# Predicting BKZ Z-Shapes on q-ary Lattices

Martin R. Albrecht[1] and Jianwei Li[2⋆]

[1] Information Security Group, Royal Holloway, University of London.
[2] Inria and DIENS, PSL.

**Abstract.** Primal attacks against the Learning With Errors (LWE) problem rely on reducing $q$-ary lattices. These reduced bases have been observed to exhibit a so-called "Z-shape" on their Gram–Schmidt vectors. We propose an efficient simulator to accurately predict this Z-shape behaviour, which we back up with extensive simulations and experiments. We also formalise (under standard heuristics) the intuition that the presence of a Z-shape makes enumeration-based primal lattice attacks faster. Furthermore, we upgrade the LWE or lattice estimator with our simulator to assess and then rule out the impact of the $q$-ary Z-shape on solving LWE instances derived from parameter sets for NIST PQC candidates. We consider this improved estimator to be of independent interest.

## 1 Introduction

Lattice-based cryptography was first proposed by Ajtai and Dwork [AD97], was later expanded by the introduction of the NTRU public-key encryption scheme [HPS98], and is gaining importance partly due to the introduction of the Learning With Errors (LWE) problem by Regev [Reg05] and later its ring and module variants [SSTX09, LPR10, LS15]. Its attractive features include its apparent resistance to quantum attacks, high asymptotic efficiency and its "expressiveness" permitting to build e.g. fully-homomorphic encryption [BV11].

Constructions in both LWE(-like)-based and NTRU-based cryptography rely on *$q$-ary lattices* generated by *$q$-ary bases* of the form (under column-representation):

$$\begin{pmatrix} q\mathbf{I}_m & \mathbf{A} \\ \mathbf{0} & \mathbf{I}_n \end{pmatrix} \in \mathbb{Z}^{(m+n)\times(m+n)}.$$

To assess the difficulty of breaking such cryptographic schemes, two classes of lattice-reduction based attacks are routinely relied on in the literature: "dual (lattice) attacks" [MR09] and "primal (lattice) attacks" [ADPS16, PV21], the central object of this work. Such attacks consist of using lattice reduction to

⋆ This work was supported by EPSRC grants EP/S020330/1 and EP/S02087X/1, by the European Union Horizon 2020 Research and Innovation Program Grant 780701, and by funding from the European Research Council (ERC) under the European Union Horizon 2020 research and innovation programme (grant agreement No 885394). Part of this work was done while Li was at Royal Holloway.

solve certain hard lattice instances. Currently, the most commonly relied upon lattice reduction algorithm is BKZ [SE94], implemented in software libraries such as FP(y)LLL [FPL19, FPy20] and G6K [ADH+19, DSvW21].

BKZ takes a parameter – the block size $\beta$ – dominating both the running time and the output quality. In particular, the block size determines the shape of the Gram–Schmidt (GSO) vectors of BKZ-reduced bases.

Primal lattice attacks against LWE comprise two expensive steps, i.e. first BKZ-reducing an input $q$-ary basis for preprocessing and then, depending on the "flavour" of the algorithm, solving the *Shortest Vector Problem* (SVP) or the *Closest Vector Problem* (CVP) on the tail projected block of the BKZ-reduced basis. The success probability and running time of the second step depends on the GSO vectors of the BKZ-reduced basis. Such reduced $q$-ary lattices bases have been observed to exhibit a so-called "Z-shape" on their GSO vectors [How07, HPS+17, DKL+18, ACW19, AD21]. That is, the GSO shape has three pieces, as illustrated in Figure 1:

- The head GSO vectors have length $q$;
- The middle GSO vectors have roughly decreasing lengths;
- The tail GSO vectors have length 1, i.e. a flat shape.

However, we note that if the block size used by BKZ is large enough, the Z-shape disappears.

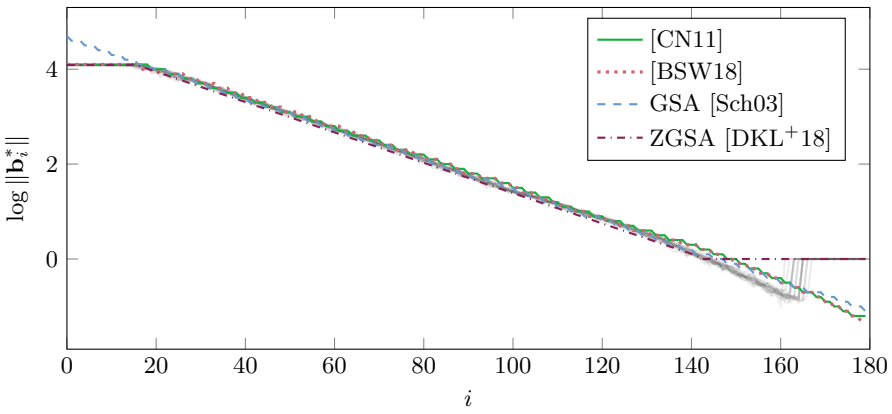


**Fig. 1.** Norms of Gram–Schmidt vectors (grey, blurry lines plot) after BKZ-65 reduction of sixteen 180 dimensional $q$-ary lattices with $q = 17$ and volume $q^{80}$ compared with the outputs of the simulators from [CN11, BSW18] as well as the GSA [Sch03] and ZGSA [DKL+18].

Cost estimates for primal lattice attacks require to predict the smallest BKZ block size sufficient to solve the targeted instances [ADPS16, AGVW17, DDGR20, PV21]. Here, the standard methodology relies on simulating the GSO shape of

BKZ-reduced bases. Several works have attempted to model the behaviour of BKZ on $q$-ary lattices [How07, HPS+17, Wun16, AD21], using the *Geometric Series Assumption* (GSA) [Sch03], BKZ simulators [CN11, BSW18] or bespoke models such as Z(T)GSA [Wun19, DKL+18, AD21]. However, known estimates and simulators model the actual behaviour of lattice reduction poorly as illustrated in Figure 1. Indeed, predicting the behaviour of lattice reduction on $q$-ary lattices was given as an open problem in [AD21, p. 11].

A related question to consider is whether the presence of a Z-shape makes primal lattice attacks faster. While, intuitively, this should be the case, the literature holds no formal answer, even for certain families of algorithms. Resolving these two challenges – predicting the shape after lattice reduction and quantifying the performance gain (if any) of the presence of a Z-shape – then would allow to answer the question of whether the "Z-shape" poses a threat to LWE or NTRU parameters currently considered for standardisation by NIST. Current cost estimates of primal lattice attacks mostly ignore the presence/impact of the Z-shape.

**Contributions.** We resolve the open question of "simulating or predicting the behaviour of BKZ on $q$-ary lattices" [AD21].

After some preliminaries in Section 2, we present an efficient simulator to model/predict the Z-shape behaviour of BKZ with blocksizes $\beta \geq 50$ on $q$-ary lattices in Section 3. Unlike previous works, our simulator matches experimental data from lattice reduction well. In particular, we validate that our simulation is fairly consistent with the FP(y)LLL and G6K implementations of BKZ-reducing $q$-ary bases.

Second, in Section 4, we formalise (under standard heuristics) the intuition that the presence of a Z-shape makes enumeration-based primal lattice attacks faster. As mentioned before, solving SVP (or CVP) on the tail projected block of a BKZ-reduced basis is one of the two expensive steps of the primal lattice attacks against LWE. We show that enumeration-based SVP solving on the tail projected block of a Z-shaped BKZ reduced basis is at least exponentially faster than on the tail block of an non Z-shape BKZ reduced basis. We note that our restriction to enumeration-based rather than sieving-based techniques is somewhat natural with current techniques, i.e. how to exploit the underlying shape of the basis is much less understood for the latter than the former.

Then, we upgraded the LWE estimator from [APS15] to support BKZ simulators in place of simply assuming the GSA. We assume this improvement has applications beyond this work. This allows us, in Section 5, to plug our own simulator into the framework to assess the impact of the $q$-ary Z-shape on solving LWE and NTRU instances derived from parameter sets for NIST PQC candidates. We observe that the Z-shape indeed "disappears" for block sizes sufficiently large to solve the targeted parameters. We can thus rule out this threat to existing lattice reduction cost estimates against PQC standardisation candidates.

Since our results critically depend on our simulations, we attach the source code of our simulator in Appendix A.

## 2 Background

**Notation.** We let matrix indices start with 0 and use column-representation for both vectors and matrices. Bold lower-case and upper-case letters denote column vectors and matrices respectively. The set of $m \times n$ matrices with coefficients in the ring $\mathbb{A}$ is denoted by $\mathbb{A}^{m \times n}$, and we identify $\mathbb{A}^m$ with $\mathbb{A}^{m \times 1}$. The notations $\log(\cdot)$ and $\ln(\cdot)$ stand for the base 2 and natural logarithms respectively.

### 2.1 Lattices

Lattices are discrete subgroups of $\mathbb{R}^m$. A lattice $\mathcal{L}$ in $\mathbb{R}^m$ is represented as a set of all integer linear combinations of $n$ linearly independent vectors $\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}$ in $\mathbb{R}^m$: $\mathcal{L} = \left\{ \sum_{i=0}^{n-1} x_i \cdot \mathbf{b}_i,\ x_i \in \mathbb{Z} \right\}$. The matrix $\mathbf{B} := (\mathbf{b}_0, \ldots, \mathbf{b}_{n-1})$ forms a *basis* of $\mathcal{L}$, and the integer $n$ is the *rank* of $\mathcal{L}$. We may write $\mathcal{L}(\mathbf{B})$ to indicate the lattice $\mathcal{L}$ generated by $\mathbf{B}$.

***q*-ary lattice.** Let $m, n$ and $\xi$ be positive integers, $q \geq 2$ be a prime modulus, and $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ be a matrix. Let $\mathbf{I}_n$ denote the $n \times n$ identity matrix. We call the following basis matrix

$$\mathbf{B} := \begin{pmatrix} q\mathbf{I}_m & \mathbf{A} \\ \mathbf{0} & \xi\mathbf{I}_n \end{pmatrix} \in \mathbb{Z}^{(m+n) \times (m+n)}$$

a (scaled) *q-ary basis*, which generates a (scaled) *q-ary lattice*.[3] This $q$-ary basis and hence its $q$-ary lattice are randomised if each entry of $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ is independently and uniformly sampled from $\mathbb{Z}_q$. For instance, the NTRU lattice $\begin{pmatrix} q\mathbf{I}_m & \mathbf{H} \\ \mathbf{0} & \mathbf{I}_n \end{pmatrix}$ (from [How07] where $\mathbf{H}$ is a circulant matrix) is also a $q$-ary lattice.

**Lattice problems.** Two of the most important hard lattice problems are the *Shortest Vector Problem* (SVP) and the *Closest Vector Problem* (CVP). Given a basis of a lattice $\mathcal{L}$ (endowed with the Euclidean norm), SVP is to find a shortest nonzero vector in $\mathcal{L}$, and CVP asks for a closest vector in $\mathcal{L}$ to a target vector $\mathbf{t}$.

The *unique Shortest Vector Problem* (uSVP) is a variant of SVP. Given a basis of a lattice $\mathcal{L}$ satisfying $\lambda_2(\mathcal{L}) \geq \gamma \cdot \lambda_1(\mathcal{L})$ for some $\gamma \geq 1$, $\mathrm{uSVP}_\gamma$ is to find the unique (up to sign) $\mathbf{v} \in \mathcal{L}$ of length $\lambda_1(\mathcal{L})$. Here, $\lambda_1(\mathcal{L}) = \min_{\mathbf{v} \in \mathcal{L} \setminus \{\mathbf{0}\}} \|\mathbf{v}\|$ is the *first minimum* of $\mathcal{L}$ and

$$\lambda_2(\mathcal{L}) = \min \{ \max\{\|\mathbf{u}\|, \|\mathbf{v}\|\} : \mathbf{u}, \mathbf{v} \in \mathcal{L} \text{ are linearly independent} \}$$

is the *second minimum* of $\mathcal{L}$.

[3] The presence of the scale $\xi$ may imply that $q\mathbb{Z}^{m+n} \not\subset \mathcal{L}(\mathbf{B})$ and thus that $\mathcal{L}(\mathbf{B})$ is not a $q$-ary lattice. We will abuse notation throughout and refer to $\mathcal{L}(\mathbf{B})$ as a $q$-ary lattice regardless.

The *Bounded Distance Decoding Problem* (BDD) is a variant of CVP. Given a basis of a lattice $\mathcal{L}$, a target vector $\mathbf{t}$, and a parameter $\alpha > 0$ such that $\mathrm{dist}(\mathbf{t}, \mathcal{L}) < \alpha \cdot \lambda_1(\mathcal{L})$, $\mathrm{BDD}_\alpha$ is to find the lattice vector $\mathbf{v} \in \mathcal{L}$ closest to $\mathbf{t}$.

BDD and uSVP are polynomial-time equivalent for small approximation factors [LM09]. Below, we will apply our results to the *Learning with Errors* (LWE) problem [Reg05]. It can be seen as an average-case version of BDD on $q$-ary lattices.

**Definition 1.** *Let $n$ be a positive integer, $q \geq 2$ be a prime modulus, $D_\sigma$ be a discrete Gaussian distribution over $\mathbb{Z}$ with standard deviation $\sigma$, and $\mathbf{s} \in \mathbb{Z}_q^n$ be a secret vector. We denote by $L_{\mathbf{s},\sigma}$ the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by sampling $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, sampling $\epsilon \in \mathbb{Z}_q$ from $D_\sigma$, and outputting $(\mathbf{a}, b) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + \epsilon) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ The (search-)LWE problem is to recover $\mathbf{s}$ from independent samples $(\mathbf{a}_i, b_i)$ drawn from $L_{\mathbf{s},\sigma}$.*

We numerically pick the minimal number $m$ of LWE samples for recovering the secret $\mathbf{s}$ when attacking concrete LWE-based schemes (e.g., using the LWE estimator [APS15, ACD+18]). It is standard to rewrite a collection of LWE samples into the matrix form $(\mathbf{A}, \mathbf{b})$ such that $\mathbf{b} \equiv \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod q$, where each entry of $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ is independently and uniformly sampled from $\mathbb{Z}_q$ and each entry of $\mathbf{e} \in \mathbb{Z}_q^m$ is sampled independently from $D_\sigma$.

**Hermite's constant.** The *Hermite invariant* of an $n$-rank lattice $\mathcal{L}$ is defined by $\gamma_n(\mathcal{L}) = \left(\lambda_1(\mathcal{L})/\mathrm{vol}(\mathcal{L})^{1/n}\right)^2$. Hermite's constant is the maximum $\gamma_n = \max \gamma_n(\mathcal{L})$ over all $n$-rank lattices $\mathcal{L}$. The best asymptotical bounds known are [CS87, MH73]: $\frac{n}{2\pi e} + \frac{\log(\pi n)}{2\pi e} \leq \gamma_n \leq \frac{1.744n}{2\pi e} + o(n)$.

**Primitive vector.** A non-zero vector $\mathbf{b}$ in a lattice $\mathcal{L}$ is *primitive* for $\mathcal{L}$ iff it can be extended to a basis of $\mathcal{L}$.

**Gram–Schmidt orthogonalisation.** Let $\mathbf{B} = (\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}) \in \mathbb{R}^{m \times n}$ be a basis of a lattice $\mathcal{L}$. Lattice algorithms often involve the orthogonal projections $\pi_i : \mathbb{R}^m \mapsto \mathrm{span}\,(\mathbf{b}_0, \ldots, \mathbf{b}_{i-1})^\perp$ for $i = 0, \ldots, n-1$. The *Gram–Schmidt orthogonalisation* (GSO) of $\mathbf{B}$ is $\mathbf{B}^* = (\mathbf{b}_0^*, \ldots, \mathbf{b}_{n-1}^*)$, where the Gram–Schmidt vector $\mathbf{b}_i^*$ is $\pi_i(\mathbf{b}_i)$. Then $\mathbf{b}_0^* = \mathbf{b}_0$ and $\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=0}^{i-1} \mu_{i,j} \cdot \mathbf{b}_j^*$ for $i = 1, \ldots, n-1$, where $\mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle}$. The projected block $(\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1}), \ldots, \pi_i(\mathbf{b}_{j-1}))$ is denoted by $\mathbf{B}_{[i,j)}$. Then the volume of the parallelepiped generated by $\mathbf{B}_{[i,j)}$ is $\mathrm{vol}(\mathbf{B}_{[i,j)}) = \prod_{k=i}^{j-1} \|\mathbf{b}_k^*\|$. In particular, $\mathbf{B}_{[0,j)} = (\mathbf{b}_0, \ldots, \mathbf{b}_{j-1})$ and $\mathrm{vol}(\mathcal{L}) = \mathrm{vol}(\mathbf{B}) = \prod_{k=0}^{n-1} \|\mathbf{b}_k^*\|$.

**Lattice reduction.** Let $\mathbf{B} = (\mathbf{b}_0, \ldots, \mathbf{b}_{n-1})$ be a basis of a lattice $\mathcal{L}$.

The basis $\mathbf{B}$ is *size-reduced* if $|\mu_{i,j}| \leq \frac{1}{2}$ for all $0 \leq j < i < n$: then $\max_i \|\mathbf{b}_i\| \leq \sqrt{n} \cdot \max_i \|\mathbf{b}_i^*\|$. Transforming a basis into this form without modifying $\mathcal{L}(\mathbf{B})$ or $\mathbf{B}^*$ is called *size reduction*, and this can be done easily and efficiently.

The basis $\mathbf{B}$ is *LLL-reduced* [LLL82] if it is size-reduced and every 2-rank projected block $\mathbf{B}_{[i,i+2)}$ satisfies Lovász's condition: $\frac{3}{4} \cdot \|\mathbf{b}_i^*\|^2 \leq \|\mu_{i+1,i} \cdot \mathbf{b}_i^* + \mathbf{b}_{i+1}^*\|^2$ for $0 \leq i \leq n-2$. The parameter $\frac{3}{4}$ can be replaced with any constant in the interval $(\frac{1}{4}, 1)$. The basis $\mathbf{B}$ is *SVP-reduced* if $\|\mathbf{b}_0\| = \lambda_1(\mathcal{L})$. There are two relaxations with $\delta \geq 1$: $\mathbf{B}$ is *$\delta$-SVP-reduced* if $\|\mathbf{b}_0\| \leq \delta \cdot \lambda_1(\mathcal{L})$; $\mathbf{B}$ is *$\delta$-HSVP-reduced* if $\|\mathbf{b}_0\| \leq \delta \cdot \mathrm{vol}(\mathcal{L})^{1/n}$. In particular, if a projected block $\mathbf{B}_{[i,j)}$ is *$\delta$-HSVP-reduced*, then $\|\mathbf{b}_i^*\| \leq \delta \cdot \mathrm{vol}(\mathbf{B}_{[i,j)})^{1/(j-i)}$. The basis $\mathbf{B}$ is *HKZ-reduced* if it is size-reduced and $\mathbf{B}_{[i,n)}$ is SVP-reduced for $i = 0, \ldots, n-1$; the basis $\mathbf{B}$ is *BKZ-$\beta$ reduced* (or *BKZ-reduced* with block size $\beta$) [Sch87] if it is size-reduced and $\mathbf{B}_{[i,\min\{i+\beta,n\})}$ is SVP-reduced for $i = 0, \ldots, n-1$.

**Gaussian heuristic.** If $\mathcal{L}$ is an $n$-rank random lattice (cf. e.g. [CN11, §2] for the definition), the classical *Gaussian heuristic* claims $\lambda_1(\mathcal{L}) \approx \mathrm{GH}(n) \cdot \mathrm{vol}(\mathcal{L})^{1/n}$ with high probability [Rog56]. Here, $\mathrm{GH}(n)$ denotes the radius of the unit-volume $n$-dimensional ball:

$$\mathrm{GH}(n) = \frac{\Gamma(n/2+1)^{1/n}}{\sqrt{\pi}} \approx \sqrt{\frac{n}{2\pi e}} \cdot (\pi n)^{\frac{1}{2n}}.$$

It is known that $\mathrm{GH}(n) < \sqrt{\gamma_n} \leq \sqrt{2} \cdot \mathrm{GH}(n)$ if $n > 24$ [Bli14]. Thus, for any lattice $\Lambda$ of rank $n > 24$, we have $\lambda_1(\Lambda) \leq \sqrt{2} \cdot \mathrm{GH}(n) \cdot \mathrm{vol}(\Lambda)^{1/n}$.

There is the following so-called *weak Gaussian heuristic* [LN20, §4.1]: For any $\epsilon \in (0, \sqrt{2}-1]$, there exists $N > 0$ such that for "most" random lattices $\mathcal{L}$ of rank $n \geq N$, we have: $\lambda_1(\mathcal{L}) \leq (1+\epsilon)\,\mathrm{GH}(n) \cdot \mathrm{vol}(\mathcal{L})^{1/n}$.

**GSA.** Let $\mathbf{B} = (\mathbf{b}_0, \ldots, \mathbf{b}_{n-1})$ be a basis. Schnorr's *Geometric Series Assumption* (GSA) [Sch03] says that $\mathbf{B}$ follows the GSA wrt some constant $r \in (1, \frac{4}{3}]$ (depending on the reduction algorithm) if its Gram–Schmidt lengths decay geometrically wrt $r$, namely $\|\mathbf{b}_i^*\|/\|\mathbf{b}_{i+1}^*\| = r$ for all $i = 0, \ldots, n-2$. In practice, it has been observed that a reduced basis produced by the LLL algorithm [LLL82] satisfies the GSA in an approximate sense when the input basis is sufficiently randomised.

Suppose that $\mathbf{B}$ is BKZ-$\beta$ reduced. Chen [Che13] gives the following length estimate for the first basis vector:

$$\|\mathbf{b}_0\| \approx \mathrm{GH}(\beta)^{\frac{n-1}{\beta-1}} \cdot \mathrm{vol}(\mathbf{B})^{1/n} \quad \text{for} \quad n > \beta > 50.$$

If $\mathbf{B}$ further satisfies the GSA wrt constant $r$, then the identity $\mathrm{vol}(\mathbf{B}) = \prod_{i=0}^{n-1} \|\mathbf{b}_i^*\|$ implies $r \approx \mathrm{GH}(\beta)^{\frac{2}{\beta-1}}$ and hence

$$\|\mathbf{b}_i^*\| \approx \mathrm{GH}(\beta)^{\frac{n-2i-1}{\beta-1}} \cdot \mathrm{vol}(\mathbf{B})^{1/n} \quad \text{for} \quad i = 0, \ldots, n-1. \tag{1}$$

### 2.2 Enumeration with cylinder pruning

Enumeration [Mic81, Kan83, FP85, SE94, MW15, ABF+20, ABLR21] is the simplest algorithm for solving SVP and requires only polynomial memory: given

a full-rank lattice $\mathcal{L}$ in $\mathbb{R}^n$ and a radius $R > 0$, enumeration outputs $\mathcal{L} \bigcap \mathrm{Ball}_n(R)$ by a depth-first tree search. By comparing all the norms of vectors in $\mathcal{L} \bigcap \mathrm{Ball}_n(R)$, one can find a shortest nonzero lattice vector if $R \geq \lambda_1(\mathcal{L})$.

$(f, R)$-*Cylinder pruning* [SH95, GNR10] speeds up enumeration by replacing the search region $\mathrm{Ball}_n(R)$ with a (much smaller) subset $\mathrm{P}_f(\mathbf{B}, R)$ defined by a bounding function $f : \{1, \ldots, n\} \to [0, 1]$, a basis $\mathbf{B}$ of $\mathcal{L}$ and $R$:

$$\mathrm{P}_f(\mathbf{B}, R) = \{\mathbf{x} \in \mathbb{R}^n : \|\pi_{n-i}(\mathbf{x})\| \leq f(i) \cdot R \text{ for all } 1 \leq i \leq n\} \subseteq \mathrm{Ball}_n(R).$$

The set $\mathrm{P}_f(\mathbf{B}, R)$ is an intersection of cylinders, since each equation $\|\pi_{n-i}(\mathbf{x}/R)\| \leq f(i)$ defines an $i$-dimensional cylinder

$$C_{f,i} = \left\{ (x_1, \ldots, x_i) \in \mathbb{R}^i : \forall\ s \leq i, \sum_{\ell=1}^{s} x_\ell^2 \leq f(s)^2 \right\} \subseteq \mathrm{Ball}_i(1) \ \text{ for } \ i = 1, \ldots, n.$$

The adaptation of (cylinder pruning) SVP enumeration to CVP is straightforward [LN13, ANS18]. Given a full-rank lattice $\mathcal{L}$ in $\mathbb{R}^n$ (specified by a basis $\mathbf{B}$) and a target vector $\mathbf{t} \in \mathbb{R}^n$, enumeration solves the CVP instance $(\mathbf{B}, \mathbf{t})$ with radius $R > 0$ by outputting $\mathcal{L} \bigcap \mathrm{Ball}_n(\mathbf{t}, R)$. $(f, R)$-Cylinder pruning solves this CVP instance by replacing the search region $\mathrm{Ball}_n(\mathbf{t}, R)$ with a subset

$$\mathrm{P}_f(\mathbf{B}, \mathbf{t}, R) = \{\mathbf{x} \in \mathbb{R}^n : \|\pi_{n-i}(\mathbf{x} - \mathbf{t})\| \leq f(i) \cdot R \text{ for all } 1 \leq i \leq n\} \subseteq \mathrm{Ball}_n(\mathbf{t}, R).$$

### 2.3 The BKZ algorithm

The (original) BKZ algorithm introduced by Schnorr and Euchner [SE94] is the most widely used lattice reduction algorithm besides LLL [LLL82] and a central tool in lattice-based cryptanalysis. Its performance drives the setting of concrete parameters (such as keysizes) for concrete lattice-based cryptographic primitives (see e.g. [ACD+18]).

At a high level, BKZ reduces a basis in high rank, using (H)SVP-oracles in low rank ($\leq \beta$) as subroutines and running the LLL algorithm [LLL82] to remove the linear dependency right after inserting a lattice vector (found by the oracle) in the current basis.

Originally, the SVP subroutine implemented in [SE94] was the simplest form of lattice enumeration, but it is now replaced by better subroutines, such as (relaxed) pruned enumeration in rank $\kappa$ with extended preprocessing in "block size" $\lceil (1 + c) \cdot \kappa \rceil$ in [ABF+20, ABLR21]'s BKZ variants for some small constant $c \geq 0$ and (asymptotically and practically) faster sieving (e.g. using the General Sieve Kernel [ADH+19, DSvW21]).

In practice, BKZ is typically implemented with an approximate (rather than exact) SVP-subroutine. Thus, Algorithm 1 slightly generalises BKZ by allowing the use of a relaxed HSVP-oracle at Step 3, as well as full LLL (instead of partial LLL) at Step 5: both are justified by Li–Nguyen's analysis [LN20].

Since the $i$-th BKZ iteration retrieves the sublattice $\mathcal{L}(\mathbf{b}_i, \ldots, \mathbf{b}_{\min\{i+\beta, n\}-1})$ whose rank varies over index $i$, we for convenience let $\beta_i$ and $n_i$ denote respectively

the rank and the end index of such a sublattice wrt index $i$ (as done in [LN20]):

$$\beta_i = \min\{\beta, n-i\} \quad \text{and} \quad n_i = \min\{i+\beta, n\} \quad \text{for} \quad i = 0, \ldots, n-1. \tag{2}$$

Notice that $\beta_i = \beta$ and $n_i = i + \beta$ for $i = 0, \ldots, n-\beta$.

**Algorithm 1** BKZ: Schnorr–Euchner's BKZ algorithm [SE94]

**Input:** A block size $\beta \in (2, n)$, the number of tours $N \in \mathbb{Z}^+$, a relaxation factor $\alpha \geq 1$, and an LLL-reduced basis $\mathbf{B} = (\mathbf{b}_0, \ldots, \mathbf{b}_{n-1})$ of a lattice $\mathcal{L} \subseteq \mathbb{Z}^m$.
**Output:** A new basis of $\mathcal{L}$.

1: **for** $j = 0$ to $N-1$ **do**
2:  **for** $i = 0$ to $n-2$ **do**
3:  Find a primitive vector $\mathbf{b}$ for the sublattice generated by the basis vectors $\mathbf{b}_i, \ldots, \mathbf{b}_{n_i-1}$ such that $\|\pi_i(\mathbf{b})\| \leq \alpha\sqrt{\gamma_{\beta_i}} \cdot \mathrm{vol}(\mathbf{B}_{[i,n_i)})^{1/\beta_i}$
4:  **if** $\|\mathbf{b}_i^*\| > \|\pi_i(\mathbf{b})\|$ **then**
5:  LLL-reduce $(\mathbf{b}_0, \ldots, \mathbf{b}_{i-1}, \mathbf{b}, \mathbf{b}_i, \ldots, \mathbf{b}_{n-1})$ to remove linear dependencies
6:  **end if** //The $i$-th iteration refers to a single execution of Steps 3-6.
7:  **end for** //A BKZ tour refers to a single execution of Steps 2-7.
8: **end for**
9: **return B**.

Building on Hanrot–Pujol–Stehlé's analysis of a certain BKZ variant (HKZ-reducing every projected block $\mathbf{B}_{[i,n_i)}$ and removing internal LLL calls) [HPS11], Li and Nguyen [LN20] justified the popular "early termination" strategy in practice of BKZ:

**Theorem 1 ([LN20, Theorem 2]).** *Let $n > \beta \geq 2$ be integers and let $0 < \varepsilon \leq 1 \leq \alpha \leq \frac{2^{(\beta-1)/4}}{\sqrt{\gamma_\beta}}$. Given as input a block size $\beta$, a relaxation factor $\alpha$, and an LLL-reduced basis of an $n$-rank lattice $\mathcal{L} \subset \mathbb{R}^m$, if $N \geq 4(\ln 2)\frac{n^2}{\beta^2}\log\frac{n^{1.5}}{(4\sqrt{3})\varepsilon}$, then Algorithm 1 outputs a basis $(\mathbf{b}_0, \ldots, \mathbf{b}_{n-1})$ of $\mathcal{L}$ such that*

$$\|\mathbf{b}_0\| \leq (1+\varepsilon) \cdot (\alpha^2\gamma_\beta)^{\frac{n-1}{2(\beta-1)} + \frac{\beta\cdot(\beta-2)}{2n\cdot(\beta-1)}} \cdot \mathrm{vol}(\mathcal{L})^{1/n}.$$

It was also mentioned in [LN20] that for $n > \beta > 8e\pi$, there is a BKZ-$\beta$ reduced basis $\mathbf{B} = (\mathbf{b}_0, \ldots, \mathbf{b}_{n-1})$ satisfying $\|\mathbf{b}_0\| = \left(\frac{\beta-1}{8e\pi}\right)^{\frac{n-1}{2\beta}} \cdot \mathrm{vol}(\mathbf{B})^{1/n}$. Since $\gamma_\beta = \Theta(\beta)$, this means that BKZ with early termination indeed provides bases almost as reduced as the full BKZ algorithm.

### 2.4 The Chen-Nguyen BKZ simulator

Chen and Nguyen [CN11] first proposed an efficient simulator (see Algorithm 2) to model/predict the practical behavior of BKZ with high block size $\beta \geq 50$. It is based on two ideas:

– It models the evolution of the Gram–Schmidt norms during the execution of BKZ, without running costly BKZ on the basis itself;
– It predicts the first minimum for each local block with the Gaussian heuristic estimate and with an experiment-driven modification for the tail blocks of the basis.

Just like real BKZ implementations, the simulator runs a sequence of tours (cf. the note at Step 16 of Algorithm 2):

– At the beginning of each tour, if there is an index $i \in [0, n-46]$ s.t. $\|\mathbf{b}_i^*\| > \mathrm{GH}(\beta_i) \cdot \mathrm{vol}(\mathbf{B}_{[i,n_i)})^{1/\beta_i}$ (say, $\underline{i}$ is such a smallest index), then this tour updates $\|\mathbf{b}_s^*\|$ into $\mathrm{GH}(\beta_s) \cdot \mathrm{vol}(\mathbf{B}_{[s,n_s)})^{1/\beta_s}$ in turn over indices $s = \underline{i}, \ldots, n-46$ (cf. Steps 7-18); otherwise, it keeps the head block $\mathbf{B}_{[0,n-45)}$ unchanged.
– At the end of each tour, the tail block of rank 45 is updated unconditionally with the average shape of HKZ-reduced bases of random 45-rank lattices of unit-volume (cf. Steps 20-22).

This simulator fits well with experiments using random lattices and Darmstadt's lattice challenges. It was refined into the Bai–Stehlé–Wen simulator [BSW18] by mainly modifying the Gaussian heuristic with probabilistic nature. Compared to [CN11], the simulator [BSW18] provides more accurate predictions of the head region (i.e. the $\|\mathbf{b}_i^*\|$'s for small indices $i$) and is similarly accurate for both body and tail regions. We remark that both simulators [CN11, BSW18] have no essential differences for modelling the practical behaviour of BKZ-reducing $q$-ary bases, as shown in Figure 1.

### 2.5 Primal lattice attacks against LWE

The *primal lattice attacks* against LWE include the "uSVP attack", the "BDD attack" and the "hybrid attack". These attacks proceed by first reducing LWE to BDD and then (optionally) reducing BDD to uSVP, and eventually solving uSVP or BDD using BKZ.

**LWE to BDD.** Given a matrix LWE instance $(\mathbf{A}, \mathbf{b})$ such that $\mathbf{b} \equiv \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod q$ (cf. Definition 1 and its subsequent statement), we have $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} + q \cdot \mathbf{c}$ for some $\mathbf{c} \in \mathbb{Z}^m$. This can be rewritten into the following form:

$$\underline{\mathbf{B}} \cdot \begin{pmatrix} \mathbf{c} \\ \mathbf{s} \end{pmatrix} - \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} -\mathbf{e} \\ \mathbf{s} \end{pmatrix},$$

where $\underline{\mathbf{B}} = \begin{pmatrix} q\mathbf{I}_m & \mathbf{A} \\ \mathbf{0} & \mathbf{I}_n \end{pmatrix} \in \mathbb{Z}^{(m+n)\times(m+n)}$ is a $q$-ary basis.

Since each entry of $\mathbf{e} \in \mathbb{Z}_q^m$ is sampled independently from a discrete Gaussian distribution $D_\sigma$, we expect $\|\mathbf{e}\| \approx \sqrt{m}\sigma$. Here, $\sigma$ is usually a small constant wrt $q$ and $n$ [AGVW17, ACD+18]. Assume that the secret $\mathbf{s} \in \mathbb{Z}_q^n$ has a small norm. This means that the vector $\left(-\mathbf{e}^T, \mathbf{s}^T\right)^T$ might have a small norm wrt the $q$-ary

**Algorithm 2** The Chen-Nguyen BKZ simulator [CN11, Alg. 2]

**Input:** The Gram-Schmidt log-norms $\{\ell_i = \log \|\mathbf{b}_i^*\|\}_{0 \le i < n}$ and a block size $\beta \in [46, n]$.
**Output:** A prediction for the Gram-Schmidt log-norms $\{\ell'_i = \log \|\mathbf{b}_i^*\|\}_{0 \le i < n}$ after running $N$ tours of BKZ-$\beta$ reduction.

1: **for** $i = 1$ to $45$ **do**
2: $r_i \leftarrow$ average $\log \|\mathbf{h}_i^*\|$ of an HKZ-reduced basis $\mathbf{H}$ of a random 45-rank lattice of unit-volume
3: **end for**
4: **for** $i = 46$ to $\beta$ **do** $c_i \leftarrow \log \mathrm{GH}(i) = \log \frac{\Gamma(i/2+1)^{1/i}}{\sqrt{\pi}}$ **end for**
5: **for** $j = 0$ to $N - 1$ **do**
6: $\phi \leftarrow \mathsf{true}$ //flag to store whether $\mathcal{L}(\mathbf{B}_{[i,n)})$ has changed.
7: **for** $i = 0$ to $n - 46$ **do**
8: $g \leftarrow c_{\beta_i} + (\sum_{s=0}^{n_i-1} \ell_s - \sum_{s=0}^{i-1} \ell'_s)/\beta_i$ //$\beta_i$ and $n_i$ are defined as in Eq. (2).
9: **if** $\phi = \mathsf{true}$ **then**
10: **if** $g < \ell_i$ **then**
11: $\ell'_i \leftarrow g$; $\phi \leftarrow \mathsf{false}$
12: **else**
13: $\ell'_i \leftarrow \ell_i$ //[CN11, Alg. 2] should include this step.
14: **end if**
15: **else**
16: $\ell'_i \leftarrow g$ //A tour refers to a single execution of Steps 6-23.
17: **end if**
18: **end for**
19: $\log V \leftarrow \sum_{s=0}^{n-1} \ell_s - \sum_{s=0}^{n-46} \ell'_s$
20: **for** $i = n - 45$ to $n - 1$ **do**
21: $\ell'_i \leftarrow \frac{\log V}{45} + r_{i+45-n}$
22: **end for**
23: $\{\ell_0, \ldots, \ell_{n-1}\} \leftarrow \{\ell'_0, \ldots, \ell'_{n-1}\}$
24: **end for**
25: **return** $\{\ell_0, \ldots, \ell_{n-1}\}$.

lattice $\mathcal{L}(\underline{\mathbf{B}})$. As a consequence, we can transform the matrix LWE instance $(\mathbf{A}, \mathbf{b})$ into a BDD instance with lattice basis $\underline{\mathbf{B}}$ and target vector $\mathbf{t} = \left(\mathbf{b}^T, \mathbf{0}^T\right)^T$: if a BDD solver finds the lattice vector $\underline{\mathbf{B}} \cdot \left(\mathbf{c}^T, \mathbf{s}^T\right)^T$ close to $\mathbf{t}$, then it allows us to recover the secret $\mathbf{s}$. This is also called the *BDD attack* against LWE.

**BDD to uSVP.** We can reduce the above BDD instance to an uSVP instance with lattice basis of rank $d := m + n + 1$

$$\overline{\mathbf{B}} = \begin{pmatrix} \underline{\mathbf{B}} & \mathbf{t} \\ \mathbf{0} & \tau \end{pmatrix} = \begin{pmatrix} q\mathbf{I}_m & \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{I}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \tau \end{pmatrix} \in \mathbb{Z}^{d \times d}$$

using Kannan's embedding technique [Kan83] with embedding factor $\tau \in \mathbb{Z}^+$.[4] The lattice $\mathcal{L}(\overline{\mathbf{B}})$ has volume $\tau \cdot q^m$ and contains a vector

$$\overline{\mathbf{B}} \cdot \begin{pmatrix} \mathbf{c} \\ \mathbf{s} \\ -1 \end{pmatrix} = \begin{pmatrix} -\mathbf{e} \\ \mathbf{s} \\ -\tau \end{pmatrix}$$

of norm $\sqrt{\|\mathbf{e}\|^2 + \|\mathbf{s}\|^2 + \tau^2}$. Assume that this lattice vector is unusually short, i.e. the gap between the first and second minimums $\lambda_2(\mathcal{L}(\overline{\mathbf{B}}))/\lambda_1(\mathcal{L}(\overline{\mathbf{B}}))$ is large, we can find this unique shortest vector using BKZ.

If the secret $\mathbf{s}$ is small and/or sparse, the above short lattice vector is unbalanced: $\frac{\|\mathbf{s}\|}{\sqrt{n}} \ll \frac{\|\mathbf{e}\|}{\sqrt{m}} \approx \sigma$. The Bai-Galbraith embedding variant [BG14] uses a scale $\xi \in \mathbb{Z}^+$ to balance the size of the secret $\mathbf{s}$ and the noise $\mathbf{e}$ by constructing the following embedding lattice basis

$$\mathbf{B} = \begin{pmatrix} q\mathbf{I}_m & \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \xi\mathbf{I}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \xi \end{pmatrix} \in \mathbb{Z}^{d \times d}.$$

Here, we simply set the embedding factor $\tau = \xi$. The $q$-ary lattice $\mathcal{L}(\mathbf{B})$ has volume $\xi^{n+1} \cdot q^m$ and contains a vector

$$\mathbf{v} := \mathbf{B} \cdot \begin{pmatrix} \mathbf{c} \\ \mathbf{s} \\ -1 \end{pmatrix} = \begin{pmatrix} -\mathbf{e} \\ \xi \cdot \mathbf{s} \\ -\xi \end{pmatrix}. \tag{3}$$

We may want to choose an appropriate value of $\xi$ such that

$$\|\mathbf{v}\| = \sqrt{\|\mathbf{e}\|^2 + \|\xi \cdot \mathbf{s}\|^2 + \xi^2} \approx \sigma\sqrt{m + n + 1} = \sigma\sqrt{d}.$$

**Hybrid attack.** Howgrave-Graham's hybrid attack combines Odlyzko's meet-in-the-middle attack (described in [HPS98, HGSW03]) with lattice reduction. We refer the reader to [How07, Wun19] for more details. Here, we may think of it as the "BDD attack" composed with a combinatorial technique to guess components of the secret.

**Success condition.** The running time of BKZ solving uSVP (or BDD) is dominated by the block size $\beta$. As a consequence, the cost estimate of the uSVP attack requires to predict the smallest block size sufficient to recover the unique shortest vector $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ defined in Equation (3). To do so, the estimate presented in [ADPS16, § 6.3] and verified in [AGVW17, PV21] uses the following three assumptions:

1. A BKZ-$\beta$ reduced basis $\widehat{\mathbf{B}} = (\mathbf{b}_0, \ldots, \mathbf{b}_{n-1})$ of $\mathcal{L}(\mathbf{B})$ satisfies the GSA, i.e. Equation (1) holds;

[4] Lattice reduction software often requires input bases to be integral.

2. The unique shortest vector $\mathbf{v}$ will be detected if its projection $\pi_{d-\beta}(\mathbf{v})$ (orthogonal to the first $d-\beta$ basis vectors $\widehat{\mathbf{B}}_{[0,d-\beta)}$) is shorter than $\mathbf{b}^*_{d-\beta}$;
3. The projection $\pi_{d-\beta}(\mathbf{v})$ is expected to have norm $\sqrt{\frac{\beta}{d}} \cdot \|\mathbf{v}\| \approx \sigma \cdot \sqrt{\beta}$.

This results in the following *success condition* on the block size $\beta$ for BKZ to solve uSVP:

$$\sigma \cdot \sqrt{\beta} \approx \|\pi_{d-\beta}(\mathbf{v})\| \leq \|\mathbf{b}^*_{d-\beta}\| \approx \mathrm{GH}(\beta)^{\frac{2\beta-d-1}{\beta-1}} \cdot \mathrm{vol}(\mathbf{B})^{1/d}. \tag{4}$$

Once finding the projection $\pm\pi_{d-\beta}(\mathbf{v}) = \pm\sum_{i=d-\beta}^{n-1} x_i \pi_{d-\beta}(\mathbf{b}_i)$ for some $x_i \in \mathbb{Z}$, we then solve a CVP instance with lattice basis $\widehat{\mathbf{B}}_{[0,d-\beta)}$ and target vector $\sum_{i=d-\beta}^{n-1} x_i \mathbf{b}_i$ because the distance

$$\mathrm{dist}(\sum_{i=d-\beta}^{n-1} x_i \mathbf{b}_i, \mathcal{L}(\widehat{\mathbf{B}}_{[0,d-\beta)})) \leq \|(\sum_{i=d-\beta}^{n-1} x_i \mathbf{b}_i - \mathbf{v}) - \sum_{i=d-\beta}^{n-1} x_i \mathbf{b}_i\| = \|\mathbf{v}\|$$

is unusually small. In particular, running Babai's nearest plane algorithm [Bab86] typically suffices [AGVW17]. This cost estimate is available under the name `primal_usvp` in the LWE estimator [APS15].

Since BKZ-$\beta$ will be more expensive than SVP in dimension $\beta$ by a polynomial factor, we may follow BKZ-$\beta$ reduction by an SVP call in dimension $\eta$ on the block $\hat{\mathbf{B}}[d-\eta, d)$ (for finding $\pi_{d-\eta}(\mathbf{v})$ instead of $\pi_{d-\beta}(\mathbf{v})$) . We may think of this as preprocessing the basis with BKZ-$\beta$ followed by a call to SVP (with Kannan's embedding) on $\hat{\mathbf{B}}[d-\eta, d)$ (or CVP (without Kannan's embedding) on a corresponding tail block) to solve the BDD instance. This tail SVP call and this tail CVP call are roughly the same in terms of cost. We may choose some integer $\eta > \beta$ to balance the cost of BKZ-$\beta$ preprocessing the basis and solving SVP on $\hat{\mathbf{B}}[d-\eta, d)$. This strategy is called `primal_bdd` in the LWE estimator [APS15].

Finally, it is not necessary to assume the Geometric Series Assumption. That is, we may replace the shape predicted by the first assumption above with the shape predicted by a simulator such as those from [CN11, BSW18]. We can then replace Equation (4) with a comparison of the appropriate length output by the simulator. Thus, the presence of a BKZ simulator predicting the Z-shape permits to cost primal lattice attacks in the presence of such Z-shapes.

## 3 A Z-shape BKZ simulator

In this section, we present an efficient simulator to model/predict the Z-shape behaviour of BKZ with block size $\beta \geq 50$ on $q$-ary lattices with sufficiently large $q/\xi$ (say, $q/\xi \geq 37/4$ which covers cryptographic parameter ranges). Our simulation is reasonably consistent with FP(y)LLL's and G6K's actual implementations of BKZ-reducing $q$-ary bases. Thus, we consider our simulator as a tool to achieve more precise security estimates for lattice-based cryptography relying on $q$-ary lattices.

### 3.1 Prototypes

Here, we present high-level pseudocode for both Chen-Nguyen's and our BKZ simulator and explain the key differences. We refer to this high-level presentation as a "prototype", which can be viewed as a (worst-case) variant of the BKZ algorithm [SE94] and semantically corresponds to a (heuristic) BKZ simulator. First, we give a prototype for the Chen-Nguyen BKZ simulator [CN11] in Algorithm 3 and then for our BKZ simulator in Algorithm 4.

Both prototypes use the same parameters (such as the same block size $\beta$), the same notation $\beta_i$ and $n_i$ (defined in Equation (2)), and a generalisation $\widetilde{\gamma}_\kappa(\Lambda)$ of the Hermite invariant $\gamma_\kappa(\Lambda)$ of a $\kappa$-rank lattice $\Lambda$ such that there exists a primitive vector $\mathbf{v}$ for $\Lambda$ satisfying

$$\|\mathbf{v}\| = \sqrt{\widetilde{\gamma}_\kappa(\Lambda)} \cdot \mathrm{vol}(\Lambda)^{1/\kappa}.$$

This nonzero vector $\mathbf{v}$ is not necessarily the shortest in $\Lambda$ and the primitive constraint ensures that $\mathbf{v}$ can be extended into a basis of $\Lambda$ whose first basis vector is $\mathbf{v}$. We simply write $\widetilde{\gamma}(\mathbf{A}) = \widetilde{\gamma}_\kappa(\Lambda)$ if $\mathbf{A}$ is a basis of $\Lambda$ and its rank is clear from context.

When letting $\widetilde{\gamma}_\kappa(\cdot) = \gamma_\kappa(\cdot)$ then it is not hard to see that both Algorithms 3 and 4 (with size-reduction) output BKZ-$\beta$ reduced bases if they run for sufficiently many tours. We take this as validation of the correctness of both algorithms/simulators.

However, conceptually, we cannot simply replace $\widetilde{\gamma}_\kappa(\cdot)$ with $\gamma_\kappa(\cdot)$ in either Algorithm 3 or Algorithm 4. When doing so, both algorithms would in general only consider two cases, namely that $\|\mathbf{b}_i^*\|$ is greater than or equal to

$$\sqrt{\gamma_{\beta_i}(\mathcal{L}(\mathbf{B}_{[i,n_i)}))} \cdot \mathrm{vol}(\mathbf{B}_{[i,n_i)})^{1/\beta_i} \left(= \lambda_1(\mathcal{L}(\mathbf{B}_{[i,n_i)}))\right)$$

for any $i$, and then modify the GSO $\mathbf{B}^*$ in the same way (i.e. update $\|\mathbf{b}_i^*\|$ to $\lambda_1(\mathcal{L}(\mathbf{B}_{[i,n_i)}))$). In other words, with $\widetilde{\gamma}_\kappa(\cdot)$, we allow the algorithm to consider the following three cases for each local block $\mathbf{B}_{[i,n_i)}$:

- Case 1: $\|\mathbf{b}_i^*\| > \sqrt{\widetilde{\gamma}(\mathbf{B}_{[i,n_i)})} \cdot \mathrm{vol}(\mathbf{B}_{[i,n_i)})^{1/\beta_i}$;
- Case 2: $\|\mathbf{b}_i^*\| = \sqrt{\widetilde{\gamma}(\mathbf{B}_{[i,n_i)})} \cdot \mathrm{vol}(\mathbf{B}_{[i,n_i)})^{1/\beta_i}$;
- Case 3: $\|\mathbf{b}_i^*\| < \sqrt{\widetilde{\gamma}(\mathbf{B}_{[i,n_i)})} \cdot \mathrm{vol}(\mathbf{B}_{[i,n_i)})^{1/\beta_i}$.

Critically, since Case 3 may actually occur for some $i \in [n-45, n-2]$, the difference between the two prototypes emerges. They modify the tail block $\mathbf{B}_{[i,n)}$ differently: Algorithm 3 updates $\|\mathbf{b}_i^*\|$ (at Step 9), while Algorithm 4 does not and just moves to retrieve the next local block $\mathbf{B}_{[i+1,n_{i+1})}$.

We note that Steps 4-6 of Algorithm 4 are based on the following common practice of implementing BKZ:

**Fact 1** *BKZ implementations first pre-process each projected block $\mathbf{B}_{[i,n_i)}$ using BKZ recursively with a smaller block size (say, $\beta_i - 20$) and then call an SVP-oracle on it, so that $\mathbf{B}_{[i,n_i)}$ indeed becomes approximately HKZ-reduced at the end of the $i$-th BKZ iteration.*

**Algorithm 3** Prototype of Chen–Nguyen simulator (cf. Algorithm 2)

**Input:** A basis $\mathbf{B} = (\mathbf{b}_0, \ldots, \mathbf{b}_{n-1})$ of a lattice $\mathcal{L}$ and a block size $\beta \in [46, n]$.
**Output:** A reduced basis of $\mathcal{L}$ after running $N$ tours of BKZ-$\beta$ on the input basis.

**for** $j = 0$ to $N - 1$ **do**
  **if** there is an index $i \in [0, n-46]$ s.t. $\|\mathbf{b}_i^*\| > \sqrt{\widetilde{\gamma}(\mathbf{B}_{[i,n_i)})} \cdot \mathrm{vol}(\mathbf{B}_{[i,n_i)})^{1/\beta_i}$ **then**
    Find the smallest index $\underline{i}$ satisfying $\|\mathbf{b}_{\underline{i}}^*\| > \sqrt{\widetilde{\gamma}(\mathbf{B}_{[\underline{i},n_{\underline{i}})})} \cdot \mathrm{vol}(\mathbf{B}_{[\underline{i},n_{\underline{i}})})^{1/\beta_{\underline{i}}}$
    **for** $s = \underline{i}$ to $n - 46$ **do**
      Modify $\mathbf{b}_s, \ldots, \mathbf{b}_{n_s-1}$ so that $\|\mathbf{b}_s^*\| = \sqrt{\widetilde{\gamma}(\mathbf{B}_{[s,n_s)})} \cdot \mathrm{vol}(\mathbf{B}_{[s,n_s)})^{1/\beta_s}$
    **end for**
  **end if**
  **for** $i = n - 45$ to $n - 2$ **do**
    Modify $\mathbf{b}_i, \ldots, \mathbf{b}_{n_i-1}$ so that $\|\mathbf{b}_i^*\| = \sqrt{\widetilde{\gamma}(\mathbf{B}_{[i,n_i)})} \cdot \mathrm{vol}(\mathbf{B}_{[i,n_i)})^{1/\beta_i}$
  **end for**
**end for**
**return** $\mathbf{B}$.

**Algorithm 4** Prototype of our BKZ simulator (cf. Algorithm 5)

**Input:** A basis $\mathbf{B} = (\mathbf{b}_0, \ldots, \mathbf{b}_{n-1})$ of a lattice $\mathcal{L}$ and a block size $\beta \in [50, n]$.
**Output:** A reduced basis of $\mathcal{L}$ after running $N$ tours of BKZ-$\beta$ on the input basis.

**for** $j = 0$ to $N - 1$ **do**
  **for** $i = 0$ to $n - 2$ **do**
    **if** $\|\mathbf{b}_i^*\| > \sqrt{\widetilde{\gamma}(\mathbf{B}_{[i,n_i)})} \cdot \mathrm{vol}(\mathbf{B}_{[i,n_i)})^{1/\beta_i}$ **then**
      **for** $s = i$ to $n_i - 2$ **do**
        Modify $\mathbf{b}_s, \ldots, \mathbf{b}_{n_i-1}$ so that $\|\mathbf{b}_s^*\| = \sqrt{\widetilde{\gamma}(\mathbf{B}_{[s,n_i)})} \cdot \mathrm{vol}(\mathbf{B}_{[s,n_i)})^{1/(n_i-s)}$
      **end for**
    **end if**
  **end for**
**end for**
**return** $\mathbf{B}$.

**Key differences.** In order to explain why Chen-Nguyen's and our BKZ simulators behave differently for modeling the Z-shape, we highlight the two main differences between both prototypes (and as a consequence, both simulators) next.

At a high level, Algorithm 3 modifies many local blocks (especially in the tail) *unconditionally*, while Algorithm 4 modifies each local block *conditionally*.

The first difference is the way of modifying the head block. Once encountering $\|\mathbf{b}_i^*\| > \sqrt{\widetilde{\gamma}(\mathbf{B}_{[i,n_i)})} \cdot \mathrm{vol}(\mathbf{B}_{[i,n_i)})^{1/\beta_i}$, Algorithm 3 updates all remaining basis vectors $\mathbf{b}_i, \ldots, \mathbf{b}_{n-1}$ by (approximately) "SVP-reducing" each local block $\mathbf{B}_{[s,n_s)}$ for $s = i, \ldots, n-2$, while Algorithm 4 just (approximately) "HKZ-reduces" the current local block $\mathbf{B}_{[i,n_i)}$ and then moves to consider the next local block $\mathbf{B}_{[i+1,n_{i+1})}$.

This way of modifying the head block affects the Z-shape. For example, suppose that $\|\mathbf{b}_0^*\| > \sqrt{\widetilde{\gamma}(\mathbf{B}_{[0,\beta)})} \cdot \mathrm{vol}(\mathbf{B}_{[0,\beta)})^{1/\beta}$ at the beginning of a tour and the 0-th iteration modifies $\mathbf{B}_{[0,\beta)}$ so that the whole GSO $\mathbf{B}^*$ is (by coincidence)

of a Z-shape: during the remaining iterations of $n-2$, Algorithm 3 updates $\mathbf{b}_1, \ldots, \mathbf{b}_{n-1}$ *unconditionally* so that this Z-shape might vanish, while Algorithm 4 will keep $\mathbf{b}_1, \ldots, \mathbf{b}_{n-1}$ unchanged.

The second difference is the way of modifying the tail block. Again, Algorithm 3 updates the tail block $\mathbf{B}_{[n-45,n)}$ *unconditionally*. Yet, Algorithm 4 (approximately) "HKZ-reduces" each local block $\mathbf{B}_{[i,n_i)}$ iff $\|\mathbf{b}_i^*\| > \sqrt{\widetilde{\gamma}(\mathbf{B}_{[i,n_i)})} \cdot \mathrm{vol}(\mathbf{B}_{[i,n_i)})^{1/\beta_i}$. For example, suppose that for some index $i \geq n-45$, the GSO $\mathbf{B}^*$ at the end of the $(i-1)$-th iteration satisfies the Z-shape,

$$\|\mathbf{b}_i^*\| < \|\mathbf{b}_i^*\|^{-\frac{1}{\beta_i - 1}} < \widetilde{\gamma}(\mathbf{B}_{[i,n_i)})^{\frac{\beta_i}{2(\beta_i-1)}} \quad \text{and} \quad \|\mathbf{b}_{i+1}^*\| = \cdots = \|\mathbf{b}_{n-1}^*\| = 1, \quad (5)$$

then $\|\mathbf{b}_i^*\| < 1 < \sqrt{\widetilde{\gamma}(\mathbf{B}_{[i,n_i)})} \cdot \mathrm{vol}(\mathbf{B}_{[i,n_i)})^{1/\beta_i}$: at the $i$-th iteration, Algorithm 3 modifies $\mathbf{b}_i, \ldots, \mathbf{b}_{n_i-1}$ to produce a new $\|\mathbf{b}_i^*\| > 1$ and a new $\|\mathbf{b}_{i_0}^*\| < 1$ for some index $i_0 > i$ and hence destroys this Z-shape, while Algorithm 4 does not.

In summary, we identified two ways in which Algorithm 2/Algorithm 3 modify both the head and the tail blocks that may "destroy" the Z-shape, while Algorithm 4/Algorithm 5 avoid these steps. Of course, it remains to be shown that these two changes allow us to faithfully model the produced Z-shape.

### 3.2 A Z-shape BKZ simulator

By replacing each $\sqrt{\widetilde{\gamma}(\mathbf{B}_{[\cdot,\cdot)})}$ at Steps 3 and 5 with a Gaussian heuristic estimate $\mathrm{GH}(\cdot)$ and with an experiment-driven modification for small ranks, Algorithm 4 can be developed into a heuristic BKZ simulator for modeling the evolution of the Gram–Schmidt norm $\|\mathbf{b}_i^*\|$'s during the execution of BKZ-reducing a "random" input basis. In particular, such a simulator can be modified to predict the "Z-shape" behaviour of BKZ on $q$-ary lattices by modeling the evolution of the logarithmic GSO quantity $\log \|\mathbf{b}_i^*\|$'s during the execution of BKZ. This gives rise to a BKZ simulator tailored for random $q$-ary lattices, namely Algorithm 5. Given as input parameters $(k, n, q, \xi)$ corresponding to $q$-ary bases of the form

$$\begin{pmatrix} q\mathbf{I}_k & * \\ \mathbf{0} & \xi\mathbf{I}_{n-k} \end{pmatrix} \in \mathbb{Z}^{n \times n},$$

Algorithm 5 outputs a prediction for the Gram-Schmidt log-norms $\{\log \|\mathbf{b}_i^*\|\}_{0 \leq i < n}$ after running a sequence of BKZ tours on the above bases.

We summary the differences between Algorithms 4 and 5 as follows:

- The input to Algorithm 4 is an arbitrary basis, while the input to Algorithm 5 are key parameters for a random $q$-ary lattice/basis; this is a minor syntactic specialisation.
- We place some restrictions on the inputs, namely $q \geq 17$ and $q/\xi \geq 37/4$. These are derived from our numerical experiments and rule out instances not adequately captured by our simulator.
- Algorithm 4 updates basis vectors at its Steps 1-9, while Algorithm 5 updates logarithmic GSO quantity $\log \|\mathbf{b}_i^*\|$'s at its Steps 11-20;

– Each $\widetilde{\gamma}(\mathbf{B}_{[\cdot,\cdot)})$ in Algorithm 4 is replaced into a heuristic analog $r_i$ of Hermite's constant $\gamma_i$ in Algorithm 5.

Note that the input lattice and hence its volume remain unchanged during the execution of Algorithm 4. There is a corresponding loop invariant for Algorithm 5: the sum (of the logarithmic GSO quantity $\log \|\mathbf{b}_i^*\|$'s) $\sum_{i=0}^{n-1} \ell_i$ remains unchanged during Steps 11-20 of Algorithm 5. Indeed, consider the effect of Steps 12-19 wrt index $i$ on the $\ell_i$'s. The only change occurs on $\ell_i, \ldots, \ell_{n_i-1}$ at Steps 14-18. Suppose that Steps 14-18 transform $\ell_i, \ldots, \ell_{n_i-1}$ into $\ell'_i, \ldots, \ell'_{n_i-1}$. Then Step 16 wrt index $s = n_i - 1$ guarantees $\ell'_{n_i-1} = c_1 + \sum_{p=i}^{n_i-1} \ell_p - \sum_{p=i}^{n_i-2} \ell'_p$. The fact $c_1 = 0$ immediately implies $\sum_{p=i}^{n_i-1} \ell'_p = \sum_{p=i}^{n_i-1} \ell_p$. This proves that $\sum_{i=0}^{n-1} \ell_i$ remains unchanged during the execution of Algorithm 5.

**Algorithm 5** A BKZ simulator tailored for random $q$-ary lattices

**Input:** $(k, n, q, \xi)$, where $q \geq 17$ is the prime modulus, $k$ is the number of $q\mathbf{e}_i$-vectors, $n$ is the dimension, and the scale $\xi \in \mathbb{Z}^+$ such that $q/\xi \geq 37/4$.

**Output:** A prediction for the Gram-Schmidt log-norms $\{\ell_i = \log \|\mathbf{b}_i^*\|\}_{0 \leq i < n}$ after running $N$ tours of BKZ-reduction with block size $\beta \in [50, n)$.

**for** $i = 1$ to $\beta$ **do**
  By Eq. (6), extract the heuristic analog $r_i$ of Hermite's constant $\gamma_i$ in the context of BKZ-reducing random $q'$-ary bases with scale $\xi' \in \mathbb{Z}^+$ s.t. $q'/\xi' \geq 37/4$
  $c_i \leftarrow \log \sqrt{r_i}$ //We take log of $\sqrt{r_i}$ like Alg. 2, where $c_1 = 0$.
**end for**
**for** $i = 0$ to $k - 1$ **do**
  $\ell_i \leftarrow \log q$ //Steps 5-10 compute the initial $\log \|\mathbf{b}_i^*\|$'s.
**end for**
**for** $i = k$ to $n - 1$ **do**
  $\ell_i \leftarrow \log \xi$
**end for**
**for** $j = 0$ to $N - 1$ **do**
  **for** $i = 0$ to $n - 2$ **do**
    $g \leftarrow \sum_{p=i}^{n_i-1} \ell_p$
    **if** $\ell_i > c_{\beta_i} + g/\beta_i$ **then**
      **for** $s = i$ to $n_i - 1$ **do**
        $\ell_s \leftarrow c_t + (g - \sum_{p=i}^{s-1} \ell_p)/t$ where $t = n_i - s$ is the rank of local block $\mathbf{B}_{[s,n_i)}$
      **end for** //$\beta_i$ and $n_i$ are defined as in Eq. (2).
    **end if** //Steps 14-18 here correspond to Steps 3-7 of Algorithm 4.
  **end for**
**end for**
**return** $(\ell_0, \ldots, \ell_{n-1})$.

**Extracting heuristic analogs of Hermite's constants.** Like previous work [C-N11, BSW18], our Z-shape BKZ simulator requires to extract heuristic analogues of Hermite's constants but for BKZ-reducing random $q$-ary bases.

To do so, we first introduce the notion of an $(\cdot,\cdot)$-*AHKZ block* for convenience. Suppose that we run BKZ with block size $\beta$ and $N$ tours on a random $q$-ary basis (with parameters $(k,n,q,\xi)$) to output a basis $\mathbf{B}$ whose GSO $\mathbf{B}^*$ is of Z-shape such that for some integers $1 \leq s < t < n$, $t-s > \beta$, $\|\mathbf{b}_0^*\| = \cdots = \|\mathbf{b}_{s-1}^*\| = q$, the middle block $\mathbf{B}_{[s,t)}$ "looks like" a typical BKZ-reduced basis and $\|\mathbf{b}_{t-1}^*\| < \|\mathbf{b}_t^*\| = \cdots = \|\mathbf{b}_{n-1}^*\| = \xi$: in particular, the block $\mathbf{B}_{[t-\beta,t)}$ "looks like" being usually HKZ-reduced. We call $\mathbf{B}_{[t-\beta,t)}$ a $(\beta,N)$-*AHKZ block* (where A stands for approximate). This is because since $\mathbf{B}$ might not be fully BKZ-reduced, $\mathbf{B}_{[t-\beta,t)}$ might not be exactly HKZ-reduced.

It is not straight-forward to extract heuristic analogs $r_i$ of Hermite's constants $\gamma_i$ for (Step 2 of) Algorithm 5: the $r_i$'s not only determine the simulated GSO shape corresponding to the $(\beta,N)$-AHKZ block, but also determine the simulated number of the tail GSO vectors of length $\xi$. So we now explain how to extract such $r_i$'s based on the Gaussian heuristic estimates and $(\cdot,\cdot)$-AHKZ blocks computed experimentally.

First, let $r(i,k,n,q,\xi,\beta,N)$ denote the average $\frac{\|\mathbf{b}_{t-i}^*\|^2}{\mathrm{vol}(\mathbf{B}_{[t-i,t)})^{2/i}}$ of $(\beta,N)$-AHKZ blocks $\mathbf{B}_{[t-\beta,t)}$ of many independent trials running BKZ with block size $\beta$ and $N$ tours on random $q$-ary bases with the same parameters $(k,n,q,\xi)$. The *key observation* is that the $r(i,k,n,q,\xi,\beta,N)$'s appear to be independent of both the choice of the parameters $(k,n,q,\xi,\beta,N)$ (for $q/\xi \geq 37/4$ and $N \geq 5$) and the use of lattice reduction library FP(y)LLL (which relies on lattice-point enumeration) or G6K (which relies on sieving). For example, Figure 2 shows that the $r(i,80,230,q,\xi,80,8)$'s from FPyLLL have similar shapes for all $(q,\xi) \in \{(17,1),(17,4),(37,1),(37,4)\}$.

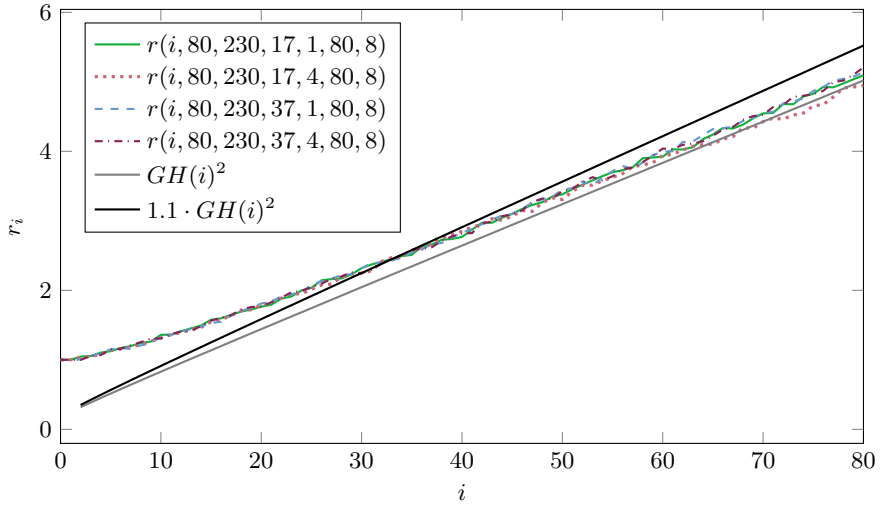


**Fig. 2.** Comparison of $r(i,\cdot,\cdot,\cdot,\cdot,\cdot,\cdot)$, $\mathrm{GH}(i)^2$ and $1.1 \cdot \mathrm{GH}(i)^2$.

Since we restrict the block size $\beta$ in Algorithm 5 to $\beta \geq 50$, we first set $r_\beta = \mathrm{GH}(\beta)^2$ based on the classical Gaussian heuristic estimate (as confirmed by the extensive experiments of [CN11] and used in [CN11, MW16, AWHT16, BSW18, LN20, ABLR21]).

Let $\widehat{\beta} := \min\{\beta - z, \beta'\}$ where $z \in \mathbb{Z}^+$ is a constant integer selected experimentally based on the concrete data $s_i$'s and block size $\beta$. Here, we experimentally compute

$$s_i \coloneqq r(i, k', n', q', \xi', \beta', N') \quad \text{for} \quad i = 1, \ldots, \beta'$$

by running many independent BKZ trials with block size $\beta' \geq 50$ and $N' \geq 5$ tours on random $q'$-ary bases with the same parameters $(k', n', q', \xi')$ where $q' \geq 17$ is a prime and $\xi'$ is a positive integer such that $q'/\xi' \geq 37/4$. We then set $r_i = s_i$ for $i = 1, \ldots, \widehat{\beta} - 2$.

It follows from Fact 1 that we set $r_i$ for $\widehat{\beta} - 1 \leq i < \beta$ by applying a simplified smooth rule, which avoids the influence of some single outlier $s_i$. Let $f_j(x) \coloneqq \frac{r_\beta - s_j}{\beta - j}(x - j) + s_j$ be the linear function of the line passing through two points $(j, s_j)$ and $(\beta, r_\beta)$. We take the average of $f_{\widehat{\beta}-2}(\widehat{\beta} - 1)$ and $f_{\widehat{\beta}-1}(\widehat{\beta} - 1)$ as $r_{\widehat{\beta}-1}$, and take the average of $f_j(i)$ over $j = \widehat{\beta}-2, \widehat{\beta}-1, \widehat{\beta}$ as $r_i$ for $i = \widehat{\beta}, \ldots, \beta-1$.

In summary, we specifically set the $r_i$'s at Step 2 of Algorithm 5 as follows:

$$r_i = \begin{cases} s_i & \text{for} \quad i = 1, \ldots, \widehat{\beta} - 2, \\ \frac{1}{2}\left(\frac{\mathrm{GH}(\beta)^2 - s_{\widehat{\beta}-2}}{\beta - \widehat{\beta} + 2} + \sum_{j=\widehat{\beta}-2}^{\widehat{\beta}-1} s_j\right) & \text{for} \quad i = \widehat{\beta} - 1, \\ \frac{1}{3}\sum_{j=\widehat{\beta}-2}^{\widehat{\beta}} \left(\frac{\mathrm{GH}(\beta)^2 - s_j}{\beta - j}(i - j) + s_j\right) & \text{for} \quad i = \widehat{\beta}, \ldots, \beta - 1, \\ \mathrm{GH}(\beta)^2 & \text{for} \quad i = \beta. \end{cases} \tag{6}$$

In our simulator source code `z.py` (attached in Appendix A.1), we set the $r_i$'s using the $s_i = r(i, 90, 200, 17, 1, 80, 8)$'s (from 20 FPyLLL BKZ trials with block size 80 and 8 tours on random $q'$-ary bases with the same parameters $(k', n', q', \xi') = (90, 200, 17, 1)$) and

$$z = \min\{7 + \lfloor \log \xi \rfloor, \beta - 4\}$$

(depending on the scale $\xi$, upper bounded by the block size $\beta$, and optimised experimentally): that is, our `z.py` uses

$$\widehat{\beta} = \min\{\beta - z, 80\} = \min\{\max\{\beta - 7 - \lfloor \log \xi \rfloor, 4\}, 80\} \in [4, 80].$$

### 3.3 Consistency with experiments

In Figure 3, we give experimental data comparing the BKZ implementation in FPyLLL with the output of our simulator in `z.py`. Further experimental evidence can be found in Appendix B. It shows that the prediction of the Gram-Schmidt norms $(\|\mathbf{b}_0^*\|, \|\mathbf{b}_1^*\|, \ldots, \|\mathbf{b}_{n-1}^*\|)$ by our simulator is fairly accurate for BKZ-reducing random $q$-ary bases. We interpret this to mean that our simulator

can give a good prediction of the "Z-shape" behaviour of BKZ with block size $\beta \geq 50$ and with 8 tours on random $q$-ary bases of scale $\xi \in \{1, 2, 3, 4\}$ satisfying $q \geq 17$ and $q/\xi \geq 37/4$, which includes cryptographic parameter ranges.

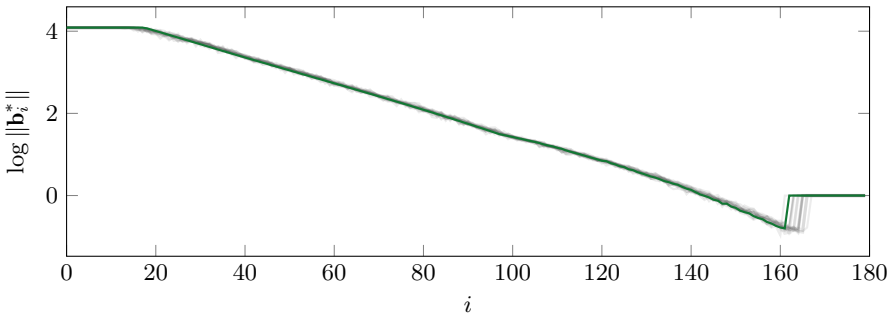


**Fig. 3.** Output (dark green line) of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 17, 1)$ compared with 16 runs of BKZ-65 (8 tours) as implemented in FPyLLL.

## 4 Faster enumeration with a Z-shape basis

A natural question to consider is whether the presence of a Z-shape makes primal lattice attacks against LWE faster/easier. That is, is solving SVP on the tail block of a Z-shaped BKZ reduced basis significantly faster than on the tail block of a random BKZ reduced basis? We provide a positive answer to this question in the context of enumeration-based attacks. More precisely, we heuristically justify that enumeration-based SVP solving on the tail projected block of a Z-shaped BKZ reduced basis is at least exponentially faster than on the tail block of an non Z-shape BKZ reduced basis.

We note that our restriction to enumeration rather than sieving is natural with current techniques. Known techniques for exploiting the basis shape in a lattice sieve to achieve super-polynomial speedups are based on (restricted) enumeration [ADH+19, DSvW21], specifically Babai's nearest plane algorithm [Bab86].

With that in mind, we now show that under reasonable heuristic assumptions, full or cylinder pruning enumeration (cf. Section 2.2) is at least exponentially faster on a tail block of a Z-shape reduced basis than that of an non Z-shape reduced basis. Here, an non Z-shape reduced basis corresponds to the models in previous literature using the GSA/the BKZ simulators [CN11, BSW18], while a Z-shape reduced basis corresponds to the model using our Z-shape BKZ simulator.

Before proceeding to our theorem, we make a few clarifying remarks.

– In the context of prime lattice attacks, we perform the last SVP enumeration on the tail block without the head $q\mathbf{e}_i$-vectors of a reduced basis of a $q$-

ary lattice (cf. Section 2.5). Therefore, we for simplicity ignore the head $q\mathbf{e}_i$-vectors in Theorem 2.

- A like-for-like comparison requires two bases $\mathbf{A}$ and $\mathbf{B}$ in Theorem 2 to be reduced by using BKZ with the same blocksize so that we can ideally assume both $\mathbf{A}$ and $\mathbf{B}_{[0,j)}$ there follow the Geometric Series Assumption (GSA) [Sch03] wrt the same constant. In practice, we can run BKZ on random $q$-ary lattices in a dual way and then extract the tail projected blocks to produce the requisite shapes for both $\mathbf{A}$ and $\mathbf{B}$.
- In the interest of a like-for-like comparison, Theorem 2 assumes that the volumes of the two input bases $\mathbf{A}$ and $\mathbf{B}$ differ at most by a polynomial factor. It is not hard to produce a situation where this condition is violated. For example, the shapes in grey in Figure 1 (results of actual BKZ-65 reductions) and the shapes predicted by the [CN11] or [BSW18] simulators visibly result in quite different volumes of tail blocks. Theorem 2 should be interpreted to mean that even if the volumes *only* differ by a polynomial factor we observe an exponential speedup. Since enumerating the canonical basis of $\mathbb{Z}^n$ is trivial, we more generally expect but do not prove that enumeration eventually becomes polynomial as the number $(n-j)$ of "Z-tail" vectors increases.
- Note that extreme cylinder pruning enumeration repeats cylinder pruning enumeration many times over different subsets $\mathrm{P}_f(\mathbf{B}, R)$ by randomising $\mathbf{B}$ [GNR10], but the numerically optimised pruning parameters in FP(y)LLL's `pruning` module [FPL19, FPy20] imply a small number of repetitions only when the basis is strongly pre-processed [ABF+20, ABLR21]. For a given specified reduced basis, only one full or one cylinder pruning enumeration is performed on it, especially in the regime where enumeration is asked to output an approximate solution to SVP rather than an exact one [ABLR21].

**Theorem 2.** *Let $\mathbf{A} = (\mathbf{a}_0, \ldots, \mathbf{a}_{n-1})$ and $\mathbf{B} = (\mathbf{b}_0, \ldots, \mathbf{b}_{n-1})$ be reduced bases of two full-rank lattices in $\mathbb{R}^n$ such that $n^{-O(1)} \leq \mathrm{vol}(\mathbf{A})/\mathrm{vol}(\mathbf{B}) \leq n^{O(1)}$, $\mathbf{B}_{[j,n)} = \xi \cdot \mathbf{I}_{n-j}$ with $j \in [1, n-1]$ and $\xi \in \mathbb{Z}^+$, and both $\mathbf{A}$ and $\mathbf{B}_{[0,j)}$ follow the same Geometric Series Assumption: there exists a constant $r \in (1, \frac{4}{3}]$ such that $\|\mathbf{a}_i^*\|/\|\mathbf{a}_{i+1}^*\| = r$ for all $i \in [0, n-2]$ and $\|\mathbf{b}_i^*\|/\|\mathbf{b}_{i+1}^*\| = r$ for all $i \in [0, j-2]$. Let $R > 0$ be a radius and $\rho \in (0, 1]$ be a constant such that*

$$f(i) = \begin{cases} \sqrt{\rho} \text{ if } 1 \leq i \leq n/2, \\ 1 \quad \text{if } n/2 < i \leq n \end{cases} \quad \text{and} \quad g(i) = \begin{cases} \sqrt{\rho} \text{ if } 1 \leq i \leq n - j/2, \\ 1 \quad \text{if } n - j/2 < i \leq n. \end{cases}$$

*Then the heuristic time complexity of one $(g, R)$-cylinder pruning enumeration on $\mathbf{B}$ is less than that of one $(f, R)$-cylinder pruning enumeration on $\mathbf{A}$ by a multiplicative factor of at least*

$$\frac{(e\pi\rho)^{(n-j)/4}}{\left(\sqrt{\frac{e\xi}{2R}} + \sqrt{\frac{e\pi\rho R}{(n-j)\xi}}\right)^{n-j}} \cdot \frac{j^{j/4}}{n^{n/4}} \cdot r^{(n^2-j^2)/8} \text{ (up to some polynomial factor).}$$

*Remark 1.* 1. Note that $\mathrm{vol}(\mathbf{A}) = r^{-n(n-1)/2} \cdot \|\mathbf{a}_0^*\|^n$ and $\mathrm{vol}(\mathbf{B}) = r^{-j(j-1)/2} \cdot \|\mathbf{b}_0^*\|^j \cdot \xi^{n-j}$. The volume constraint $n^{-O(1)} \leq \mathrm{vol}(\mathbf{A})/\mathrm{vol}(\mathbf{B}) \leq n^{O(1)}$ implies

$$\frac{\|\mathbf{a}_0^*\|^n}{n^{O(1)} \cdot \|\mathbf{b}_0^*\|^j \cdot \xi^{n-j}} \leq r^{\frac{n(n-1)-j(j-1)}{2}} \leq \frac{n^{O(1)} \cdot \|\mathbf{a}_0^*\|^n}{\|\mathbf{b}_0^*\|^j \cdot \xi^{n-j}}.$$

This means that for given $\|\mathbf{a}_0^*\|, \|\mathbf{b}_0^*\|, \xi$ and $r$, the number $(n-j)$ is impossible to be too large wrt $n$.

2. If $c := n - j$ is a constant, then one $(g, R)$-cylinder pruning enumeration on $\mathbf{B}$ is heuristically faster than one $(f, R)$-cylinder pruning enumeration on $\mathbf{A}$ by a multiplicative factor of at least

$$\frac{r^{\frac{cn}{4}}}{\left(\sqrt{\frac{e\xi}{2R}} + \sqrt{\frac{e\pi\rho R}{c\xi}}\right)^c} \text{ (up to some polynomial factor).}$$

This matches the intuition that the larger the number $(n-j)$ of $\|\mathbf{b}_j^*\| = \cdots = \|\mathbf{b}_{n-1}^*\| = \xi$, the (asymptotically) faster the enumeration on $\mathbf{B}$.

3. Let $R = \mathrm{GH}(n) \cdot \max\{\mathrm{vol}(\mathbf{A})^{1/n}, \mathrm{vol}(\mathbf{B})^{1/n}\}$. Suppose that both $\mathbf{A}$ and $\mathbf{B}$ are reduced by a lattice reduction algorithm with blocksize $\beta = O(n) \leq n$ so that the GSA constant $r = \mathrm{GH}(\beta)^{\frac{2}{\beta-1}}$ and $j = zn$ for some constant $z \in (0, 1)$. Using the approximation $\mathrm{GH}(\beta) \approx \sqrt{\frac{\beta}{2e\pi}} \cdot (\pi\beta)^{\frac{1}{2\beta}}$ and the polynomial bounds of $\mathrm{vol}(\mathbf{A})/\mathrm{vol}(\mathbf{B})$ , the speedup factor in Theorem 2 is essentially at least

$$\frac{(2\rho)^{(1-z)n/4} \cdot z^{zn/4}}{\left(\sqrt{\frac{e\xi}{\mathrm{vol}(\mathbf{A})^{1/n}}} + \sqrt{\frac{\rho \cdot \mathrm{vol}(\mathbf{A})^{1/n}}{(1-z)\xi}}\right)^{(1-z)n}} \cdot \left(\frac{\beta}{2e\pi}\right)^{\frac{(1-z^2)n^2}{8(\beta-1)}}.$$

For example, if $\beta = n - o(n)$, then the heuristic time complexity of one $(g, R)$-cylinder pruning enumeration on $\mathbf{B}$ and one $(f, R)$-cylinder pruning enumeration on $\mathbf{A}$ are respectively $2^{O(n)} \cdot \left(\sqrt{\frac{e\xi}{\mathrm{vol}(\mathbf{A})^{1/n}}} + \sqrt{\frac{\rho \cdot \mathrm{vol}(\mathbf{A})^{1/n}}{(1-z)\xi}}\right)^{(1-z)n} \cdot n^{\frac{z^2 n}{8}}$ and $2^{O(n)} \cdot n^{\frac{n}{8}}$: hence, the above speedup factor is indeed

$$\frac{2^{O(n)} \cdot n^{\frac{(1-z^2)n}{8}}}{\left(\sqrt{\frac{e\xi}{\mathrm{vol}(\mathbf{A})^{1/n}}} + \sqrt{\frac{\rho \cdot \mathrm{vol}(\mathbf{A})^{1/n}}{(1-z)\xi}}\right)^{(1-z)n}}.$$

4. Under the standard heuristic cost analysis [GNR10], both one $(f, R)$- and one $(g, R)$-cylinder pruning enumeration on $\mathbf{A}$ have the same cost.
5. If $\rho = 1$, then both enumerations on $\mathbf{A}$ and $\mathbf{B}$ are full (i.e. without pruning).

**Proof of Theorem 2.** We first estimate the cost of enumeration on $\mathbf{A}$.

[GNR10] showed that one $(f, R)$-cylinder pruning enumeration (for solving SVP) on $\mathbf{A}$ takes $T(\mathbf{A}) = \sum_{i=1}^{n} N_i$ polynomial-time operations, where $N_i$ denotes the cardinal of the set $\pi_{n-i}(\mathcal{L}(\mathbf{A}) \bigcap \mathrm{P}_f(\mathbf{A}, R))$. Under the Gaussian heuristic, we have:

$$N_i \approx \frac{\mathrm{vol}(\pi_{n-i}(\mathrm{P}_f(A, R)))}{\mathrm{vol}(\pi_{n-i}(\mathbf{A}))} = \frac{\mathrm{vol}(C_{f,i}) \cdot R^i}{\mathrm{vol}(\mathbf{A}_{[n-i,n)})} = \frac{\mathrm{vol}(C_{f,i}) \cdot R^i \cdot \mathrm{vol}(\mathbf{A}_{[0,n-i)})}{\mathrm{vol}(\mathbf{A})}.$$

By the analysis of [GNR10], since $\mathbf{A}$ satisfies the GSA, the number of enumeration nodes is maximized in the middle depth $i \approx n/2$. Let us consider

$$N_{n/2} \approx \frac{\mathrm{vol}(C_{f,n/2}) \cdot R^{n/2} \cdot \mathrm{vol}(\mathbf{A}_{[0,n/2)})}{\mathrm{vol}(\mathbf{A})}.$$

Since $\mathbf{A}$ satisfies the GSA that $\|\mathbf{a}_i^*\|/\|\mathbf{a}_{i+1}^*\| = r$ for all $i \in [0, n-2]$, it follows that $\mathrm{vol}(\mathbf{A}_{[0,n/2)}) = r^{n^2/8}\sqrt{\mathrm{vol}(\mathbf{A})}$. By our choice of $f$, $C_{f,n/2}$ is an $\frac{n}{2}$-dimensional ball of radius $\sqrt{\rho}$ and has volume $\mathrm{vol}(C_{f,n/2}) = \rho^{n/4} \cdot \mathrm{V}_{n/2}$, where $\mathrm{V}_i$ denotes the volume of the $i$-dimensional unit Euclidean ball. Note that $\mathrm{V}_{n/2} = \frac{\pi^{n/4}}{\Gamma(n/4+1)}$ and $\Gamma(n/4+1) \approx \sqrt{\frac{\pi n}{2}} \cdot \left(\frac{n}{4e}\right)^{n/4}$ (Stirling's formula). This implies:

$$N_{n/2} \approx \sqrt{\frac{2}{\pi n}} \cdot \left(\frac{4e\pi\rho}{n}\right)^{n/4} \cdot \frac{r^{n^2/8} R^{n/2}}{\sqrt{\mathrm{vol}(\mathbf{A})}}.$$

The standard heuristic estimate for the cost of one $(f, R)$-cylinder pruning enumeration on $\mathbf{A}$ is therefore for some $u(n) = n^{O(1)} \geq 1$:

$$T(\mathbf{A}) = u(n) \cdot \left(\frac{4e\pi\rho}{n}\right)^{n/4} \cdot \frac{r^{n^2/8} R^{n/2}}{\sqrt{\mathrm{vol}(\mathbf{A})}}. \tag{7}$$

We now estimate the cost of enumeration on $\mathbf{B}$. Since $\mathbf{B}$ is not a typical reduced basis due to its tail block $\mathbf{B}_{[j,n)}$, the standard heuristic cost analysis [GNR10] does not directly apply to the enumeration on $\mathbf{B}$. To address this issue, we use two observations:

- The top $(n-j)$ levels of one $(g, R)$-cylinder pruning enumeration on $\mathbf{B}$ indeed correspond to one full enumeration on the tail block $\mathbf{B}_{[j,n)}$ with radius $\sqrt{\rho} \cdot R$.
- The last $j$ levels of one $(g, R)$-cylinder pruning enumeration on $\mathbf{B}$ indeed perform cylinder pruning enumeration for solving the CVP instance

  $$(\mathbf{B}_{[0,j)}, -(\mathbf{b}_j, \ldots, \mathbf{b}_{n-1}) \cdot \mathbf{x})$$

  with radius $R$ over all

  $$\mathbf{x} \in \mathcal{S} := \left\{\mathbf{x} \in \mathbb{Z}^{n-j} : \|\mathbf{B}_{[j,n)}\mathbf{x}\| \leq \sqrt{\rho} \cdot R\right\}.$$

  Since $\mathbf{B}_{[0,j)}$ is a typical reduced block, we can apply the standard heuristic cost analysis [GNR10] here.

It follows that the number $T(\mathbf{B})$ of nodes of one $(g, R)$-cylinder pruning enumeration on $\mathbf{B}$ is the sum of the number $T(\mathbf{B}_{[j,n)})$ of nodes of one full enumeration on $\mathbf{B}_{[j,n)}$ with radius $\sqrt{\rho}R$ and the number $T(\mathbf{B}_{[0,j)}, \mathbf{x})$ of nodes of one cylinder pruning enumeration for solving the CVP instance $(\mathbf{B}_{[0,j)}, -(\mathbf{b}_j, \ldots, \mathbf{b}_{n-1}) \cdot \mathbf{x})$ with radius $R$ over all $\mathbf{x} \in \mathcal{S}$:

$$T(\mathbf{B}) = \sum_{\mathbf{x} \in \mathcal{S}} T(\mathbf{B}_{[0,j)}, \mathbf{x}) + T(\mathbf{B}_{[j,n)}).$$

Let $N$ denote the cardinal of the set $\mathcal{S}$ and $T(\mathbf{B}_{[0,j)}, \mathbf{y})$ (with $\mathbf{y} \in \mathcal{S}$) denote the maximum of $T(\mathbf{B}_{[0,j)}, \mathbf{x})$ over all $\mathbf{x} \in \mathcal{S}$. Then

$$T(\mathbf{B}) \leq N \cdot T(\mathbf{B}_{[0,j)}, \mathbf{y}) + T(\mathbf{B}_{[j,n)}).$$

Since $\mathbf{B}_{[0,j)}$ satisfies the GSA that $\|\mathbf{b}_i^*\|/\|\mathbf{b}_{i+1}^*\| = r$ for all $i \in [0, j-2]$, the standard heuristic cost analysis of cylinder pruning enumeration for solving SVP [GNR10] applies to the $(g', R)$-cylinder pruning enumeration for solving the CVP instance $(\mathbf{B}_{[0,j)}, -(\mathbf{b}_j, \ldots, \mathbf{b}_{n-1}) \cdot \mathbf{y})$ [LN13, ANS18], where $g'(\cdot)$ is the truncation of the bounding function $g(\cdot)$:

$$g'(i) = g(i + (n-j)) = \begin{cases} \sqrt{\rho} \text{ if } 1 \leq i \leq j/2, \\ 1 \quad \text{if } j/2 < i \leq j. \end{cases}$$

Similar to Equation (7), we have for some $u(j, \mathbf{y}) = j^{O(1)} \geq 1$:

$$T(\mathbf{B}_{[0,j)}, \mathbf{y}) = u(j, \mathbf{y}) \cdot \left(\frac{4e\pi\rho}{j}\right)^{j/4} \cdot \frac{r^{j^2/8} R^{j/2}}{\sqrt{\mathrm{vol}(\mathbf{B}_{[0,j)})}}.$$

Note that $\mathbf{B}_{[j,n)} = \xi \cdot \mathbf{I}_{n-j}$ and hence $\mathcal{S} := \left\{\mathbf{x} \in \mathbb{Z}^{n-j} : \|\mathbf{x}\| \leq \sqrt{\rho}R/\xi\right\}$, the Gaussian heuristic estimates its cardinal $N$ with an exponential error (cf. [MO90, Theorem 3]). We instead upper bound $N$ using the so-called Gaussian mass:

$$N \leq \left(\sqrt{e} + \sqrt{\frac{2e\pi\rho}{n-j}} \cdot \frac{R}{\xi}\right)^{n-j}. \tag{8}$$

*Proof of Eq.* (8). The *Gaussian mass* of a lattice $\mathcal{L}$ with parameter $s > 0$ is defined as $\rho_s(\mathcal{L}) = \sum_{\mathbf{y} \in \mathcal{L}} e^{-\frac{\pi\|\mathbf{y}\|^2}{s^2}}$. The Gaussian mass can be used to count discrete lattice points in a ball (cf. [SD17, p. 69]):

$$N \leq e^{\pi\rho R^2/(\xi s)^2} \cdot \rho_s(\mathbb{Z}^{n-j}) \quad \text{for any} \quad s > 0.$$

It is known that $\rho_s(\mathbb{Z}^{n-j}) = \rho_s(\mathbb{Z})^{n-j}$ for any $s > 0$ (cf. [SD17, p. 81]).

It follows from the Euler integral $2\int_0^\infty e^{-y^2} dy = \sqrt{\pi}$ that for any $s > 0$,

$$\rho_s(\mathbb{Z}) = 1 + 2\sum_{k=1}^{\infty} e^{-\pi k^2/s^2} \leq 1 + 2\int_0^\infty e^{-\pi y^2/s^2} dy = 1 + \frac{2s}{\sqrt{\pi}} \int_0^\infty e^{-y^2} dy = 1 + s.$$

Combining the above equations, we deduce that

$$N \leq e^{\pi\rho R^2/(\xi s)^2} \cdot (1+s)^{n-j} \quad \text{for any} \quad s > 0.$$

By setting $s = \sqrt{\frac{2\pi\rho}{n-j}} \cdot \frac{R}{\xi}$, this proves Eq. (8). □

Returning to the proof of Theorem 2, we now estimate $T(\mathbf{B}_{[j,n)})$. To enumerate all $\mathbf{x} \in \mathcal{S}$, we can first enumerate all $\mathbf{x}$ in the subset $\mathcal{S}' := \left\{\mathbf{x} \in \mathbb{N}^{n-j} : \|\mathbf{x}\| \leq \sqrt{\rho}R/\xi\right\}$ and then add signs to each non-zero coordinate of $\mathbf{x}$. It was shown in [ANS18, Alg. 3 and Th. 11] that the number $T'$ of enumeration nodes over $\mathcal{S}'$ can be linear in the cardinal $N'$ of $\mathcal{S}'$, namely $T' \leq O(n-j) \cdot N'$. This implies:

$$T(\mathbf{B}_{[j,n)}) \leq T' + N \leq O(n-j) \cdot N' + N \leq O(n-j) \cdot N.$$

Since $\mathrm{vol}(\mathbf{B}) = \mathrm{vol}(\mathbf{B}_{[0,j)}) \cdot \xi^{n-j}$, we combine it with the above four inequalities to deduce for some $v(n) = n^{O(1)} \geq 1$:

$$T(\mathbf{B}) \leq v(n) \cdot \left(\sqrt{e\xi} + \sqrt{\frac{2e\pi\rho}{(n-j)\xi}} \cdot R\right)^{n-j} \cdot \left(\frac{4e\pi\rho}{j}\right)^{j/4} \cdot \frac{r^{j^2/8} R^{j/2}}{\sqrt{\mathrm{vol}(\mathbf{B})}}.$$

We conclude that the speed-up of one $(g, R)$-cylinder pruning enumeration on $\mathbf{B}$ over one $(f, R)$-cylinder pruning enumeration on $\mathbf{A}$ is

$$\frac{T(\mathbf{A})}{T(\mathbf{B})} \geq \frac{u(n)}{v(n)} \cdot \sqrt{\frac{\mathrm{vol}(\mathbf{B})}{\mathrm{vol}(\mathbf{A})}} \cdot \frac{(e\pi\rho)^{(n-j)/4}}{\left(\sqrt{\frac{e\xi}{2R}} + \sqrt{\frac{e\pi\rho R}{(n-j)\xi}}\right)^{n-j}} \cdot \frac{j^{j/4}}{n^{n/4}} \cdot r^{(n^2-j^2)/8}.$$

This completes the proof of Theorem 2. □

## 5 Applications

In this section we assess the impact of the $q$-ary Z-shape on solving LWE instances derived from parameter sets in the literature. For this, we apply our new simulator to the NIST PQC Round 3 candidates Kyber [SAB+20], Saber [DKR+20] and NTRU [CDH+20]; we also consider Round 2 candidate LAC [LLJ+19] due to its choice of a small prime modulus $q = 251$. To do so, it does not suffice to run our simulator (that accurately models the Z-shape) on the $(d, q, \beta)$ tuples from the literature because the presence of a Z-shape might alter the smallest BKZ block size $\beta$ at which an attack succeeds. We thus refactored the "LWE estimator" [APS15] to support lattice reduction simulators such as ours and then used this improved "lattice estimator" to cost primal lattice attacks against LWE-based and NTRU-based schemes.

We present our results in Table 1 and our source code to produce this table in Appendix A. In this table the multicolumn "GSA" corresponds to cost estimates assuming the Geometric Series Assumption, the multicolumn "CN11" corresponds to the cost under the simulator from [CN11], the multicolumn "Z"

corresponds to the cost under our simulator. In all cases we assume sieving as the SVP oracle inside BKZ and for any BDD calls. In particular, we assume the cost model from [SAB+20] for lattice reduction but with the improvement for sieving from [MAT22] applied. In particular, we do not take the techniques from Section 4 into account here as sieving outperforms enumeration for the parameters considered here.[5]

The block of rows labelled "uSVP" refers to cost estimates using the `primal_usvp` function from the estimator as explained in Section 2.5. The block of rows labelled "BDD" refers to cost estimates using the `primal_bdd` from the estimator as also mentioned in Section 2.5. The block of rows labelled "Hybrid" refers to cost estimates using the `primal_hybrid` function from the estimator. Note that this class of algorithms is not relevant to the parameter sets considered here, solving costs are higher compared to the other two classes, but we include it here for completeness.

As can be observed by inspecting Table 1, the Z-shape indeed "disappears" for block sizes sufficiently large to solve the targeted LWE parameters. This can be seen by comparing the multicolumn "CN11" with "Z". We can thus rule out this threat to existing lattice reduction cost estimates against standardisation candidates.

## Acknowledgements

We thank Léo Ducas and Phong Q. Nguyen for helpful discussions.

## 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 Micciancio and Ristenpart [MR20], pages 186–212.

ABLR21. Martin R. Albrecht, Shi Bai, Jianwei Li, and Joe Rowell. Lattice reduction with approximate enumeration oracles - practical algorithms and concrete performance. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 732–759, Virtual Event, August 2021. Springer, Heidelberg.

ACD+18. Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer. Estimate all the LWE, NTRU schemes! In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 351–367. Springer, Heidelberg, September 2018.

ACW19. Martin R. Albrecht, Benjamin R. Curtis, and Thomas Wunderer. Exploring trade-offs in batch bounded distance decoding. In Kenneth G. Paterson and Douglas Stebila, editors, *SAC 2019*, volume 11959 of *LNCS*, pages 467–491. Springer, Heidelberg, August 2019.

[5] In this comparison, we ignore the potentially significant cost of memory access incurred by lattice sieving.

| Scheme | uSVP GSA cost | GSA $\beta$ | GSA $\eta$ | uSVP CN11 cost | CN11 $\beta$ | CN11 $\eta$ | uSVP Z cost | Z $\beta$ | Z $\eta$ |
|---|---|---|---|---|---|---|---|---|---|
| Kyber 512 | 143.8 | 406 | 406 | 146.8 | 417 | 417 | 146.8 | 417 | 417 |
| Kyber 768 | 204.9 | 624 | 624 | 209.9 | 642 | 642 | 209.9 | 642 | 642 |
| Kyber 1024 | 275.1 | 874 | 874 | 282.4 | 900 | 900 | 282.4 | 900 | 900 |
| LightSaber | 141.4 | 397 | 397 | 144.1 | 407 | 407 | 144.1 | 407 | 407 |
| Saber | 208.7 | 637 | 637 | 213.9 | 656 | 656 | 213.9 | 656 | 656 |
| FireSaber | 276.0 | 877 | 877 | 283.0 | 902 | 902 | 283.0 | 902 | 902 |
| NTRUHPS2048509Enc | 134.6 | 373 | 373 | 137.3 | 383 | 383 | 137.3 | 383 | 383 |
| NTRUHPS2048677Enc | 174.6 | 516 | 516 | 178.5 | 530 | 530 | 178.5 | 530 | 530 |
| NTRUHPS4096821Enc | 203.4 | 618 | 618 | 208.4 | 636 | 636 | 208.4 | 636 | 636 |
| NTRUHRSS701 | 162.3 | 471 | 471 | 165.7 | 483 | 483 | 165.9 | 484 | 484 |
| LAC-Light-v3a | 155.2 | 449 | 449 | 157.7 | 458 | 458 | 157.7 | 458 | 458 |
| LAC-128-v3a | 172.1 | 510 | 510 | 175.4 | 522 | 522 | 175.4 | 522 | 522 |
| LAC-192-v3a | 305.9 | 986 | 986 | 311.0 | 1007 | 1007 | 311.0 | 1007 | 1007 |
| LAC-256-v3a | 324.9 | 1054 | 1054 | 320.1 | 1037 | 1037 | 320.1 | 1037 | 1037 |

| Scheme | BDD GSA cost | GSA $\beta$ | GSA $\eta$ | BDD CN11 cost | CN11 $\beta$ | CN11 $\eta$ | BDD Z cost | Z $\beta$ | Z $\eta$ |
|---|---|---|---|---|---|---|---|---|---|
| Kyber 512 | 140.3 | 391 | 421 | 142.2 | 396 | 430 | 142.3 | 397 | 430 |
| Kyber 768 | 201.0 | 606 | 641 | 204.0 | 617 | 651 | 204.0 | 617 | 651 |
| Kyber 1024 | 270.8 | 855 | 890 | 275.4 | 870 | 908 | 275.5 | 872 | 907 |
| LightSaber | 137.8 | 382 | 412 | 139.6 | 387 | 420 | 139.7 | 387 | 421 |
| Saber | 204.9 | 620 | 655 | 208.7 | 633 | 669 | 208.7 | 633 | 669 |
| FireSaber | 271.8 | 858 | 894 | 278.1 | 880 | 917 | 278.1 | 880 | 917 |
| NTRUHPS2048509Enc | 131.1 | 357 | 390 | 131.4 | 358 | 391 | 131.5 | 358 | 391 |
| NTRUHPS2048677Enc | 170.8 | 498 | 533 | 170.1 | 496 | 530 | 170.1 | 496 | 530 |
| NTRUHPS4096821Enc | 199.7 | 601 | 636 | 200.7 | 604 | 640 | 200.7 | 603 | 641 |
| NTRUHRSS701 | 158.9 | 455 | 490 | 159.6 | 457 | 493 | 159.8 | 458 | 493 |
| LAC-Light-v3a | 150.8 | 427 | 462 | 149.0 | 420 | 456 | 149.0 | 420 | 456 |
| LAC-128-v3a | 167.3 | 486 | 522 | 172.4 | 460 | 542 | 172.7 | 458 | 543 |
| LAC-192-v3a | 300.6 | 962 | 998 | 304.7 | 884 | 1015 | 304.7 | 884 | 1015 |
| LAC-256-v3a | 323.5 | 990 | 1082 | 323.0 | 1034 | 1080 | 323.0 | 1034 | 1080 |

| Scheme | Hybrid GSA cost | GSA $\beta$ | GSA $\eta$ | Hybrid CN11 cost | CN11 $\beta$ | CN11 $\eta$ | Hybrid Z cost | Z $\beta$ | Z $\eta$ |
|---|---|---|---|---|---|---|---|---|---|
| Kyber 512 | 260.3 | 405 | 2 | 262.7 | 417 | 2 | 262.9 | 417 | 2 |
| Kyber 768 | 356.7 | 623 | 2 | 360.5 | 641 | 2 | 360.5 | 641 | 2 |
| Kyber 1024 | 498.8 | 873 | 2 | 508.9 | 899 | 2 | 508.9 | 899 | 2 |
| LightSaber | 278.4 | 397 | 2 | 280.7 | 407 | 2 | 281.0 | 407 | 2 |
| Saber | 432.5 | 637 | 2 | 438.0 | 655 | 2 | 438.1 | 655 | 2 |
| FireSaber | 559.2 | 877 | 2 | 571.3 | 901 | 2 | 571.4 | 901 | 2 |
| NTRUHPS2048509Enc | 185.5 | 373 | 2 | 187.8 | 383 | 2 | 187.8 | 383 | 2 |
| NTRUHPS2048677Enc | 229.2 | 515 | 2 | 233.0 | 527 | 2 | 233.0 | 527 | 2 |
| NTRUHPS4096821Enc | 297.8 | 617 | 2 | 300.5 | 635 | 2 | 300.6 | 635 | 2 |
| NTRUHRSS701 | 232.9 | 471 | 2 | 234.2 | 483 | 2 | 234.4 | 483 | 2 |
| LAC-Light-v3a | 183.1 | 371 | 2 | 186.3 | 399 | 2 | 186.3 | 399 | 2 |
| LAC-128-v3a | 240.2 | 509 | 2 | 249.7 | 521 | 2 | 249.7 | 521 | 2 |
| LAC-192-v3a | 359.5 | 863 | 2 | 367.3 | 885 | 2 | 367.3 | 885 | 2 |
| LAC-256-v3a | 432.3 | 1052 | 2 | 446.1 | 1037 | 2 | 446.1 | 1037 | 2 |

**Table 1.** Parameters for NIST Candidates

AD97. Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *29th ACM STOC*, pages 284–293. ACM Press, May 1997.

AD21. M. R. Albrecht and L. Ducas. Lattice attacks on NTRU and LWE: A history of refinements. https://eprint.iacr.org/2021/799, 2021.

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.

ADPS16. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016*, pages 327–343. USENIX Association, August 2016.

AGVW17. Martin R. Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. Revisiting the expected cost of solving uSVP and applications to LWE. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 297–322. Springer, Heidelberg, December 2017.

ANS18. Yoshinori Aono, Phong Q. Nguyen, and Yixin Shen. Quantum lattice enumeration and tweaking discrete pruning. In Peyrin and Galbraith [PG18], pages 405–434.

APS15. Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

AWHT16. Yoshinori Aono, Yuntao Wang, Takuya Hayashi, and Tsuyoshi Takagi. Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. In Fischlin and Coron [FC16], pages 789–819.

Bab86. L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

BG14. Shi Bai and Steven D. Galbraith. Lattice decoding attacks on binary LWE. In Willy Susilo and Yi Mu, editors, *ACISP 14*, volume 8544 of *LNCS*, pages 322–337. Springer, Heidelberg, July 2014.

Bli14. H. F. Blichfeldt. A new principle in the geometry of numbers, with some applications. *Trans. Am. Math. Soc.*, 16:227–235, 1914.

BSW18. Shi Bai, Damien Stehlé, and Weiqiang Wen. Measuring, simulating and exploiting the head concavity phenomenon in BKZ. In Peyrin and Galbraith [PG18], pages 369–404.

BV11. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011.

CDH+20. Cong Chen, Oussama Danba, Jeffrey Hoffstein, Andreas Hulsing, Joost Rijneveld, John M. Schanck, Peter Schwabe, William Whyte, Zhenfei Zhang, Tsunekazu Saito, Takashi Yamakawa, and Keita Xagawa. NTRU. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

Che13. Y. Chen. *Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe*. Ph.D. thesis, ENS, France, 2013.

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.

CS87. J. H. Conway and N. J. A. Sloane. *Sphere-packings, Lattices, and Groups*. Springer, 1987.

DDGR20. Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. LWE with side information: Attacks and concrete security estimation. In Micciancio and Ristenpart [MR20], pages 329–358.

DKL+18. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR TCHES*, 2018(1):238–268, 2018. https://tches.iacr.org/index.php/TCHES/article/view/839.

DKR+20. Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, Frederik Vercauteren, Jose Maria Bermudo Mera, Michiel Van Beirendonck, and Andrea Basso. SABER. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

DSvW21. Léo Ducas, Marc Stevens, and Wessel P. J. van Woerden. Advanced lattice sieving on GPUs, with tensor cores. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 249–279. Springer, Heidelberg, October 2021.

FC16. Marc Fischlin and Jean-Sébastien Coron, editors. *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*. Springer, Heidelberg, May 2016.

FP85. U. Fincke and M. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of Computation*, 44(170):463–463, May 1985.

FPL19. FPLLL development team. FPLLL, a lattice reduction library. Available at https://github.com/fplll/fplll, 2019.

FPy20. FPyLLL development team. FPyLLL, a Python interface to FPLLL. Available at https://github.com/fplll/fpylll, 2020.

Gil10. Henri Gilbert, editor. *EUROCRYPT 2010*, volume 6110 of *LNCS*. Springer, Heidelberg, May / June 2010.

GNR10. Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In Gilbert [Gil10], pages 257–278.

HGSW03. N. Howgrave-Graham, J. H. Silverman, and W. Whyte. A meet-in-the-middle attack on an NTRU private key. Technical report 004, 2003. Version 2. NTRU Cryptosystems.

How07. Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 150–169. Springer, Heidelberg, August 2007.

HPS98. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *ANTS*, pages 267–288, 1998.

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.

HPS+17. Jeffrey Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte, and Zhenfei Zhang. Choosing parameters for NTRUEncrypt. In Helena Handschuh, editor, *CT-RSA 2017*, volume 10159 of *LNCS*, pages 3–18. Springer, Heidelberg, February 2017.

Kan83. R. Kannan. Improved algorithms for integer programming and related lattice problems. In *STOC*, pages 193–206, 1983. Full version at *Mathematics of Operations Research* 1987 with title "Minkowski's Convex Body Theorem and Integer Programming".

LLJ+19. Xianhui Lu, Yamin Liu, Dingding Jia, Haiyang Xue, Jingnan He, Zhenfei Zhang, Zhe Liu, Hao Yang, Bao Li, and Kunpeng Wang. LAC. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

LLL82. Arjen Klaas Lenstra, Hendrik W. Lenstra Jr., and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:366–389, 1982.

LM09. Vadim Lyubashevsky and Daniele Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 577–594. Springer, Heidelberg, August 2009.

LN13. Mingjie Liu and Phong Q. Nguyen. Solving BDD by enumeration: An update. In Ed Dawson, editor, *CT-RSA 2013*, volume 7779 of *LNCS*, pages 293–309. Springer, Heidelberg, February / March 2013.

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.

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Gilbert [Gil10], pages 1–23.

LS15. Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptogr.*, 75(3):565–599, 2015.

MAT22. MATZOV. Report on the Security of LWE: Improved Dual Lattice Attack, April 2022.

MH73. J. Milnor and D. Husemoller. *Symmetric bilinear forms*. Springer, 1973.

Mic81. P. Michael. On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications. *SIGSAM Bulletin*, 15:37–44, 1981.

MO90. J. E. Mazo and A. M. Odlyzko. Lattice points in high-dimensional spheres. *Monatshefte für Mathematik*, 110:47–61, 1990.

MR09. Daniele Micciancio and Oded Regev. Lattice-based cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*, pages 147–191. Springer, Heidelberg, Berlin, Heidelberg, New York, 2009.

MR20. Daniele Micciancio and Thomas Ristenpart, editors. *CRYPTO 2020, Part II*, volume 12171 of *LNCS*. Springer, Heidelberg, August 2020.

MW15. Daniele Micciancio and Michael Walter. Fast lattice point enumeration with minimal overhead. In Piotr Indyk, editor, *26th SODA*, pages 276–294. ACM-SIAM, January 2015.

MW16. Daniele Micciancio and Michael Walter. Practical, predictable lattice basis reduction. In Fischlin and Coron [FC16], pages 820–849.

PG18. Thomas Peyrin and Steven Galbraith, editors. *ASIACRYPT 2018, Part I*, volume 11272 of *LNCS*. Springer, Heidelberg, December 2018.

PV21. Eamonn W. Postlethwaite and Fernando Virdia. On the success probability of solving unique SVP via BKZ. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 68–98. Springer, Heidelberg, May 2021.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

Rog56. C. A. Rogers. The number of lattice points in a set. *Proceedings of the London Mathematical Society*, 3-6, 1956.

SAB+20. Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

Sch87. Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, pages 201–224, 1987.

Sch03. Claus-Peter Schnorr. Lattice reduction by random sampling and birthday methods. In *Proceedings of STACS '03*, volume 2607 of *LNCS*, pages 145–156. Springer, 2003.

SD17. N. Stephens-Davidowitz. *On the Gaussian Measure Over Lattices*. Ph.d. thesis, New York University, 2017.

SE94. Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.*, 66:181–199, 1994.

SH95. Claus-Peter Schnorr and Horst Helmut Hörner. Attacking the Chor-Rivest cryptosystem by improved lattice reduction. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *EUROCRYPT'95*, volume 921 of *LNCS*, pages 1–12. Springer, Heidelberg, May 1995.

SSTX09. Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 617–635. Springer, Heidelberg, December 2009.

Wun16. Thomas Wunderer. Revisiting the hybrid attack: Improved analysis and refined security estimates. Cryptology ePrint Archive, Report 2016/733, 2016. `https://eprint.iacr.org/2016/733`.

Wun19. Thomas Wunderer. A detailed analysis of the hybrid lattice-reduction and meet-in-the-middle attack. *Journal of Mathematical Cryptology*, 13(1):1–26, 2019.

# A Source code

Our code relies on a revised version of the LWE estimator from [APS15] available at `https://anonymous.4open.science/r/lattice-estimator-0F4F/README.rst`. We do not claim this revised version of this estimator as a contribution of this work, only the feature allowing to "plug in" BKZ simulators such as ours into this framework. We also attach the source code files listed below, see z.py and estimate.py.

## A.1 z.py

```
# -*- coding: utf-8 -*-
from sage.all import log, log_gamma, pi, exp, floor, RealField
from sage.all import cached_function


def gso(d, m, q, scale, prec=53):
    """
    Initial GSO of the q-ary lattice
    :param d: Lattice dimension.
    :param m: Number of ‘q‘ vectors
    :param q: Modulus ‘q‘
    """

    RR = RealField(prec)

    r = [0] * d
    for i in range(0, m, 1):
        r[i] = RR(q ** 2)
    for j in range(m, d, 1):
        r[j] = scale ** 2
    return r


@cached_function
def ah_constant(beta, scale, prec=53):
    RR = RealField(prec)

    h = [
        RR(1.00000000000000),
        RR(1.00000000000000),
        RR(1.02492695474985),
        RR(1.05161980747554),
        RR(1.10445074764426),
        RR(1.12738094965280),
        RR(1.18113914825808),
        RR(1.21037399094975),
        RR(1.25380229130682),
        RR(1.31275912013118),
        RR(1.32737148909975),
        RR(1.36036038598451),
        RR(1.43966294868932),
        RR(1.46160335381056),
        RR(1.55544109937333),
        RR(1.51556419271870),
        RR(1.62261393856106),
        RR(1.62261542586149),
        RR(1.69386128187285),
        RR(1.74294139717984),
        RR(1.76511645842542),
        RR(1.85062593391314),
        RR(1.92910191727097),
```

```
    RR(1.95924301565493),
    RR(2.01827070103134),
    RR(2.09555655837831),
    RR(2.10779553489076),
    RR(2.16072906694966),
    RR(2.17866707693959),
    RR(2.24637113675736),
    RR(2.30374935080714),
    RR(2.32703241781875),
    RR(2.38719370287110),
    RR(2.44070021621789),
    RR(2.47587251254528),
    RR(2.57636556347475),
    RR(2.59580776592391),
    RR(2.70737106887894),
    RR(2.71293172456551),
    RR(2.80972868887803),
    RR(2.85337256682321),
    RR(2.96643112072014),
    RR(2.92901889083372),
    RR(2.99441283605227),
    RR(3.03884800785631),
    RR(3.09367293382578),
    RR(3.15593321975939),
    RR(3.21699662499604),
    RR(3.26306485591489),
    RR(3.32320705128515),
    RR(3.37871256503968),
    RR(3.49314144958915),
    RR(3.52315959119194),
    RR(3.60596719185516),
    RR(3.62302348800297),
    RR(3.67106448779994),
    RR(3.75288918941554),
    RR(3.80729231066355),
    RR(3.83017943995872),
    RR(3.87680907526471),
    RR(3.99304695483873),
    RR(4.04539449313441),
    RR(4.12835783024051),
    RR(4.12989317768338),
    RR(4.25784341671535),
    RR(4.23117443161792),
    RR(4.34900834929189),
    RR(4.37617075735239),
    RR(4.44178322064643),
    RR(4.53471153665029),
    RR(4.54352770842019),
    RR(4.65495019943735),
    RR(4.63508961564190),
    RR(4.73512141747094),
    RR(4.76413486921241),
    RR(4.87996877805749),
    RR(4.94652066938356),
    RR(4.98495063065338),
    RR(5.07692160777643),
    RR(5.04745948303442),
    RR(5.12518339690435),
]

# h[i] is the average
# $\frac{\|\vec{c}_{80-i}^{\ast}\|^{2}}{\vol(C_{\range{80-i,80}})^{2/i}}$ of
# approximately HKZ-reduced projected block $C$ in rank 80 of 20 random
# \(q\)-ary lattices with parameters (m,n,q,scale)=(90,200,17,1) after
# running FPyLLL BKZ-80 reduction with max-loops 8. h[i] and ah[i] are the
# heuristic analog of Hermite's constant $\gamma_{i}$ in the context of
# FPyLLL BKZ-reduction.
```

```
    ah = [RR(0.0)] * (beta + 1)
    c = [RR(0.0)] * (beta + 1)
    for j in range(0, min(beta, 80) + 1, 1):
        ah[j] = h[j]
    if beta > 49:
        lga = RR(
            log_gamma(beta / 2 + 1) * (2 / beta) - log(pi)
        )
        ah[beta] = RR(exp(lga))
        # ah[beta]=GH(beta)^2. x is chosen to smooth ah[j] for j in [x,beta-1],
        # where ''7'' is the ''best'' using the above data h[i]'s.
        z = min(
            7 + floor(log(scale, 2)), beta - 4
        )  # z is related to scale, but can be upper bounded.
        x = min(beta - z, 80)
        ah[x - 1] = (
            (ah[beta] - h[x - 2]) / (beta - x + 2) / 2
            + h[x - 2] / 2
            + h[x - 1] / 2
        )
        for j in range(x, beta, 1):
            # (j,u) is on the stright line through two points (x-2,h[x-2]) and (beta,ah[beta]).
            u = (ah[beta] - h[x - 2]) * (j - x + 2) / (beta - x + 2) + h[x - 2]
            # (j,v) is on the stright line through two points (x-1,h[x-1]) and (beta,ah[beta]).
            v = (ah[beta] - h[x - 1]) * (j - x + 1) / (beta - x + 1) + h[x - 1]
            # (j,w) is on the straight line through two points (x,h[x]) and (beta,ah[beta]).
            w = (ah[beta] - h[x]) * (j - x) / (beta - x) + h[x]
            ah[j] = (
                u + v + w
            ) / 3  # We take the average of u, v and w as ah[j].
    c[0] = 0
    c[1] = 0
    for j in range(2, beta + 1, 1):
        c[j] = 0.5 * RR(log(ah[j], 2.0))
    return c


def one_tour(r, block_size, c):
    """
    Updates the log GSO for one (simulated) BKZ tour on q-ary lattice.
    """
    d = len(r)
    for i in range(d - 1):
        p = min(block_size, d - i)
        j = min(i + block_size, d)
        g = sum([r[z] for z in range(i, j, 1)])
        if r[i] > c[p] + g / p:
            r[i] = c[p] + g / p
            for s in range(i + 1, j, 1):
                t = j - s
                y = sum([r[z] for z in range(i, s, 1)])
                r[s] = c[t] + (g - y) / t
    return r


def qary_simulate_r(r, block_size, tours=1, scale=1, prec=53):
    RR = RealField(prec)
    c = ah_constant(block_size, scale)
    r = [RR(log(r_, 2.0)) / 2.0 for r_ in r]
    for i in range(tours):
        r = one_tour(r, block_size, c)
    r = [2.0 ** (2 * r_) for r_ in r]
    return r


def qary_simulate(d, m, q, block_size, tours=1, scale=1):
    r = gso(d, m, q, scale)
    r = qary_simulate_r(r, block_size=block_size, tours=tours, scale=scale)
```

```
    return r
```

## A.2 estimate.py

```
# -*- coding: utf-8 -*-
"""
Run with ::

    sage: sage: from estimate import *
    sage: %time r = batch_estimate(schemes, algorithms, jobs=16)

"""

from functools import partial

from estimator.estimator.util import batch_estimate  # noqa
from estimator.estimator.simulator import qary_simulator, CN11, GSA  # noqa
from estimator.estimator import LWE, ND, Logging  # noqa
from estimator.estimator.lwe_primal import primal_usvp, primal_bdd, primal_hybrid
# noqa

from estimator.estimator.schemes import (
    Kyber512,
    Kyber768,
    Kyber1024,
    LightSaber,
    Saber,
    FireSaber,
    NTRUHPS2048509Enc,
    NTRUHPS2048677Enc,
    NTRUHPS4096821Enc,
    NTRUHRSS701Enc,
)


def ZSimulator(d, m, q, beta, xi=1, tau=1, dual=False):
    """

    :param d: Lattice dimension.
    :param m: Number of ‘q‘ vectors
    :param q: Modulus ‘q‘
    :param beta: Block size $\beta$.
    :param xi: Scaling factor $\xi$ for identity part.
    :param tau: Kannan factor $\tau$.
    :param dual: perform reduction on the dual.

    """
    from z import qary_simulate_r

    def f(r, beta):
        return qary_simulate_r(r, beta, tours=8)

    return qary_simulator(f=f, d=d, n=m, q=q, beta=beta, xi=xi, tau=tau, dual=dual)


LACLightv3a = LWE.Parameters(
    n=512,
    q=251,
    Xs=ND.SparseTernary(512, 128 // 2),
    Xe=ND.SparseTernary(512, 128 // 2),
    m=512,
    tag="LAC-Light-v3a",
)
LAC128v3a = LWE.Parameters(
    n=512,
    q=251,
    Xs=ND.SparseTernary(512, 256 // 2),
    Xe=ND.SparseTernary(512, 256 // 2),
```

```
    m=512,
    tag="LAC-128-v3a",
)
LAC192v3a = LWE.Parameters(
    n=1024,
    q=251,
    Xs=ND.SparseTernary(1024, 256 // 2),
    Xe=ND.SparseTernary(1024, 256 // 2),
    m=1024,
    tag="LAC-192-v3a",
)
LAC256v3a = LWE.Parameters(
    n=1024,
    q=251,
    Xs=ND.SparseTernary(1024, 384 // 2),
    Xe=ND.SparseTernary(1024, 384 // 2),
    m=1024,
    tag="LAC-256-v3a",
)

schemes = (
    Kyber512,
    Kyber768,
    Kyber1024,
    LightSaber,
    Saber,
    FireSaber,
    NTRUHPS2048509Enc,
    NTRUHPS2048677Enc,
    NTRUHPS4096821Enc,
    NTRUHRSS701Enc,
    LACLightv3a,
    LAC128v3a,
    LAC192v3a,
    LAC256v3a,
)

algorithms_usvp = (
    partial(primal_usvp, red_shape_model=GSA),
    partial(primal_usvp, red_shape_model=CN11),
    partial(primal_usvp, red_shape_model=ZSimulator),
)
algorithms_bdd = (
    partial(primal_bdd, red_shape_model=GSA),
    partial(primal_bdd, red_shape_model=CN11),
    partial(primal_bdd, red_shape_model=ZSimulator),
)

algorithms_hybrid = (
    partial(primal_hybrid, red_shape_model=GSA),
    partial(primal_hybrid, red_shape_model=CN11),
    partial(primal_hybrid, red_shape_model=ZSimulator),
)

algorithms = algorithms_usvp + algorithms_bdd + algorithms_hybrid
```

# B Experimental evidence

## B.1 Comparison with FPyLLL

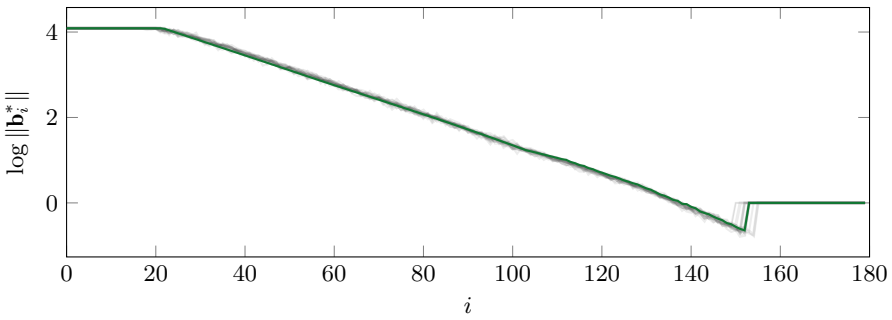


**Fig. 4.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 17, 1)$ compared with 16 runs of BKZ-50 (8 tours) as implemented in FPyLLL.

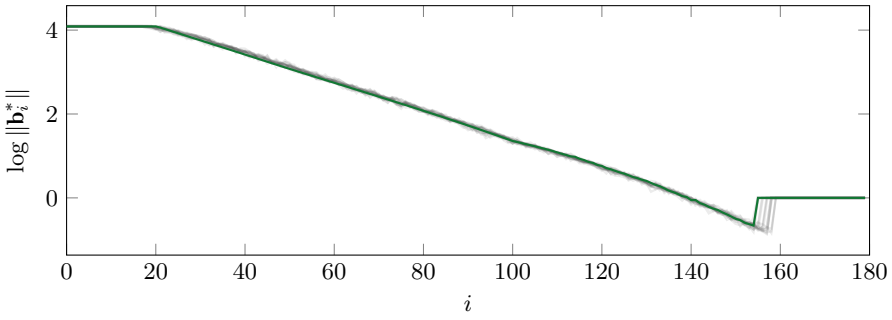


**Fig. 5.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 17, 1)$ compared with 16 runs of BKZ-55 (8 tours) as implemented in FPyLLL.

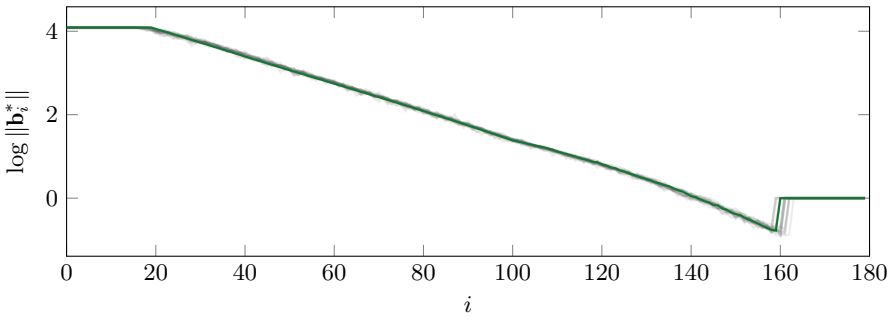


**Fig. 6.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 17, 1)$ compared with 16 runs of BKZ-60 (8 tours) as implemented in FPyLLL.

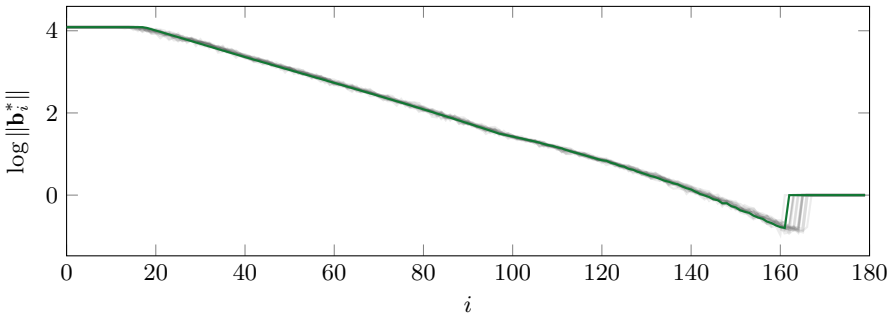


**Fig. 7.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 17, 1)$ compared with 16 runs of BKZ-65 (8 tours) as implemented in FPyLLL.

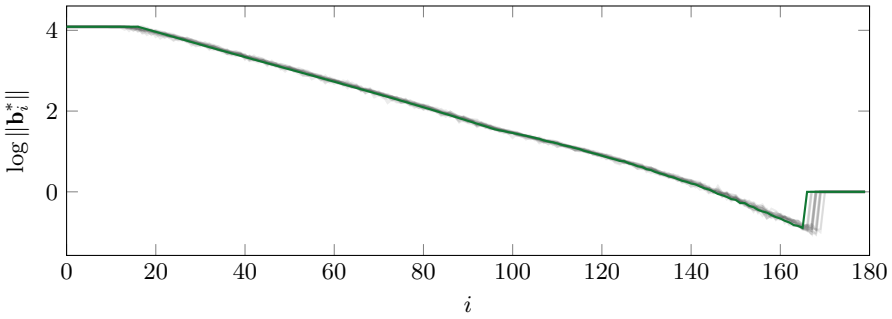


**Fig. 8.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 17, 1)$ compared with 16 runs of BKZ-70 (8 tours) as implemented in FPyLLL.

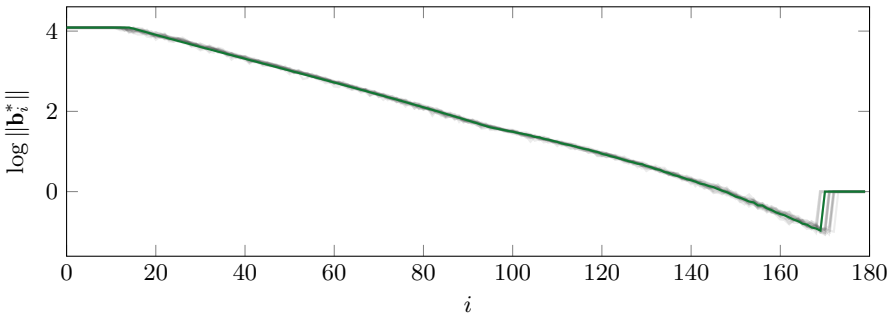


**Fig. 9.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 17, 1)$ compared with 16 runs of BKZ-75 (8 tours) as implemented in FPyLLL.

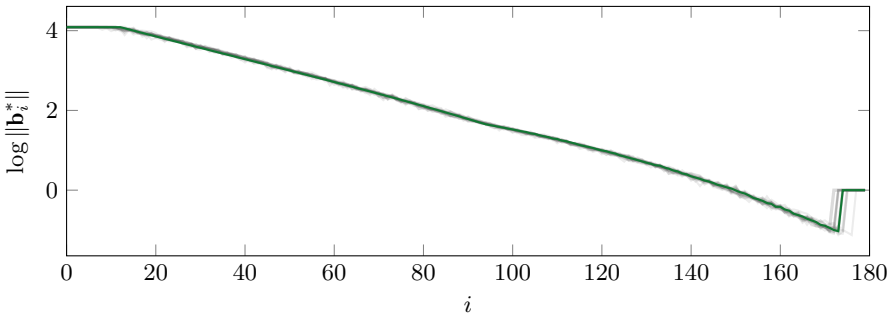


**Fig. 10.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 17, 1)$ compared with 16 runs of BKZ-80 (8 tours) as implemented in FPyLLL.

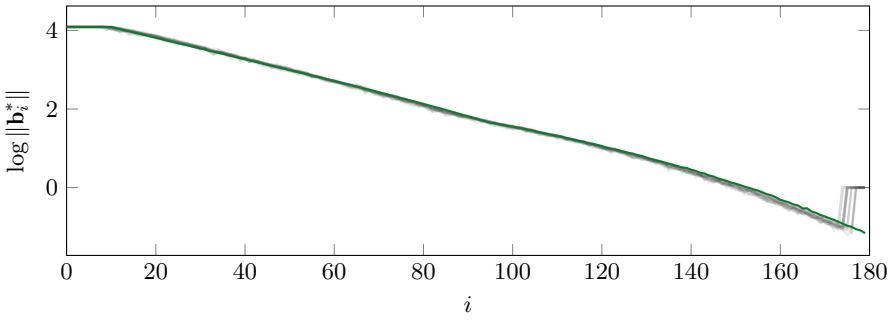


**Fig. 11.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 17, 1)$ compared with 16 runs of BKZ-85 (8 tours) as implemented in FPyLLL.

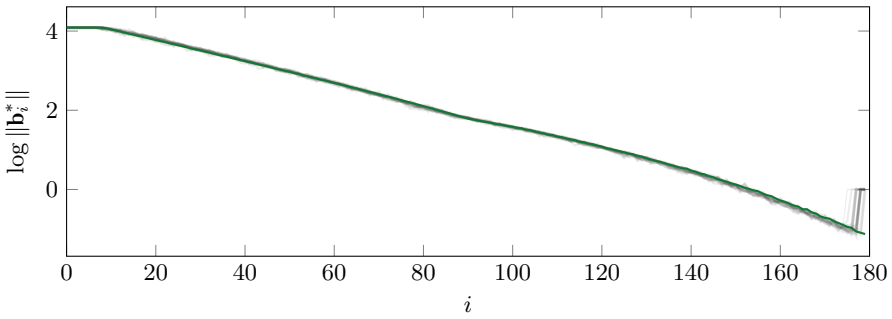


**Fig. 12.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 17, 1)$ compared with 16 runs of BKZ-90 (8 tours) as implemented in FPyLLL.

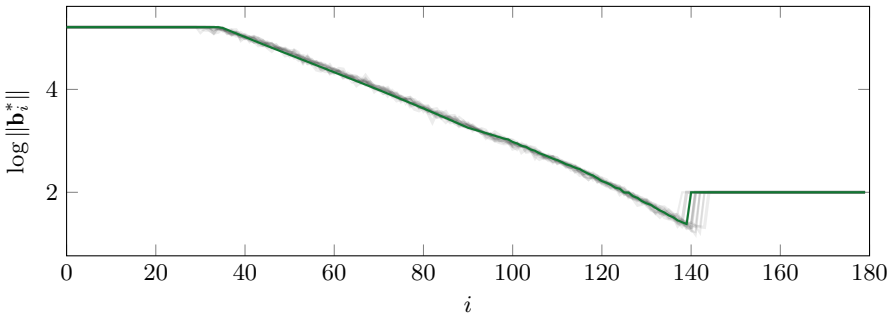


**Fig. 13.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 37, 4)$ compared with 16 runs of BKZ-50 (8 tours) as implemented in FPyLLL.

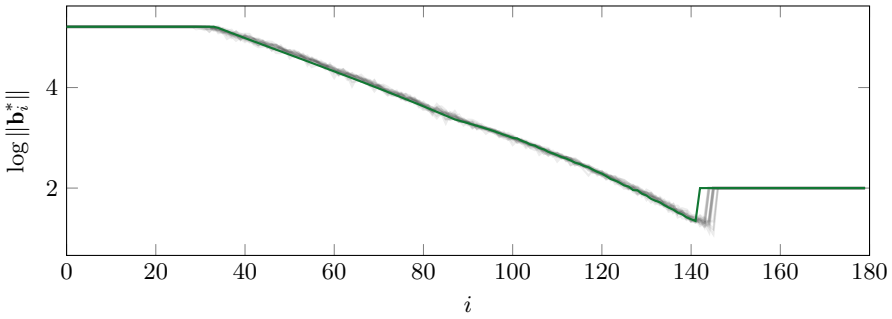


**Fig. 14.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 37, 4)$ compared with 16 runs of BKZ-55 (8 tours) as implemented in FPyLLL.

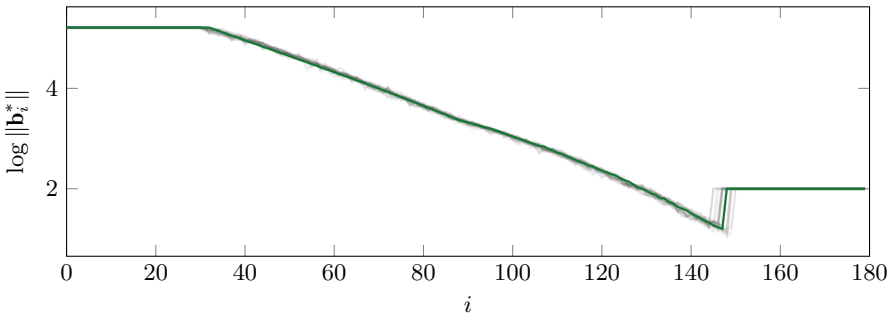


**Fig. 15.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 37, 4)$ compared with 16 runs of BKZ-60 (8 tours) as implemented in FPyLLL.

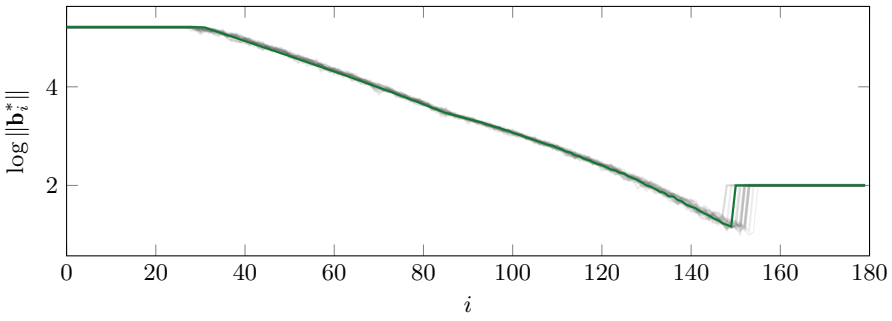


**Fig. 16.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 37, 4)$ compared with 16 runs of BKZ-65 (8 tours) as implemented in FPyLLL.

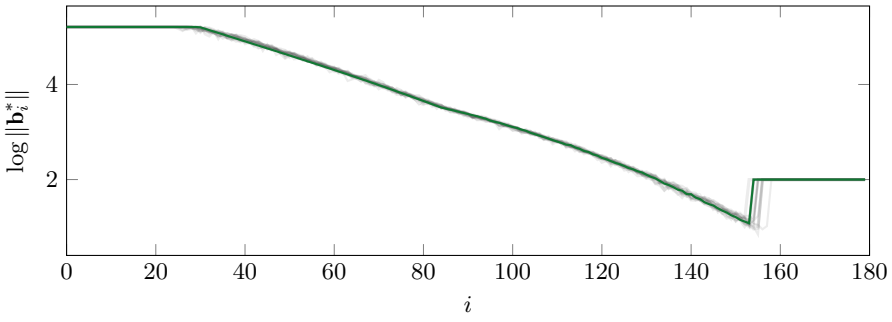


**Fig. 17.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 37, 4)$ compared with 16 runs of BKZ-70 (8 tours) as implemented in FPyLLL.

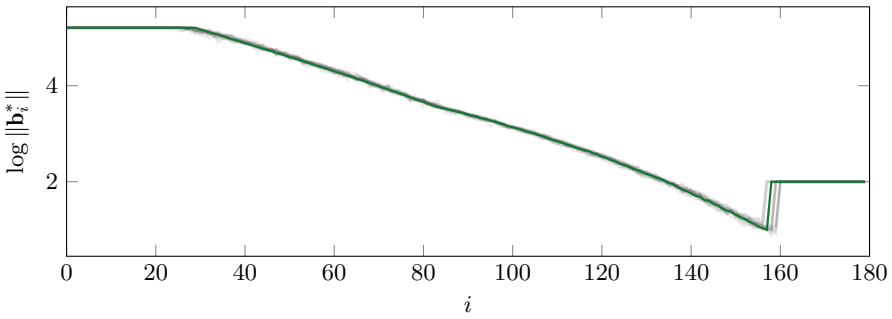


**Fig. 18.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 37, 4)$ compared with 16 runs of BKZ-75 (8 tours) as implemented in FPyLLL.

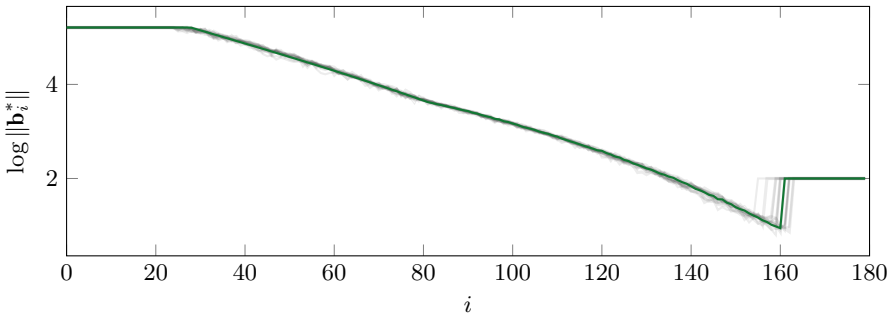


**Fig. 19.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 37, 4)$ compared with 16 runs of BKZ-80 (8 tours) as implemented in FPyLLL.

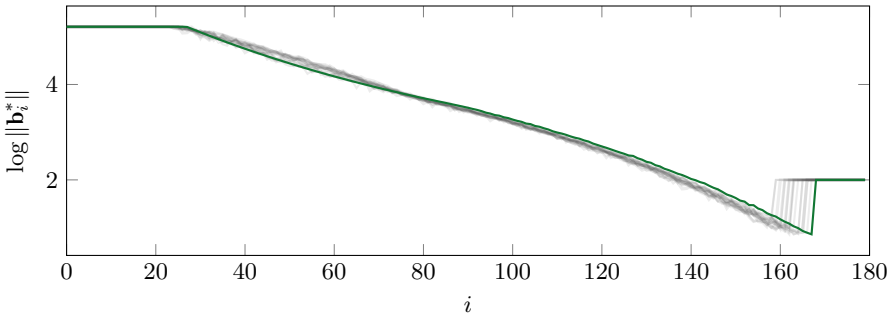


**Fig. 20.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 37, 4)$ compared with 16 runs of BKZ-85 (8 tours) as implemented in FPyLLL.

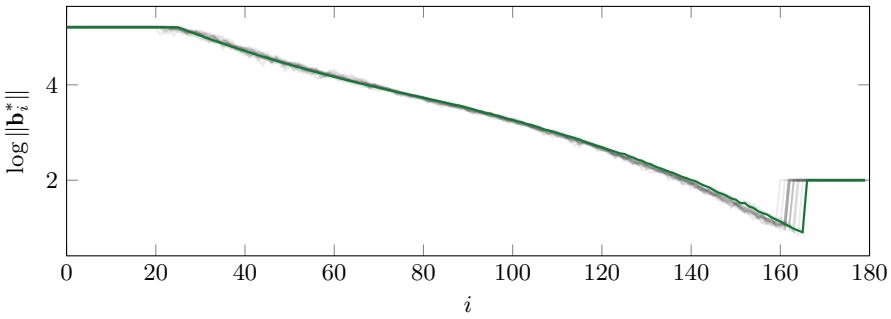


**Fig. 21.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 37, 4)$ compared with 16 runs of BKZ-90 (8 tours) as implemented in FPyLLL.

### B.2 Comparison with G6K

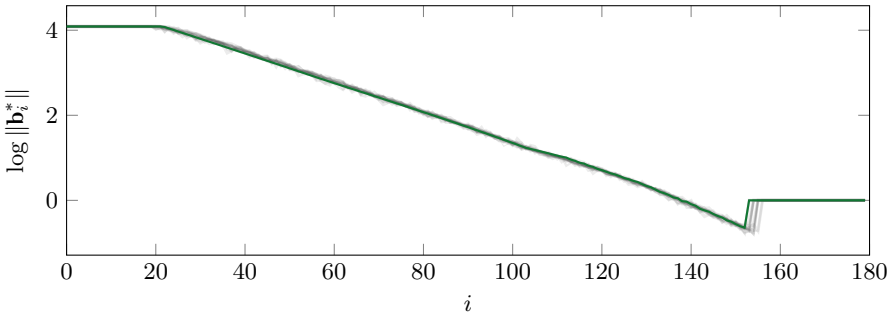


**Fig. 22.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 17, 1)$ compared with 16 runs of BKZ-50 (8 tours) as implemented in G6K.

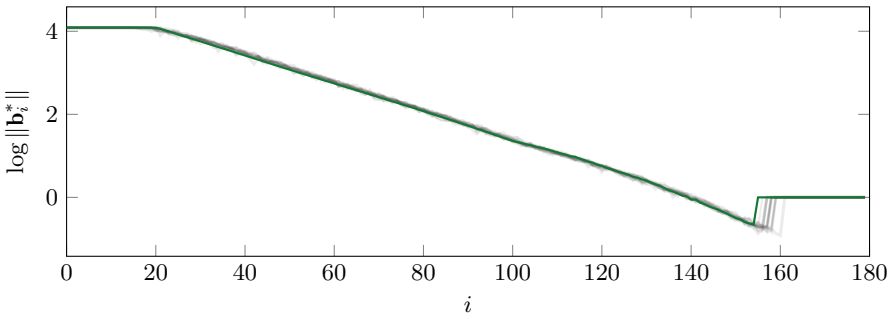


**Fig. 23.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 17, 1)$ compared with 16 runs of BKZ-55 (8 tours) as implemented in G6K.

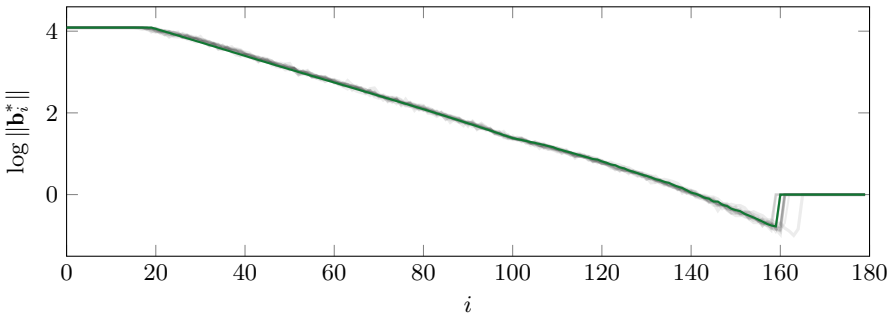


**Fig. 24.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 17, 1)$ compared with 16 runs of BKZ-60 (8 tours) as implemented in G6K.

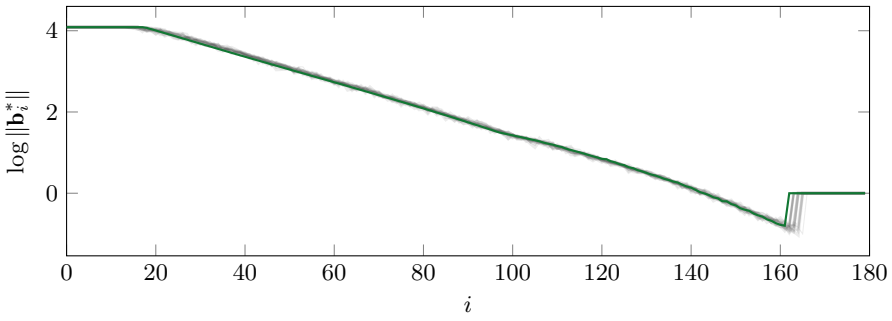


**Fig. 25.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 17, 1)$ compared with 16 runs of BKZ-65 (8 tours) as implemented in G6K.

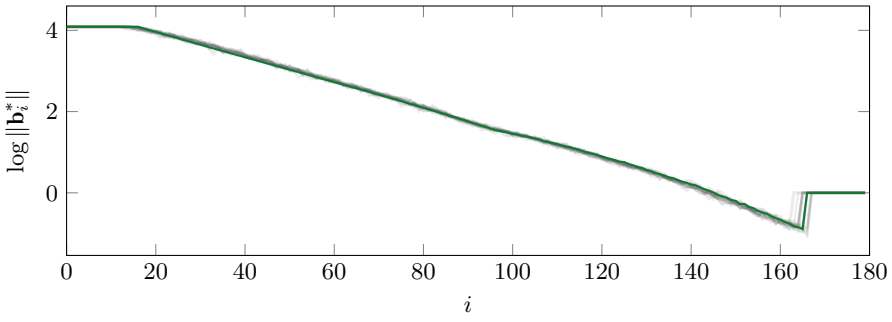


**Fig. 26.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 17, 1)$ compared with 16 runs of BKZ-70 (8 tours) as implemented in G6K.

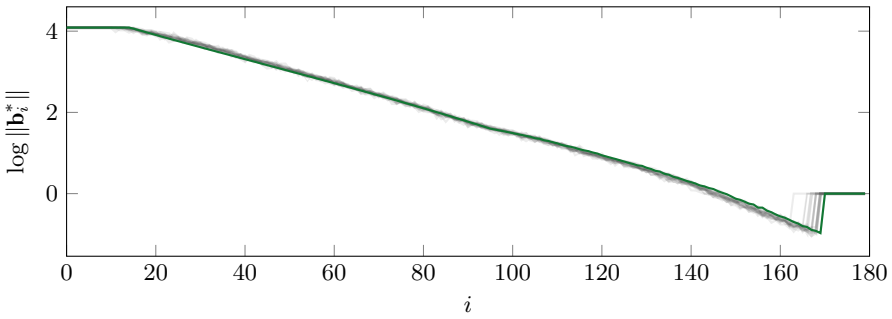


**Fig. 27.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 17, 1)$ compared with 16 runs of BKZ-75 (8 tours) as implemented in G6K.

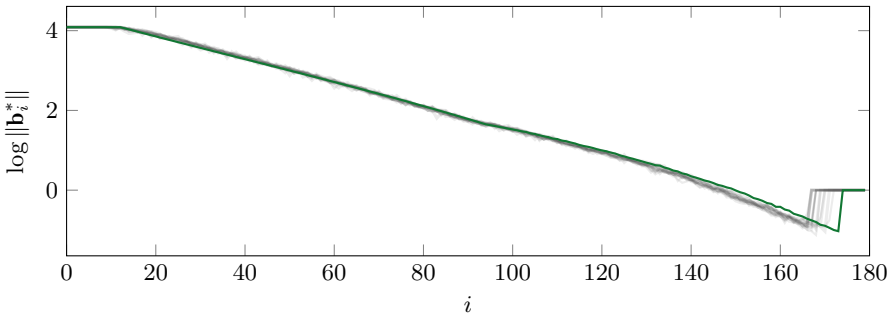


**Fig. 28.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 17, 1)$ compared with 16 runs of BKZ-80 (8 tours) as implemented in G6K.

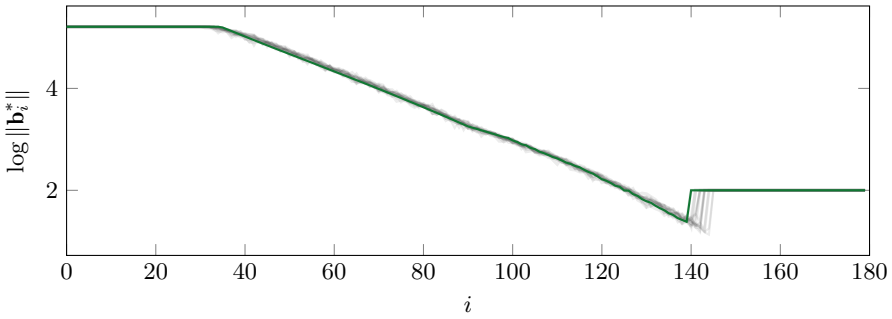


**Fig. 29.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 37, 4)$ compared with 16 runs of BKZ-50 (8 tours) as implemented in G6K.

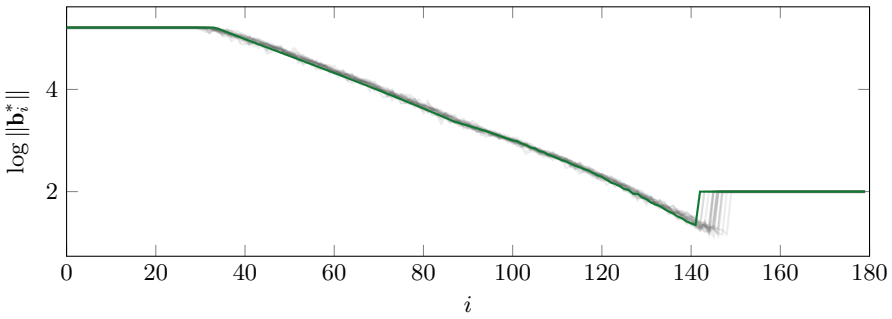


**Fig. 30.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 37, 4)$ compared with 16 runs of BKZ-55 (8 tours) as implemented in G6K.

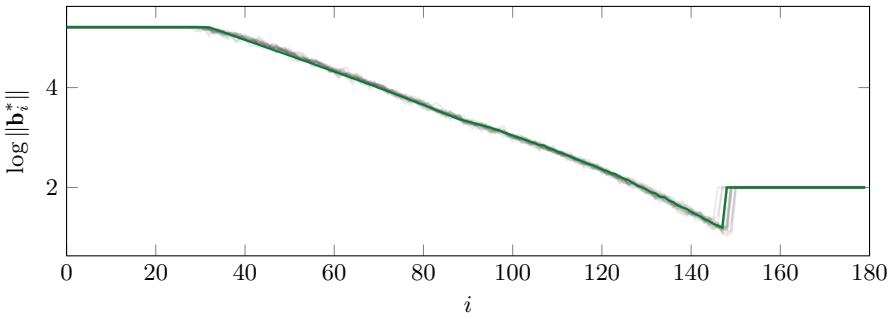


**Fig. 31.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 37, 4)$ compared with 16 runs of BKZ-60 (8 tours) as implemented in G6K.

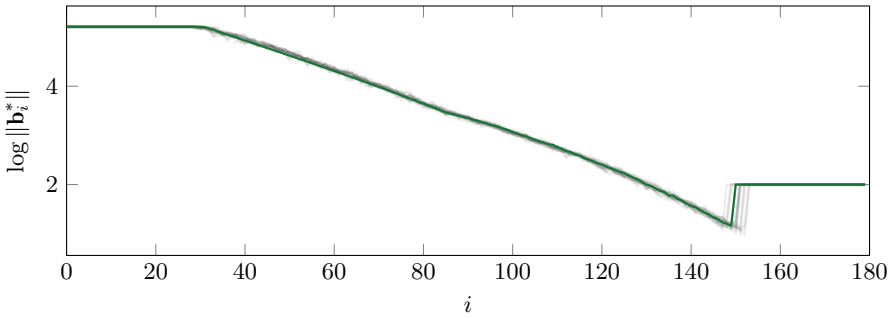


**Fig. 32.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 37, 4)$ compared with 16 runs of BKZ-65 (8 tours) as implemented in G6K.

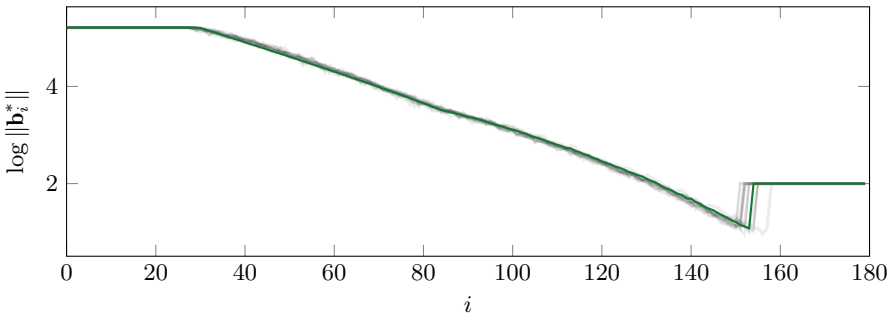


**Fig. 33.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 37, 4)$ compared with 16 runs of BKZ-70 (8 tours) as implemented in G6K.

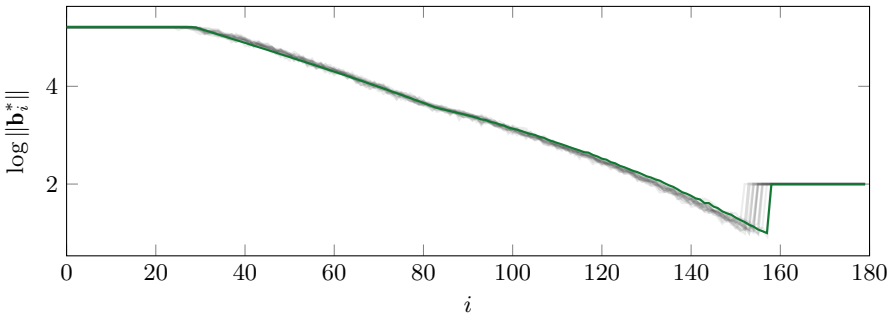


**Fig. 34.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 37, 4)$ compared with 16 runs of BKZ-75 (8 tours) as implemented in G6K.

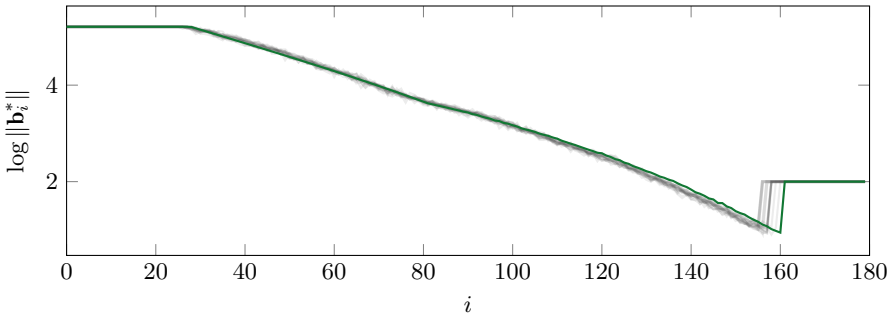


**Fig. 35.** Output of our simulator on $q$-ary bases with parameters $(k, n, q, \xi) = (80, 180, 37, 4)$ compared with 16 runs of BKZ-80 (8 tours) as implemented in G6K.