case for slide reduction. To the best of our knowledge, this work is the first attempt of improving the practical performance of slide reduction beyond speeding up the SVP oracle.

### 1.2 Techniques

We define a class of algorithms, which we call Slide-type reductions, and use the dynamical systems analysis introduced in [HPS11] to analyze their behavior by studying the properties of a system $\mathbf{x} \mapsto \mathbf{A}\mathbf{x} + \mathbf{b}$. Here, the variable $\mathbf{x}$ is a function of the current basis during the execution and $\mathbf{A}$ and $\mathbf{b}$ depend on the reduction algorithm (see Section 2.2 for details). The fixed point of the system determines the result of the reduction and the norm of $\mathbf{A}$ its running time. After modeling Slide-type reductions in this way, we confirm that the fixed point yields the expected output quality as was proven in previous work for algorithms that fall into the class of Slide-type reductions, but we are actually more interested in the convergence of the system. Accordingly, we wish to study the norm of $\mathbf{A}$, which in our case has the following form:

$$\mathbf{A} = \begin{pmatrix} 1-2\beta & \beta & & \\ \beta & 1-2\beta & \beta & \\ & & \cdots & \\ & & \beta & 1-2\beta \end{pmatrix}$$

for some $0 < \beta \leq 1/4$ that depends on the parameters of the algorithm. Our goal is to bound some norm (induced by some vector $p$-norm) of $\mathbf{A}$ away from 1, i.e. show that $\|\mathbf{A}\|_p \leq 1 - \epsilon$ for some large enough $\epsilon > 0$. Clearly, this does not work for the row or column sum norm ($p = \infty$ and $p = 1$, respectively), since they are 1. We conjecture that the spectral norm ($p = 2$) is in fact smaller than 1, but this seems hard to prove directly. Instead, we apply a trick implicitly used by Neumaier [Neu17] to analyze SDBKZ: we apply a change of variable. We make Neumaier's trick explicit in Section 5 and then apply a similar change to our system. This results in a new matrix, for which we can easily bound the row sum norm ($p = \infty$), which implies our results.

### 1.3 Open Problems

Our results show that slide reduction finds short vectors in terms of the Hermite factor much faster than was previously proven. By using a well-known reduction due to Lovász [Lov86], one can also find short vectors in terms of the approximation factor at the cost of calling slide reduction $O(n)$ times, increasing the running time by this factor. However, the resulting approximation factor is somewhat worse than what is proved in [GN08a]. An interesting open

problem is whether one can prove that the approximation factor of [GN08a] can also be achieved with a number of oracle calls similar to our bound. Conversely, it might be that achieving this approximation factor does indeed require many more oracle calls.

We show in Section 4.2 that our variant of slide reduction is competitive with state-of-the-art reduction algorithms, but does not outperform them. However, given the lack of research into practical variants of slide reduction, we believe this might well be possible. We outline some avenues in Section 4.2 to improve our variant.

## 2 Preliminaries

*Notation.* Numbers and reals are denoted by lower case letters. For $n_1 \leq n_2 \in \mathbb{Z}$ we denote the set $\{n_1, \ldots, n_2\}$ by $[n_1, n_2]$. For vectors we use bold lower case letters and the $i$-th entry of a vector $\mathbf{v}$ is denoted by $v_i$. Let $\langle \mathbf{v}, \mathbf{w} \rangle = \sum_i v_i \cdot w_i$ be the scalar product of two vectors. If $p \geq 1$ the $p$ norm of a vector $\mathbf{v}$ is $\|\mathbf{v}\|_p = (\sum |v_i|^p)^{1/p}$. We will only be concerned with the norms given by $p = 1$, 2, and $\infty$. Whenever we omit the subscript $p$, we mean the standard Euclidean norm, i.e. $p = 2$. Matrices are denoted by bold upper case letters. The $i$-th column of a matrix $\mathbf{B}$ is denoted by $\mathbf{b}_i$ and the entry in row $i$ and column $j$ by $\mathbf{B}_{i,j}$. For any matrix $\mathbf{B}$ and $p \geq 1$ we define the induced norm to be $\|\mathbf{B}\|_p = \max_{\|\mathbf{x}\|_p=1} (\|\mathbf{B}\mathbf{x}\|_p)$. For $p = 1$ (resp. $\infty$) this is often denoted by the column (row) sum norm, since $\|\mathbf{B}\|_1 = \max_j \sum_i |\mathbf{B}_{i,j}|$ and $\|\mathbf{B}\|_\infty = \max_i \sum_j |\mathbf{B}_{i,j}|$; for $p = 2$ this is also known as the spectral norm, i.e. the largest singular value of $\mathbf{B}$.

### 2.1 Lattices

A *lattice* $\Lambda$ is a discrete subgroup of $\mathbb{R}^m$ and is generated by a matrix $\mathbf{B} \in \mathbb{R}^{m \times n}$, i.e. $\Lambda = \mathcal{L}(\mathbf{B}) = \{\mathbf{B}\mathbf{x} \colon \mathbf{x} \in \mathbb{Z}^n\}$. If $\mathbf{B}$ has full column rank, it is called a *basis* of $\Lambda$ and $\dim(\Lambda) = n$ is the dimension (or rank) of $\Lambda$. Any lattice of dimension larger than 1 has infinitely many bases, which are related to each other by right-multiplication with unimodular matrices. We use the notion of projected subblocks $\mathbf{B}_{[i,j]}$ for $i < j < n$, i.e. $\mathbf{B}_{[i,j]}$ is the matrix consisting of the columns $\mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_j$ projected onto the space orthogonal to $\mathrm{span}_{\mathbb{R}}(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{i-1})$. We define the *Gram-Schmidt-Orthogonalization* (GSO) $\mathbf{B}^*$ of $\mathbf{B}$, where the $i$-th column $\mathbf{b}_i^*$ of $\mathbf{B}^*$ is defined as $\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j<i} \mu_{i,j} \mathbf{b}_j^*$ and $\mu_{i,j} = \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle / \|\mathbf{b}_j^*\|^2$ (and $\mathbf{b}_1^* = \mathbf{b}_1$). In other words, $\mathbf{b}_i^* = \mathbf{B}_{[i,i]}$. For every basis of a lattice with dimension larger than 1 there are infinitely many bases that have the same GSO vectors $\mathbf{b}_i^*$, among which there is a (not necessarily unique) basis that minimizes $\|\mathbf{b}_i\|$ for all $i$. Transforming a basis into this form is commonly known as *size-reduction* and is easily and efficiently done using a slight modification of the Gram-Schmidt process. In this work, we will implicitly assume all bases to be size-reduced. The reader can simply assume that any basis operation is followed by a size-reduction.

Every lattice $\Lambda$ has invariants associated to it. One of them is its determinant $\det(\mathcal{L}(\mathbf{B})) = \prod_i \|\mathbf{b}_i^*\|$ for any basis $\mathbf{B}$. Note that this implies that for any two bases $\mathbf{B}$ and $\mathbf{B}'$ of the same lattice we have $\prod_i \|\mathbf{b}_i^*\| = \prod_i \|(\mathbf{b}_i')^*\|$ and the determinant is efficiently computable given any basis. Furthermore, for every lattice $\Lambda$ we denote the length of its shortest non-zero vector (also known as its *first minimum*) by $\lambda_1(\Lambda)$, which is always well defined. We use the short-hand notations $\det(\mathbf{B}) = \det(\mathcal{L}(\mathbf{B}))$ and $\lambda_1(\mathbf{B}) = \lambda_1(\mathcal{L}(\mathbf{B}))$ if no confusion may arise.

Hermite's constant is defined as $\gamma_n = \sup_{\Lambda:\dim(\Lambda)=n} (\lambda_1(\Lambda) / \det(\Lambda))^2$. Minkowski's theorem is a classic result that shows that $\gamma_n \leq n$. Viewing a shortest vector as the basis of a 1-dimensional sublattice of $\Lambda$ leads to a straightforward generalization of the first minimum to the densest $k$-dimensional sublattice $\mu_k(\Lambda) = \min_{\Lambda' \subset \Lambda:\dim(\Lambda')=k} \det(\Lambda')$. The corresponding generalization of Hermite's constant is known as Rankin's constant $\gamma_{k,n} = \sup_{\Lambda:\dim(\Lambda=n)} \left(\mu_k(\Lambda) / \det(\Lambda)^{k/n}\right)^2$.

There is a heuristic version of Minkowski's bound based on the Gaussian heuristic which states that most lattices that arise in practice satisfy $\lambda_1(\Lambda) \approx \sqrt{d/2\pi e} \det(\Lambda)^{1/n}$, unless there is an explicit reason to believe otherwise (e.g. an unusually short vector is planted in the lattice). We note that there is a theory of random lattices, which allows to turn this bound into a rigorous average-case version of Minkowski's bound, see e.g. [ALNS20] and references therein. For this work it is sufficient to know that the Gaussian heuristic is precise enough for lattices with dimension larger than 45 arising in lattice block reduction to predict its behavior in practice [CN11,GN08b,MW16].

**Heuristic 1** [Gaussian Heuristic] *For any lattice $\Lambda$ with $\dim(\Lambda) \geq 45$ arising in lattice reduction we assume that $\lambda_1(\Lambda) \approx \sqrt{d/2\pi e} \det(\Lambda)^{1/n}$.*

For every lattice $\Lambda$, its *dual* is defined as $\hat{\Lambda} = \{\mathbf{w} \in \mathrm{span}_{\mathbb{R}}(\Lambda) \,|\, \langle \mathbf{w}, \mathbf{v} \rangle \in \mathbb{Z} \text{ for all } \mathbf{v} \in \Lambda\}$. It is a classical fact that $\det(\hat{\Lambda}) = \det(\Lambda)^{-1}$. For a lattice basis $\mathbf{B}$, let $\widehat{\mathbf{B}}$ be the unique matrix that satisfies $\mathrm{span}_{\mathbb{R}}(\mathbf{B}) = \mathrm{span}_{\mathbb{R}}(\widehat{\mathbf{B}})$ and $\mathbf{B}^T\widehat{\mathbf{B}} = \widehat{\mathbf{B}}^T\mathbf{B} = \mathbf{R}$, where $\mathbf{R}$ is the identity matrix with reversed columns (see Section 5). Then $\widehat{\mathcal{L}(\mathbf{B})} = \mathcal{L}(\widehat{\mathbf{B}})$ and we denote $\widehat{\mathbf{B}}$ as the *reversed dual basis* of $\mathbf{B}$. Note that $\widehat{\mathbf{B}_{[i,j]}} = \widehat{\mathbf{B}}_{[n+1-j,n+1-i]}$.

**Definition 1.** *Let $\mathbf{B} \in \mathbb{Z}^{m \times n}$ be a lattice basis. We call $\mathbf{B}$ $k$-partial HKZ reduced if $\|\mathbf{b}_i^*\| = \lambda_1(\mathbf{B}_{[i,n]})$ for all $i \in [1,k]$.*

An $n$-dimensional basis $\mathbf{B}$ is SVP reduced (HKZ reduced), if it is 1-partial ($n$-partial, resp.) HKZ reduced. The *root Hermite factor* achieved by $\mathbf{B}$ is defined as $(\|\mathbf{b}_1\| / \det(\mathbf{B})^{1/n})^{1/n}$.

We use some notation from [HS07]:

**Definition 2.** *For a lattice basis $\mathbf{B}$ we define $\pi_{[j,k]}(\mathbf{B}) = \left(\prod_{i=j}^{k} \|\mathbf{b}_i^*\|\right)^{1/(k-j+1)}$ and $\Gamma_n(k) = \prod_{i=d-k}^{d-1} \gamma_{i+1}^{\frac{1}{2i}}$. We sometimes omit $\mathbf{B}$ and simply write $\pi_{[j,k]}$ if $\mathbf{B}$ is clear from context.*

Using these definitions, [HS07] proves useful inequalities regarding the geometry of ($k$-partial) HKZ reduced bases. We will use the following:

**Lemma 1 ([HS07]).** *If* $\mathbf{B}$ *is* $k$*-partial HKZ reduced, then*

$$\pi_{[1,k]} \leq \Gamma_d(k)^{d/k} \pi_{k+1,d}.$$

The proof is pretty straightforward using Minkowski's bound and induction.

**Definition 3.** *A basis* $\mathbf{B} \in \mathbb{R}^{m \times n}$ *is called* LLL-reduced[2] *if* $\|\mathbf{b}_i^*\| = \lambda_1\left(\mathbf{B}_{[i,i+1]}\right)$, *which implies* $\|\mathbf{b}_i^*\| \leq \gamma_2 \|\mathbf{b}_{i+1}^*\|$, *for all* $i \in [1, n-1]$.

We will need the following two facts about LLL.

**Fact 1** *If* $\mathbf{B} \in \mathbb{R}^{m \times n}$ *is LLL-reduced, then we have*

$$\pi_{[1,i]} \leq \gamma_2^{\frac{n-i}{2}} \pi_{[1,n]}$$

*for all* $i \in [1, n]$.

See e.g. [PT09] for a proof.

**Fact 2** *Let* $\mathbf{B} \in \mathbb{R}^{m \times n}$ *be a lattice basis and* $\mathbf{B'}$ *be the result of applying LLL to* $\mathbf{B}$. *Then we have*

$$\pi_{[1,i]}(\mathbf{B'}) \leq \pi_{[1,i]}(\mathbf{B})$$

*for all* $i \in [1, n]$.

Fact 2 can be seen to be true from a similar argument to the one showing that the potential function used to analyze LLL may only decrease under the swaps that LLL performs. More specifically, LLL reduction only applies two types of operations: size-reduction, which does not change the value $\pi_{[1,i]}(\mathbf{B})$ for any $i$, and swapping consecutive vectors. Swapping vectors only affects the value $\pi_{[1,i]}(\mathbf{B})$ for exactly one $i$ and the condition, under which such swaps are performed, ensures that this value can only decrease.

Finally, BKZ is a block-wise generalization of LLL.

**Definition 4.** *A basis* $\mathbf{B} \in \mathbb{R}^{m \times n}$ *is called* $d$-BKZ *reduced if* $\|\mathbf{b}_i^*\| = \lambda_1\left(\mathbf{B}_{[i,\ell]}\right)$, *where* $\ell = \min(i + d, n)$, *for all* $i \in [1, n]$.

[2] Technically, LLL reduction also requires size-reduction and usually contains a slack factor in the inequality to guarantee termination in polynomial time. Neither of these additional requirements are important for this work, so we ignore it here for simplicity.

### 2.2 Discrete-Time Affine Dynamical Systems

Consider some dynamical system

$$\mathbf{x} \mapsto \mathbf{A}\mathbf{x} + \mathbf{b} \tag{1}$$

and assume that $\|\mathbf{A}\|_p < 1$ for some $p$. This implies two facts:

1. Equation (1) has at most one fixed point $\mathbf{x}^* = \mathbf{A}\mathbf{x}^* + \mathbf{b}$, and
2. if Equation (1) has a fixed point $\mathbf{x}^*$ it converges to $\mathbf{x}^*$ exponentially fast in the number of iterations (with base $e^{-(1-\|\mathbf{A}\|_p)}$).

To see 1, note that two distinct fixed points $\mathbf{x}_1^* \neq \mathbf{x}_2^*$ would imply

$$0 \neq \|\mathbf{x}_1^* - \mathbf{x}_2^*\|_p = \|\mathbf{A}\left(\mathbf{x}_1^* - \mathbf{x}_2^*\right)\|_p \leq \|\mathbf{A}\|_p \|\mathbf{x}_1^* - \mathbf{x}_2^*\|_p < \|\mathbf{x}_1^* - \mathbf{x}_2^*\|_p$$

which is a contradiction. For 2, let $\mathbf{x}^*$ be the unique fixed point of Equation (1). We can write any input $\mathbf{x}'$ as $\mathbf{x}' = \mathbf{x}^* + \mathbf{e}$ for some "error vector" $\mathbf{e}$. When applying the system to it, we get $\mathbf{x}' \mapsto \mathbf{A}\mathbf{x}' + \mathbf{b} = \mathbf{x}^* + \mathbf{A}\mathbf{e}$. So the error vector $\mathbf{e}$ is mapped to $\mathbf{A}\mathbf{e}$. Applying this $\ell$ times maps $\mathbf{e}$ to $\mathbf{A}^\ell\mathbf{e}$, which means after $\ell$ iterations the error vector has norm $\|\mathbf{A}^\ell\mathbf{e}\|_p \leq \|\mathbf{A}^\ell\|_p\|\mathbf{e}\|_p$. Let $\|\mathbf{A}\|_p \leq 1 - \epsilon$ for some $\epsilon > 0$, then $\|\mathbf{A}^\ell\|_p \leq \|\mathbf{A}\|_p^\ell \leq (1-\epsilon)^\ell \leq e^{-\epsilon\ell}$, so the error vector will decay exponentially in $\ell$ with base $e^{-\epsilon}$ and the system converges to the fixed point $\mathbf{x}^*$.

Let $\mathbf{D}$ be an invertible matrix. We can use $\mathbf{D}$ for a change of variable to $\mathbf{y} = \mathbf{D}\mathbf{x}$, which allows to rewrite Equation (1) to

$$\mathbf{y} = \mathbf{D}\mathbf{x} \mapsto \mathbf{D}\mathbf{A}\mathbf{D}^{-1}\mathbf{y} + \mathbf{D}\mathbf{b} \tag{2}$$

It is easy to see that for any fixed point $\mathbf{x}^*$ of Equation (1), $\mathbf{y}^* = \mathbf{D}\mathbf{x}^*$ is a fixed point of Equation (2). This can be useful as it is often more convenient to bound $\|\mathbf{D}\mathbf{A}\mathbf{D}^{-1}\|_p$ for some $\mathbf{D}$ and $p$ than $\|\mathbf{A}\|_p$ (as we will see). If additionally the condition number $\kappa_p(\mathbf{D}) = \|\mathbf{D}\|_p\|\mathbf{D}^{-1}\|_p$ is small, then system (1) converges almost as quickly as system (2):

**Fact 3** *Let $\mathbf{x}^\ell$ be a vector resulting from applying system (1) $\ell$ times to the input $\mathbf{x}^0$ and denote $\mathbf{e}^\ell = \mathbf{x}^\ell - \mathbf{x}^*$. Let $\mathbf{D}$ be an invertible matrix such that $\|\mathbf{D}\mathbf{A}\mathbf{D}^{-1}\|_p = 1 - \epsilon$ for some $\epsilon > 0$. Then $\|\mathbf{e}^\ell\|_p \leq \exp(-\ell\epsilon)\,\kappa_p(\mathbf{D})\,\|\mathbf{e}^0\|_p$.*

*Proof.* Let $\mathbf{y}^0 = \mathbf{D}\mathbf{x}^0$ and $\mathbf{y}^{\ell+1} = \mathbf{D}\mathbf{A}\mathbf{D}^{-1}\mathbf{y}^\ell + \mathbf{D}\mathbf{b}$ for all $\ell > 0$. Induction shows that $\mathbf{y}^\ell = \mathbf{D}\mathbf{x}^\ell$. By above argument, we have $\|\mathbf{y}^\ell - \mathbf{y}^*\|_p \leq \exp(-\ell\epsilon)\,\|\mathbf{y}^0 - \mathbf{y}^*\|_p$. Now the result follows from

$$\begin{aligned}
\|\mathbf{e}^\ell\|_p &= \|\mathbf{x}^\ell - \mathbf{x}^*\|_p \\
&= \|\mathbf{D}^{-1}\mathbf{y}^\ell - \mathbf{D}^{-1}\mathbf{y}^*\|_p \\
&\leq \|\mathbf{D}^{-1}\|_p\|\mathbf{y}^\ell - \mathbf{y}^*\|_p \\
&\leq \exp(-\ell\epsilon)\,\|\mathbf{D}^{-1}\|_p\|\mathbf{y}^0 - \mathbf{y}^*\|_p \\
&\leq \exp(-\ell\epsilon)\,\|\mathbf{D}^{-1}\|_p\|\mathbf{D}\|_p\|\mathbf{e}^0\|_p.
\end{aligned}$$

□

*Application to Lattice Reduction.* Dynamical systems are a useful tool to study lattice reduction algorithms. As was first observed in [HPS11], for an iteration of some lattice reduction algorithm we can often show that $\mathbf{y} \leq \mathbf{A}\mathbf{x} + \mathbf{b}$, where $\mathbf{x}$ ($\mathbf{y}$) is some characterization of the input (output, resp.) basis for this iteration. If all entries in $\mathbf{A}$ are non-negative, we can iterate this inequality to derive inequalities for consecutive iterations. So the system $\mathbf{x} \mapsto \mathbf{A}\mathbf{x} + \mathbf{b}$ describes valid upper bounds for the vector $\mathbf{x}$ characterizing the current basis during the execution of the algorithm.

## 3 Slide-type Reductions

Let $\mathsf{O}_{k,d}$ be an oracle that takes as input an $n$-dimensional basis $\mathbf{B}$ and an index $i < n - d$ and modifies $\mathbf{B}$ such that $\pi_{[i,i+k-1]} \leq \alpha \cdot \pi_{[i,i+d-1]}$ (and leaves the rest unchanged). In Algorithm 1, we present a class of algorithms which resemble slide reduction and are parameterized by such an oracle $\mathsf{O}_{k,d}$. The algorithm runs in primal and dual tours. During a primal tour, the $n/d$ disjoint blocks of the basis are reduced using $\mathsf{O}_{k,d}$. Then the reversed dual basis is computed and $n/d - 1$ disjoint blocks are passed to the oracle. The blocks in the dual tour are chosen such that the corresponding primal blocks are shifted by $k$ with respect to the blocks in the primal tour. Slide reduction itself (or rather a natural variant) can be recovered by instantiating $\mathsf{O}_{k,d}$ with an SVP oracle in dimension $d$, hence $k = 1$ and $\alpha = \sqrt{\gamma_d}$. Block-Rankin reduction corresponds to using a $(k, d)$-DSP oracle, in which case $\alpha = \gamma_{k,d}^{1/2k}$. Finally, we can also define a new algorithm by letting $\mathsf{O}_{k,d}$ be an algorithm that $k$-partial HKZ reduces a $d$-dimensional basis. In that case, Lemma 1 implies $\alpha = \Gamma_d(k)^{(d-k)/k}$.

**Definition 5.** *Let $\mathsf{O}_{k,d}$ be an algorithm that $k$-partial HKZ reduces a $d$-dimensional basis. We call Algorithm 1 instantiated with $\mathsf{O}_{k,d}$ $(k, d)$-*HKZ-slide reduction.

**Algorithm 1** Slide-type Reduction. $\mathsf{O}_{k,d}$ is an oracle that takes as input a basis $\mathbf{B}$ and an index $i$ and modifies $\mathbf{B}$ such that $\pi_{[i,i+k-1]} \leq \alpha \cdot \pi_{[i,i+d-1]}$ (and leaves the rest unchanged.)

**procedure** SLIDE-TYPE REDUCTION($\mathbf{B}, \mathsf{O}_{k,d}(\cdot,\cdot)$)
  **while** progress is made **do**
    $\mathbf{B} \leftarrow \mathsf{O}_{k,d}(\mathbf{B}, id + 1)$ for all $i \in [0, n/d - 1]$
    $\mathbf{B} \leftarrow \widehat{\mathbf{B}}$
    $\mathbf{B} \leftarrow \mathsf{O}_{k,d}(\mathbf{B}, id - k)$ for all $i \in [1, n/d - 1]$
    $\mathbf{B} \leftarrow \widehat{\mathbf{B}}$
  **end while**
**end procedure**

We remark that it is customary in lattice reduction theory to apply LLL reduction in between the calls to the oracle. This is important to control the

size of the numbers, which in turn allows to bound the complexity of the oracle itself. Since we focus on the number of calls to the oracle, we chose to present Algorithm 1 without any calls to LLL. Note that none of such calls will have any effect on our bounds due to Fact 2, since we will work with upper bounds on the subdeterminants $\pi_{[1,i]}$. These can only decrease during the application of LLL, so any upper bound that held before applying LLL also holds afterwards.

### 3.1 Convergence

The following theorem contains the main technical contribution of this work and the remainder of this subsection is devoted to proving it.

**Theorem 2.** *Let $\mathbf{B} \in \mathbb{R}^{m \times n}$ be a lattice basis with $\det(\mathcal{L}(\mathbf{B})) = 1$. Let $k \leq d \in \mathbb{Z}$ such that $n = pd$ for some $d \in \mathbb{Z}$, $p \geq 2$ and $\mathsf{O}_{k,d}$ be an oracle that on input a basis $\mathbf{B}'$ and index $i < n - d$ produces a basis $\mathbf{C}$ such that*

- *$\pi_{[i,i+k-1]}(\mathbf{C}) \leq \alpha \cdot \pi_{[i,i+d-1]}(\mathbf{B}')$ and*
- *$\mathbf{c}_j = \mathbf{b}'_j$ for all $j \notin [i, i+d-1]$.*

*Let $\mu_i = i(p-i)\frac{d}{d-k}\ln\alpha$, $\mathbf{B}_\ell$ the basis after the $\ell$-th iteration and $\epsilon_\ell = \max_{i\in[1,p]} |\ln\left(\pi_{[1,id]}(\mathbf{B}_\ell)\right) - \mu_i|$. Then we have*

$$\epsilon_\ell \leq \exp\left(\frac{-4k(d-k)}{n^2}\ell\right)\frac{p^2}{4(p-1)}\epsilon_0$$

*after $\ell$ iterations of Slide-type reduction with oracle $\mathsf{O}_{k,d}$.*

*Proof.* During a primal tour, Slide-type reduction turns a basis $\mathbf{B}$ into a basis $\mathbf{B}'$ such that

$$\ln\pi_{[id+1,id+k]}(\mathbf{B}') \leq \ln\pi_{[id+1,id+d]}(\mathbf{B}) + \ln\alpha \tag{3}$$

for $i \in [0, p-1]$. Similarly, a dual tour yields

$$\ln\pi_{[id+1,id+k]}(\mathbf{B}') \geq \ln\pi_{[(i-1)d+2,id+1]}(\mathbf{B}) - \ln\alpha \tag{4}$$

We consider the leading subdeterminants corresponding to the blocks considered by Algorithm 1. Let $y_i = id\ln\pi_{[1,id]}(\mathbf{B})$ for $i \in [1, p-1]$. (Note that $y_p = 0$, since we assume that the lattice has unit determinant, so we may ignore this variable.) Now we apply a primal tour and denote $x_i = ((i-1)d+k)\ln\pi_{[1,(i-1)d+k]}(\mathbf{B}')$ for $i \in [1, p]$ after that tour. Then we have by Equation (3)

$$x_i \leq \frac{d-k}{d}y_{i-1} + \frac{k}{d}y_i + k\ln\alpha$$

for $i \in [1, p]$, where we define $y_0 = y_p = 0$. In matrix form we have $\mathbf{x} \leq \mathbf{A}_p\mathbf{y} + \mathbf{b}_p$ with

$$\mathbf{A}_p = \begin{pmatrix} \frac{k}{d} & & & \\ \omega & \frac{k}{d} & & \\ & \omega & \frac{k}{d} & \\ & & \cdots & \\ & & & \frac{k}{d} \\ & & & \omega \end{pmatrix} \in \mathbb{R}^{p\times(p-1)}$$

where $\omega = \frac{d-k}{d}$ and $\mathbf{b}_p = k \ln \alpha \cdot \mathbf{1} \in \mathbb{R}^p$.

Now let $y'_i$ as $y_i$ above but after the next dual tour. From Equation (4) we get

$$x_i - y'_{i-1} \geq \frac{k}{d}(x_i - x_{i-1}) - k \ln \alpha$$

or equivalently

$$y_i \leq \omega x_{i+1} + \frac{k}{d} x_i + k \ln \alpha$$

for $i \in [1, p-1]$. Again, in matrix form $\mathbf{y} \leq \mathbf{A}_d \mathbf{x} + \mathbf{b}_d$, where $\mathbf{b}_d = k \ln \alpha \cdot \mathbf{1} \in \mathbb{R}^{p-1}$ and

$$\mathbf{A}_d = \begin{pmatrix} \frac{k}{d} & \omega & & \\ & \frac{k}{d} & \omega & \\ & & \cdots & \\ & & & \frac{k}{d} \ \omega \end{pmatrix} = \mathbf{A}_p^T$$

By combining the two set of inequalities, we obtain:

$$\mathbf{y}' \leq \mathbf{A}_d \mathbf{x} + \mathbf{b}_d \leq \mathbf{A}_d (\mathbf{A}_p \mathbf{y} + \mathbf{b}_p) + \mathbf{b}_d = \mathbf{A}_p^T \mathbf{A}_p \mathbf{y} + \left(\mathbf{A}_p^T \mathbf{b}_p + \mathbf{b}_d\right)$$

Thus, the general matrix that characterizes a primal and dual tour is

$$\mathbf{A} = \mathbf{A}_p^T \mathbf{A}_p = \begin{pmatrix} \tilde{\omega} & \frac{k\omega}{d} & & \\ \frac{k\omega}{d} & \tilde{\omega} & \frac{k\omega}{d} & \\ & \cdots & & \\ & & \frac{k\omega}{d} & \tilde{\omega} \end{pmatrix} = \begin{pmatrix} 1-2\beta & \beta & & \\ \beta & 1-2\beta & \beta & \\ & & \cdots & \\ & & \beta & 1-2\beta \end{pmatrix} \in \mathbb{R}^{(p-1)\times(p-1)} \tag{5}$$

where $\tilde{\omega} = \omega^2 + (k/d)^2$ and $\beta = \frac{k(d-k)}{d^2}$. And with $\mathbf{b} = \mathbf{A}_p^T \mathbf{b}_p + \mathbf{b}_d = 2 \cdot \mathbf{b}_d$ the dynamical system we are interested in is

$$\mathbf{y} \mapsto \mathbf{A}\mathbf{y} + \mathbf{b}. \tag{6}$$

The theorem now follows from Lemma 2 and 3 below, in which we analyze the fixed point and the convergence of system (6), resp. □

**Lemma 2.** *For the system in Equation (6) and the vector* $\mathbf{y}^* \in \mathbb{R}^{p-1}$ *with*

$$y_i^* = i\,(p-i)\,\frac{d^2}{d-k} \ln \alpha$$

*we have* $\mathbf{A}\mathbf{y}^* + \mathbf{b} = \mathbf{y}^*$.

*Proof.* Note that we can extend the definition of $y_i^*$ to $i = 0$ and $i = p$, in which case we have $y_0^* = y_p^* = 0$. So the lemma follows if we can show that

$$\beta y_{i-1}^* + (1-2\beta)\, y_i^* + \beta y_{i+1}^* + 2k \ln \alpha = y_i^*$$

for all $i \in [1, p-1]$. This is equivalent to

$$\beta \left(y_{i-1}^* + y_{i+1}^* - 2y_i^*\right) + 2k \ln \alpha = 0$$

which is easily seen to be true by straightforward calculation. □

**Lemma 3.** *Let* $\mathbf{A}$ *as in Equation (5). Then there exists an invertible matrix* $\mathbf{D}$ *with* $\kappa_\infty(\mathbf{D}) = \frac{p^2}{4(p-1)}$ *such that*

$$\|\mathbf{DAD}^{-1}\|_\infty \leq 1 - \frac{4k(d-k)}{n^2}$$

*for any* $p \geq 2$.

*Proof.* Let $\mathbf{D}$ be the diagonal matrix such that

$$\mathbf{D}^{-1} = \begin{pmatrix} p-1 & & & \\ & 2(p-2) & & \\ & & \cdots & \\ & & & p-1 \end{pmatrix}$$

We now analyze the matrix

$$\mathbf{DAD}^{-1} = \begin{pmatrix} 1-2\beta & \frac{2(p-2)}{p-1}\beta & & \\ \frac{p-1}{2(p-2)}\beta & 1-2\beta & \frac{3(p-3)}{2(p-2)}\beta & \\ & \frac{2(p-2)}{3(p-3)}\beta & 1-2\beta & \frac{4(p-4)}{3(p-3)}\beta \\ & & \cdots & \\ & & \frac{(p-2)2}{p-1}\beta & 1-2\beta \end{pmatrix}$$

The sum of the $i$-th row is

$$\begin{aligned} S_i &= 1 - 2\beta + \beta\frac{(i-1)(p-i+1) + (i+1)(p-i-1)}{i(p-i)} \\ &= 1 - 2\beta\left(1 - \frac{ip - i^2 - 1}{ip - i^2}\right) \\ &= 1 - \frac{2\beta}{ip - i^2} \\ &\leq 1 - \frac{8\beta}{p^2} \\ &= 1 - \frac{8k(d-k)}{n^2} \end{aligned}$$

for $i \in [2, \ldots, p-2]$. Finally, we have

$$S_1 = S_{p-1} \leq 1 - \frac{2pk(d-k)}{n^2}$$

from which the lemma follows. □

### 3.2 Implications

We now show how Theorem 2 implies bounds for the running time of Slide-type reduction algorithms.

**Corollary 1.** *Let* $\mathbf{B} \in \mathbb{R}^{m \times n}$ *be an LLL-reduced lattice basis with* $\det(\mathcal{L}(\mathbf{B})) = 1$ *and* $\epsilon > 0$ *an arbitrary constant. After* $\ell \geq \frac{n^2}{4k(d-k)} \ln\left(\frac{\frac{n^2}{2d} + \frac{n^3}{4d^3} \ln \alpha}{\epsilon}\right)$ *tours of Slide-type reduction with oracle* $\mathsf{O}_{k,d}$ *such that* $\alpha \geq \gamma_2$*, the output basis satisfies*

$$\pi_{[1,d]} = \prod_{i=1}^{d} \|\mathbf{b}_i^*\|^{\frac{1}{d}} \leq \exp(\epsilon + \mu_1) \approx (1+\epsilon)\, \alpha^{\frac{n-d}{d-k}}.$$

*Accordingly, the number of oracle queries is bounded by* $\frac{n^3}{2dk(d-k)} \ln\left(\frac{\frac{n^2}{2d} + \frac{n^3}{4d^3} \ln \alpha}{\epsilon}\right)$.

*Proof.* Theorem 2 shows that in order to obtain $\epsilon_\ell \leq \epsilon$ for arbitrary $\epsilon > 0$, it is sufficient to set

$$\ell \geq \frac{n^2}{4k(d-k)} \ln\left(\frac{p^2 \epsilon_0}{4(p-1)\epsilon}\right).$$

By Fact 1 we have

$$\epsilon_0 = \max_{i \in [1,p]} |\ln \pi_{[1,id]}(\mathbf{B}) - \mu_i| \leq \frac{n-1}{2} \ln \gamma_2 + \frac{n^2}{4d(d-k)} \ln \alpha \leq n + \frac{n^2}{2d^2} \ln \alpha$$

where we assume that $k \leq d/2$. Finally, notice that $p^2/(4(p-1)) \leq p/2 = n/2d$ for all $p \geq 2$. □

Corollary 1 implies the following corollaries.

**Corollary 2.** *Let* $\mathbf{B} \in \mathbb{R}^{m \times n}$ *be an LLL-reduced lattice basis with* $\det(\mathcal{L}(\mathbf{B})) = 1$ *and* $\epsilon > 0$ *an arbitrary constant. After* $O\left(\frac{n^3}{dk(d-k)} \ln\left(\frac{n}{\epsilon}\right)\right)$ *calls to the* $(k,d)$*-DSP oracle, the output basis of block-Rankin reduction satisfies*

$$\pi_{[1,d]} = \prod_{i=1}^{d} \|\mathbf{b}_i^*\|^{\frac{1}{d}} \leq \exp(\epsilon + \mu_1) = \exp(\epsilon)\, \gamma_{k,d}^{\frac{n-d}{2k(d-k)}} \approx (1+\epsilon)\, \gamma_{k,d}^{\frac{n-d}{2k(d-k)}}.$$

*One more call to the oracle yields*

$$\pi_{[1,k]} \leq \exp(\epsilon)\, \gamma_{k,d}^{\frac{n-k}{2k(d-k)}} \approx (1+\epsilon)\, \gamma_{k,d}^{\frac{n-k}{2k(d-k)}}.$$

The case of slide reduction follows as a special case ($k = 1$) and we note that the number of SVP calls matches the one proven for other lattice reduction algorithms using this technique [HPS11,LN20,MW16]. Recall that the bound on the number of oracle queries proven in [LN14] is $O\left(\frac{n^3 \log \max_i \|\mathbf{b}_i\|}{\epsilon d^2}\right)$. For degenerate cases $\max_i \|\mathbf{b}_i\|$ can be arbitrarily large (within the restriction that its logarithm is polynomial in the input size) even for LLL-reduced bases of lattices with determinant 1. Similar to the recent work of [LN20], we are able to achieve a bound that is independent of $\max_i \|\mathbf{b}_i\|$ using the dynamical systems approach. The length of the vectors just contributes to the $\log n$ factor in our bound. ([HPS11] does not claim to achieve this but obtains a bound with a

doubly logarithmic dependence on $\max_i \|\mathbf{b}_i\|$.) Furthermore, the dependence on $\epsilon$ is much tighter in two ways: 1) in [LN14] the slack factor in the output quality is $(1+\epsilon)^{(n-k)/(4(d-k))}$, while in Corollary 2 it is just $\exp(\epsilon) \approx (1+\epsilon)$. 2) The dependence of the bound on the number of oracle queries is linear in $1/\epsilon$, while in our bound it is only logarithmic. Finally, the remaining polynomial factor matches in the two bounds for small values of $k$, but our bound depends on $k$ and actually decreases with growing $k$ up to an improvement of $1/d$ for $k = d/2$. This seems to be a feature of the dynamical systems analysis as it is unclear if the LLL-style potential function analysis of [LN14] can be used to study the dependence of the number of calls on $k$.

**Corollary 3.** *Let $\mathbf{B} \in \mathbb{R}^{m \times n}$ be an LLL-reduced lattice basis with $\det(\mathcal{L}(\mathbf{B})) = 1$ and $\epsilon > 0$ an arbitrary constant. After $O\left(\frac{n^3}{dk(d-k)} \ln\left(\frac{n}{\epsilon}\right)\right)$ calls to the $k$-partial HKZ oracle, the output basis of $(k,d)$-HKZ-slide reduction satisfies*

$$\pi_{[1,d]} = \prod_{i=1}^{d} \|\mathbf{b}_i^*\|^{\frac{1}{d}} \leq \exp(\epsilon + \mu_1) = \exp(\epsilon)\, \Gamma_d(k)^{\frac{n-d}{k}} \approx (1+\epsilon)\, \Gamma_d(k)^{\frac{n-d}{k}}.$$

*One more call to the oracle yields*

$$\|\mathbf{b}_1\| \leq \exp(\epsilon) \sqrt{\gamma_d} \Gamma_d(k)^{\frac{n-d}{k}} \approx (1+\epsilon) \sqrt{\gamma_d} \Gamma_d(k)^{\frac{n-d}{k}}.$$

We can try to get bounds on the Hermite factor of $(k,d)$-HKZ-slide reduction in terms of $\gamma_d$ by using some straightforward bounds on $\Gamma_d(k)$.

**Lemma 4.** *For a $(k,d)$-HKZ-slide reduced basis we have*

$$\|\mathbf{b}_1\| \leq \sqrt{d}^{1+\frac{n-d}{k}\log\frac{d}{d-k}} \det(\mathbf{B})^{\frac{1}{n}} \leq \sqrt{d}^{\frac{n-k}{d-k}} \det(\mathbf{B})^{\frac{1}{n}} \tag{7}$$

$$\|\mathbf{b}_1\| \leq \sqrt{\gamma_{d-k+1}}^{\frac{n-1}{d-k}} \det(\mathbf{B})^{\frac{1}{n}} \tag{8}$$

*Proof.* Both follow from Corollary 3. For Equation (7) use the bound $\Gamma_d(k) \leq \sqrt{d}^{\log\frac{d}{d-k}}$ proven in [HS07] and $\log 1 + x \leq x$.

For Equation (8), recall Mordell's inequality $\gamma_n^{\frac{1}{n-1}} \leq \gamma_k^{\frac{1}{k-1}}$, which shows that $\Gamma_d(k) \leq \sqrt{\gamma_{d-k+1}}^{\frac{k}{d-k}}$. So we have

$$\|\mathbf{b}_1\| \leq \sqrt{\gamma_d} \sqrt{\gamma_{d-k+1}}^{\frac{n-d}{d-k}} \det(\mathbf{B})^{\frac{1}{n}}.$$

Finally, use Mordell's inequality again to see that $\sqrt{\gamma_d} \leq \sqrt{\gamma_{d-k+1}}^{\frac{d-1}{d-k}}$ to conclude. □

The bound on the Hermite factor achieved by HKZ-slide reduction suggests that running $(k,d)$-HKZ-slide reduction is no better than running $(1, d-k+1)$-HKZ-slide reduction, i.e. vanilla slide reduction with block size $d-k+1$. Since solving SVP in dimension $d-k+1$ is easier by a factor $2^{\Omega(k)}$ than $k$-partial HKZ reduction in dimension $d$, it stands to reason that using $k=1$ is optimal. However, in the next sections we will make heuristic arguments and show experimental evidence that using larger $k$ can be worthwhile in practice.

## 4 HKZ-Slide Reduction in Practice

In this section we give heuristic arguments (Section 4.1) and experimental evidence showing that HKZ-slide reduction can outperform slide reduction and yield a faster algorithm in practice.

### 4.1 Heuristic Analysis

Note that the convergence analysis in Section 3.1 is agnostic to the value $\alpha$. So we can use the same analysis for a heuristic evaluation, but instead of using Minkowski's inequality, we use the Gaussian heuristic. So by defining $g_d = \sqrt{d/2\pi e}$ and $\alpha = G_d(k) = \prod_{i=d-k}^{d-1} g_{i+1}^{\frac{1}{i}}$ we can get a bound on the density of the first block of a $(k,d)$-HKZ-slide reduced basis based on Heuristic 1, which is

$$\pi_{[1,d]} \approx G_d(k)^{\frac{n-d}{k}} \det(\mathbf{B})^{\frac{1}{n}}$$

which implies

$$\|\mathbf{b}_1\| \approx g_d G_d(k)^{\frac{n-d}{k}} \det(\mathbf{B})^{\frac{1}{n}}.$$

Now we can compare the quality that we achieve by using different overlaps and block sizes. See Figure 1 for an example. Running $(k,d)$-HKZ-slide reduction yields a better basis than running slide reduction with block size $k-d+1$ (but also needs a partial HKZ oracle in larger dimension).

To estimate the practical behavior of HKZ-slide reduction and slide reduction, we make the following assumptions: 1) we assume that the dependence of the running time of $(k,d)$-HKZ-slide reduction on the overlap $k$ is $1/k(d-k)$, and 2) that the complexity of the $k$-partial HKZ oracle is $2^{d/3+O(1)}$ and independent of $k$. The first assumption is supported by our analysis in Section 3.1. The second assumption is supported by the observation in [ADH+19] that SVP oracles in practice tend to not only find the shortest vector in a lattice, but additionally HKZ reduce the head of the basis "for free". The complexity of the oracle is a crude estimate of heuristic bounds on the complexity of sieving. More accurate estimates are a little smaller than what we assumed above. Adapting the following argument would thus provide slightly better results.

As a baseline for our comparison we select 90-slide reduction on a 270 dimensional lattice and analyze how reducing the block size to $90-k'$ and increasing the overlap to $k$ compare in terms of speed-up while ensuring that both yield similar output quality. Specifically, for every $k$ we numerically compute $k' < k$ such that $(90-k')$-slide reduction achieves similar root Hermite factor as $(k,90)$-HKZ-slide reduction. The speed-up of $(k,90)$-HKZ-slide reduction over 90-slide reduction is $k(d-k)/(d-1)$ given our assumptions. The speed-up achieved by $(90-k')$-slide reduction is $2^{k'/3}(d-k'+1)/d$. (We ignore the issue of divisibility of block size and lattice dimension here for simplicity.) The ratio of the two quantities is given in Figure 2. The figure suggests that $(k,90)$-HKZ-slide reduction with a well-chosen overlap $k$ can be up to 4 times faster than slide reduction with similar output quality.

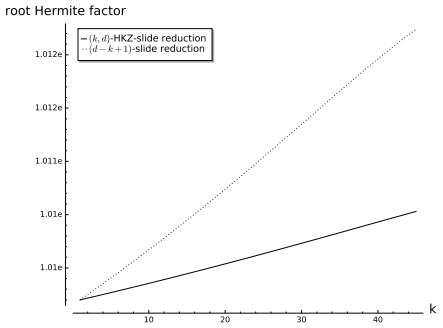



Fig. 1: Comparison of root Hermite factor for running $(k, 90)$-HKZ-slide reduction on a basis with dimension 270 vs $(90 - k)$-slide reduction

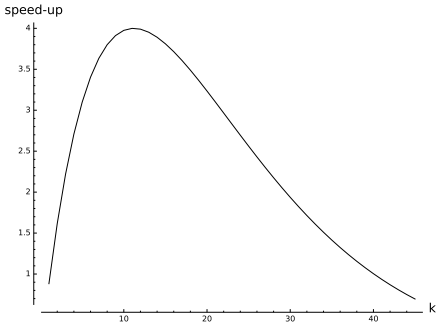



Fig. 2: Speed-up factor of running $(k, 90)$-HKZ-slide reduction on a basis with dimension 270 vs $(90 - k')$-slide reduction with comparable Hermite factor.

### 4.2 Experiments

We provide an implementation of HKZ-slide reduction[3] in the G6K framework of [ADH+19], which (among a lot of other things) provides an interface to an SVP algorithm based on sieving. The authors observe that, in fact, the output of this algorithm seems to approximate partial-HKZ reduction. Their work also shows that basic (called *naive* in [ADH+19]) BKZ based on sieving starts outperforming state-of-the-art enumeration based methods for block sizes below 80, and more carefully tuned variants well below 65.

For our implementation we treat the SVP algorithm of G6K as a $k$-partial-HKZ oracle for arbitrary $k \leq 15$, which seems justified by the observations made in [ADH+19]. To test the hypothesis of the previous section, we run $(k, d)$-HKZ-slide reduction for $k \in \{1, 5, 10, 15\}$ and $d \in \{60, 85\}$ on lattices from the lattice challenge [BLR08]. To avoid issues with block sizes not dividing the dimension we select the dimension as the largest integer multiple of $d$ such that the algorithm does not run into numerical issues. For $d = 60$ and $d = 85$, this is $n = 180$ (i.e. $p = 3$ blocks) and $n = 170$ (i.e. $p = 2$ blocks), respectively. The results are shown in Figures 3a and 3c. All data points are averaged (in both axes) over the same 10 lattices (challenge seeds 0 to 9), which are preprocessed using `fplll` [dt16] with block size 45 (for $d = 60$) and 60 (for $d = 85$).

Figure 3a demonstrates that for relatively small block sizes, the behavior of $k$-HKZ-slide reduction is actually better than expected: not only does a larger $k$ lead to a faster convergence (which is expected), all of the tested $k$ also lead to better output quality. This can at least in part be explained by the relatively small block size and the corresponding approximation error of the Gaussian

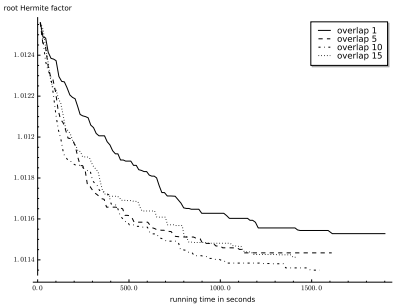
[3] Code available at: `http://pub.ist.ac.at/~mwalter/publication/hkz_slide/hkz_slide.zip`

heuristic. This is supported by Figure 3c, where at least the overlaps $k = 5$ and $k = 15$ behave as expected: faster convergence but poorer output quality. (Note though that the difference in output quality between overlaps 1 and 5 is minor.) However, the case of $k = 10$ seems to be a special case that behaves exceptionally well even for large block size. We cannot explain this phenomenon beyond baseless speculation at this point and leave an in-depth investigation to future work. In summary, we believe that the results give sufficient evidence that the trade-off achieved by HKZ-slide reduction can indeed be very favorable when considering overlaps larger than 1 (i.e. beyond slide reduction).

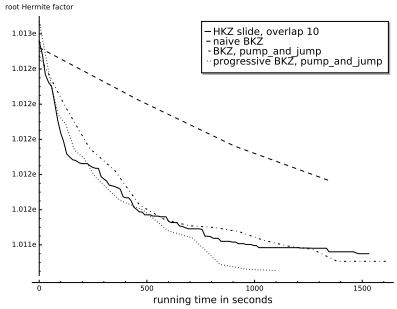
To put the results into context, we also compare HKZ-slide reduction with the BKZ variants implemented in G6K on the same lattices. For HKZ-slide reduction we chose $k = 10$. We compared to three "standard" variants of BKZ: 1) naive BKZ, which treats the SVP algorithm as a black box; 2) the "Pump and Jump" (PnJ) variant, which recycles computation done during previous calls to the SVP algorithm to save cost in later calls; 3) a progressive variant of the PnJ strategy, which starts with smaller block sizes and successively runs BKZ tours with increasing block size. We leave all parameters for the PnJ versions at their default. [ADH+19] reported that some fine-tuning can improve the PnJ variant further, but since our goal is only to demonstrate the competitiveness of HKZ-slide reduction rather than a fine-grained comparison, we do not believe such fine-tuning is necessary here. Naive BKZ and the PnJ variant is called with the same block size (on the same bases as HKZ-slide reduction) and the number of tours is chosen such that the running time is roughly in the ballpark of the HKZ-slide reduction experiments. For progressive PnJ, we run 1 tour of each block size starting from $d - 10$ up to $d + 5$, where $d$ is the block size chosen for the other algorithms. The results are shown in Figure 3b and 3d respectively. They show that HKZ-slide reduction can outperform the naive version of BKZ significantly, but it also seems to be better than PnJ. However, progressive PnJ seems to have the edge over HKZ-slide reduction, but we consider the latter at least competitive.

*Caveats.* We focus our attention in these experiments on the root Hermite factor that the different algorithms achieve in a given amount of time. This has been established as the main measure of output quality for lattice reduction, since they are usually used to find short vectors. When targeting a short vector, (HKZ-) slide reduction has the advantage that it focuses on improving a set of pivot points distributed across the basis, while BKZ attempts to improve the entire basis. This seems to result in a lower cost for slide reduction. But finding short vectors is not the only use case: often one is interested in a basis that is reduced according to a more global measure, e.g. one wants all basis vectors to be short or the GSO vectors should not drop off too quickly. In this case, BKZ seems to be the more natural choice.

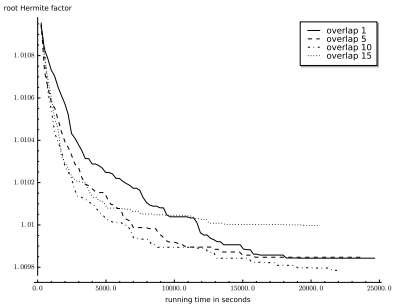
*Potential Improvements.* We do not make any attempts to fine-tune the SVP oracle to HKZ-slide reduction and its parameters. The SVP-oracle itself has several



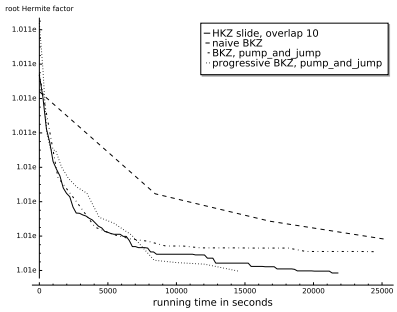

(a) HKZ-slide reduction on a lattice with dimension 180 and block size 60




(b) Comparison of HKZ-slide reduction and BKZ on a lattice with dimension 180 and block size 60




(c) HKZ-slide reduction on a lattice with dimension 170 and block size 85




(d) Comparison of HKZ-slide reduction and BKZ on a lattice with dimension 170 and block size 85

Fig. 3: Comparison of HKZ-slide-reduction with different overlaps and with various BKZ variants

parameters which potentially influence how well it performs as a $k$-partial-HKZ oracle. We leave such a fine-tuning as interesting future work.

Furthermore, we note that applying BKZ/PnJ with increasing block sizes results in significant improvements. It stands to reason that including an element of "progressiveness" could significantly improve HKZ-slide reduction. However, the strength of HKZ-slide reduction of focusing its attention on pivot points instead of the entire basis could be a disadvantage here: it may not be as suitable as a preprocessing for other algorithms, possibly including itself. Still, finding an effective way of naturally progressing slide reduction might lead to improvements, but we believe simply increasing the block size is unlikely to be sufficient here. Finally, given the above observations, a natural approach seems to be to use progressive BKZ/PnJ as a preprocessing and only apply HKZ-slide reduction in the final step to find a short vector.

## 5 SDBKZ: Revisiting Neumaier's Analysis

We conclude this work by revisiting Neumaier's analysis [Neu17] of SDBKZ [MW16]. Using a change of variable allows us to recast it as a variant of the conventional dynamic analysis. The matrix used in Section 3 for the change of variable was inspired by this reformulation.

### 5.1 Reminders

We first give a brief description of the SDBKZ algorithm and the analysis from [MW16]. The algorithm can be viewed as iterating the following 2 steps:

1. perform a forward tour by applying the SVP oracle successively to the projected blocks of the basis (i.e. a truncated BKZ tour)
2. compute the reversed dual of the basis.

For convenience, the SDBKZ lattice reduction algorithm is provided as Algorithm 2.

**Algorithm 2** SDBKZ. $\mathsf{O}_d$ is an oracle that takes as input a basis $\mathbf{B}$ and an index $i$ and modifies $\mathbf{B}$ such that $\mathbf{B}_{[i,i+d-1]}$ is SVP reduced (and leaves the rest unchanged.)

**procedure** SDBKZ($\mathbf{B}, \mathsf{O}_d(\cdot,\cdot)$)
  **while** progress is made **do**
    $\mathbf{B} \leftarrow \mathsf{O}_d(\mathbf{B}, i)$ for all $i \in [0, n-d]$
    $\mathbf{B} \leftarrow \widehat{\mathbf{B}}$
  **end while**
**end procedure**

Let $\mathbf{B}$ be a lattice basis. In [MW16], the following variables were considered

$$\mathbf{x} = (\log \det(\mathbf{b}_1, \ldots, \mathbf{b}_{d+i-1}))_{1 \le i \le n-d}.$$

When applying the two steps of SDBKZ to a lattice basis, [MW16] showed that for the output basis $\mathbf{B}'$ we have $\mathbf{x}' \le \mathbf{R}\mathbf{A}\mathbf{x} + \mathbf{R}\mathbf{b}$, where

$$\mathbf{b} = \alpha d \begin{bmatrix} 1-\omega \\ \vdots \\ 1-\omega^{n-d} \end{bmatrix} \quad \mathbf{A} = \frac{1}{d} \begin{bmatrix} 1 & & & \\ \omega & 1 & & \\ \vdots & \ddots & \ddots & \\ \omega^{n-d-1} & \cdots & \omega & 1 \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} & & & 1 \\ & & 1 & \\ & \iddots & & \\ 1 & & & \end{bmatrix}$$

$\alpha = \frac{1}{2}\log\gamma_d$ and $\omega = (1 - \frac{1}{d})$. This lead to the analysis of the dynamical system

$$\mathbf{x} \mapsto \mathbf{RAx} + \mathbf{Rb}. \tag{9}$$

[MW16] showed that this system has exactly one fixed point $\mathbf{x}^*$ with

$$x_i^* = \frac{(d+i-1)(n-d-i+1)}{d-1}\alpha$$

which can be used to obtain bounds on the output quality of the algorithm. Here we are more interested in the convergence analysis. For this, note that

$$\|\mathbf{RA}\|_\infty = \|\mathbf{A}\|_\infty = 1 - \omega^{n-d}$$

which means that the number of tours required to achieve $\|\mathbf{e}\|_\infty \leq c$ for some constant $c$ is proportional to $\exp((n-d)/d)$. This is polynomial as long as $d = \Omega(n)$, but for $d = o(n)$ this results in a superpolynomial bound.

### 5.2 Neumaier's Analysis

As stated above, Neumaier's analysis of SDBKZ [Neu17] can be viewed as a change of variable for $\mathbf{x}$. Neumaier implicitly chose the diagonal matrix

$$\mathbf{D^{-1}} = \begin{bmatrix} d(n-d) & & & \\ & (d+1)(n-d-1) & & \\ & & \ddots & \\ & & & n-1 \end{bmatrix}$$

which yields the new fixed point $\mathbf{y}^* = \frac{\alpha}{d-1}\mathbf{1}$ (cf. $\mu_s$ from [Neu17]). We now analyze the matrix $\mathbf{A'} = \mathbf{DRAD}^{-1}$: First, we observe that

$$\mathbf{A}_{ij} = \begin{cases} \frac{1}{d}\omega^{i-j} & i \geq j \\ 0 & i < j \end{cases}$$

and so

$$(\mathbf{RA})_{ij} = \begin{cases} \frac{1}{d}\omega^{(n-d+1-i)-j} & i+j \leq n-d+1 \\ 0 & i+j > n-d+1 \end{cases}$$

and finally

$$\mathbf{A}'_{ij} = (\mathbf{DRAD}^{-1})_{ij} = \begin{cases} \frac{(d+j-1)(n-d-j+1)}{d(d+i-1)(n-d-i+1)}\omega^{(n-d+1-i)-j} & i+j \leq n-d+1 \\ 0 & i+j > n-d+1 \end{cases} \tag{10}$$

**Lemma 5.** *Let $\mathbf{A'}$ as defined in Equation (10). Then, $\|\mathbf{A'}\|_\infty \leq 1 - \epsilon$, where $\epsilon = \left(1 + \frac{n^2}{4d(d-1)}\right)^{-1}$.*

*Proof.* Let $S_i = \sum_j \mathbf{A}'_{ij}$ be the sum of every row in $\mathbf{A}'$. We have

$$\begin{aligned} S_i &= \frac{1}{d(d+i-1)(n-d-i+1)} \sum_{j=1}^{n-d-i+1} (d+j-1)(n-d-j+1)\omega^{n-d+1-i-j} \\ &= \frac{(d+i)(n-d-i)}{(d+i-1)(n-d-i+1)}\omega S_{i+1} + \frac{i(n-i)}{d(d+i-1)(n-d-i+1)} \end{aligned}$$

(where we set $S_{n-d+1} = 0$.) We now show by induction on $i$ that $S_i \leq 1-\epsilon$. Clearly, the bound holds for $S_{n-d+1}$ since $\epsilon \leq 1$. So now we have

$$\begin{aligned} S_i &= \frac{(d+i)(n-d-i)}{(d+i-1)(n-d-i+1)}\omega S_{i+1} + \frac{i(n-i)}{d(d+i-1)(n-d-i+1)} \\ &\leq \frac{(d+i)(n-d-i)}{(d+i-1)(n-d-i+1)}\omega(1-\epsilon) + \frac{i(n-i)}{d(d+i-1)(n-d-i+1)} \\ &= \frac{(d-1)(d+i)(n-d-i)}{d(d+i-1)(n-d-i+1)}(1-\epsilon) + \frac{i(n-i)}{d(d+i-1)(n-d-i+1)} \end{aligned}$$

by assumption. Showing that the RHS is less than $1-\epsilon$ is equivalent to showing that

$$(d-1)(d+i)(n-d-i)(1-\epsilon) + i(n-i) \leq d(d+i-1)(n-d-i+1)(1-\epsilon)$$

which is equivalent to

$$i(n-i) \leq \left[d(d+i-1)(n-d-i+1) - (d-1)(d+i)(n-d-i)\right](1-\epsilon).$$

It is straightforward (though a little tedious) to verify that

$$d(d+i-1)(n-d-i+1) - (d-1)(d+i)(n-d-i) = i(n-i) + d(d-1).$$

which yields the condition

$$i(n-i) \leq \left[i(n-i) + d(d-1)\right](1-\epsilon)$$

which again is equivalent to

$$\epsilon\left[i(n-i) + d(d-1)\right] \leq d(d-1)$$

and thus $\epsilon \leq \left(1 + \frac{i(n-i)}{d(d-1)}\right)^{-1}$. We note this quantity is minimized for $i = n/2$ and thus by definition of $\epsilon$, this condition holds. Since all our transformations were equivalences, this proves the bound on $S_i$. □

Readers familiar with Neumaier's work will recognize the calculations. It is easy to see that $\kappa(\mathbf{D}) = \frac{n^2}{4(n-1)}$, which is small enough so that the number of tours required for the algorithm is proportional to $1 + \frac{n^2}{4d(d-1)}$. This matches the bound obtained in [Neu17].

*Acknowledgment* This work was initiated in discussions with Léo Ducas, when the author was visiting the *Simons Institute for the Theory of Computation* during the program "Lattices: Algorithms, Complexity, and Cryptography". We thank Thomas Espitau for pointing out a bug in a proof in an earlier version of this manuscript.

## References

ABF+20. Martin R. Albrecht, Shi Bai, Pierre-Alain Fouque, Paul Kirchner, Damien Stehlé, and Weiqiang Wen. Faster enumeration-based lattice reduction: Root hermite factor $k^{1/(2k)}$ time $k^{k/8+o(k)}$. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 186–212. Springer, Heidelberg, August 2020.

ADH+19. Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W. Postlethwaite, and Marc Stevens. The general sieve kernel and new records in lattice reduction. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 717–746. Springer, Heidelberg, May 2019.

ALNS20. Divesh Aggarwal, Jianwei Li, Phong Q. Nguyen, and Noah Stephens-Davidowitz. Slide reduction, revisited - filling the gaps in SVP approximation. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 274–295. Springer, Heidelberg, August 2020.

AWHT16. Yoshinori Aono, Yuntao Wang, Takuya Hayashi, and Tsuyoshi Takagi. Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 789–819. Springer, Heidelberg, May 2016.

BLR08. Johannes A. Buchmann, Richard Lindner, and Markus Rückert. Explicit hard instances of the shortest vector problem. In Johannes Buchmann and Jintai Ding, editors, *Post-quantum cryptography, second international workshop, PQCRYPTO 2008*, pages 79–94. Springer, Heidelberg, October 2008.

CN11. Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2011.

DM13. Daniel Dadush and Daniele Micciancio. Algorithms for the densest sublattice problem. In Sanjeev Khanna, editor, *24th SODA*, pages 1103–1122. ACM-SIAM, January 2013.

dt16. The FPLLL development team. fplll, a lattice reduction library. Available at `https://github.com/fplll/fplll`, 2016.

GN08a. Nicolas Gama and Phong Q. Nguyen. Finding short lattice vectors within Mordell's inequality. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 207–216. ACM Press, May 2008.

GN08b. Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 31–51. Springer, Heidelberg, April 2008.

HPS11. Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In Phillip Rogaway, editor,

*CRYPTO 2011*, volume 6841 of *LNCS*, pages 447–464. Springer, Heidelberg, August 2011.

HS07. Guillaume Hanrot and Damien Stehlé. Improved analysis of kannan's shortest lattice vector algorithm. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 170–186. Springer, Heidelberg, August 2007.

LLL82. Arjen K. Lenstra, Hendrik W. Lenstra, Jr., and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:513–534, 1982.

LN14. Jianwei Li and Phong Nguyen. Approximating the densest sublattice from rankin's inequality. *LMS Journal of Computation and Mathematics [electronic only]*, 17, 08 2014.

LN20. Jianwei Li and Phong Q. Nguyen. A complete analysis of the bkz lattice reduction algorithm. Cryptology ePrint Archive, Report 2020/1237, 2020. `https://eprint.iacr.org/2020/1237`.

Lov86. Lásló Lovász. *An algorithmic theory of numbers, graphs and convexity*, volume 50 of *CBMS*. SIAM, 1986.

MW16. Daniele Micciancio and Michael Walter. Practical, predictable lattice basis reduction. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 820–849. Springer, Heidelberg, May 2016.

Neu17. Arnold Neumaier. Bounding basis reduction properties. *Des. Codes Cryptogr.*, 84(1-2):237–259, 2017.

PT09. G. Pataki and Mustafa Tural. Unifying lll inequalities. 2009.

Sch87. Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53(2–3):201–224, August 1987.

SE94. Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1-3):181–199, August 1994. Preliminary version in FCT 1991.

Wal15. Michael Walter. Lattice point enumeration on block reduced bases. In Anja Lehmann and Stefan Wolf, editors, *ICITS 15*, volume 9063 of *LNCS*, pages 269–282. Springer, Heidelberg, May 2015.