An LLL algorithm for module lattices

Changmin Lee[1], **Alice Pellet-Mary**[2], Damien Stehlé[1]
and Alexandre Wallet[3]

[1] ENS de Lyon, [2] KU Leuven, [3] NTT Tokyo

Lattices: Geometry, Algorithms and Hardness
February 19, 2020

https://eprint.iacr.org/2019/1035

KU LEUVEN

ENS DE LYON

NTT

What is this talk about?

LLL-type algorithm for modules over a ring of integers

- all number fields
- approx-factor $\approx$ exponential in module rank
- quantum poly-time ... given a CVP oracle depending on K

Structured lattices

Motivation (crypto)

Improve efficiency of lattice-based schemes using structured lattices

Structured lattices

Motivation (crypto)

Improve efficiency of lattice-based schemes using structured lattices

Example: NIST post-quantum standardization process

- 26 remaining candidates (2nd round)
- 12 lattice-based
- 11 using structured lattices

Structured lattices

Motivation (crypto)

Improve efficiency of lattice-based schemes using structured lattices

Example: NIST post-quantum standardization process

- 26 remaining candidates (2nd round)
- 12 lattice-based
- 11 using structured lattices

	Frodo (unstructured lattices)	Kyber (structured lattices)
public key size (in Bytes)	9 616	800
ciphertexts size (in Bytes)	9 720	736

(CCA2 KEMs, round 2 version, security level 1)

Structured lattices

Motivation (crypto)

Improve efficiency of lattice-based schemes using structured lattices

Example: NIST post-quantum standardization process

- 26 remaining candidates (2nd round)
- 12 lattice-based
- 11 using structured lattices

	Frodo (unstructured lattices)	Kyber (structured lattices)
public key size (in Bytes)	9 616	800
ciphertexts size (in Bytes)	9 720	736

(CCA2 KEMs, round 2 version, security level 1)

All 11 schemes use module lattices

Module

- $K = \mathbb{Q}[X]/P(X)$
- $R = \mathbb{Z}[X]/P(X)$ (or $R = O_K$)

with P monic and irreducible, degree d

Module

A (free) module M is a subset of K^k of the form $M = \{B\vec{x} \mid \vec{x} \in R^k\}$, with $B \in K^{k \times k}$ invertible. B is a basis of M; k is the rank of M.

Module

- $K = \mathbb{Q}[X]/P(X)$
- $R = \mathbb{Z}[X]/P(X)$ (or $R = O_K$)

with P monic and irreducible, degree d

Module

A (free) module M is a subset of K^k of the form $M = \{B\vec{x} \mid \vec{x} \in R^k\}$, with $B \in K^{k \times k}$ invertible. B is a basis of M; k is the rank of M.

A module is a 'lattice' of rank k over R.

Module lattice

Reminder:

- $K = \mathbb{Q}[X]/P(X)$ and $R = \mathbb{Z}[X]/P(X)$
- $M = \{B\vec{x} \,:\, \vec{x} \in R^d\}$, with $B = (\vec{b}_1, \cdots, \vec{b}_k)$ linearly independent

Module lattice

Reminder:

- $K = \mathbb{Q}[X]/P(X)$ and $R = \mathbb{Z}[X]/P(X)$
  with $\alpha_1, \cdots, \alpha_d$ roots of P in $\mathbb{C}$
- $M = \{B\vec{x} : \vec{x} \in R^d\}$, with $B = (\vec{b}_1, \cdots, \vec{b}_k)$ linearly independent

Canonical embedding

$$\begin{aligned}\sigma : K &\to \mathbb{C}^d \\ x &\mapsto (x(\alpha_1), \cdots, x(\alpha_d))\end{aligned}$$

Module lattice

Reminder:

- $K = \mathbb{Q}[X]/P(X)$ and $R = \mathbb{Z}[X]/P(X)$
  with $\alpha_1, \cdots, \alpha_d$ roots of P in $\mathbb{C}$
- $M = \{B\vec{x} \,:\, \vec{x} \in R^d\}$, with $B = (\vec{b}_1, \cdots, \vec{b}_k)$ linearly independent

Canonical embedding

$$\begin{aligned} \sigma : K &\to \mathbb{C}^d \\ x &\mapsto (x(\alpha_1), \cdots, x(\alpha_d)) \end{aligned} \qquad \begin{aligned} \sigma : K^k &\to \mathbb{C}^{kd} \\ (x_1, \cdots, x_k) &\mapsto (\sigma(x_1)\| \cdots \|\sigma(x_k)) \end{aligned}$$

Module lattice

Reminder:

- $K = \mathbb{Q}[X]/P(X)$ and $R = \mathbb{Z}[X]/P(X)$
  with $\alpha_1, \cdots, \alpha_d$ roots of P in $\mathbb{C}$
- $M = \{B\vec{x} \,:\, \vec{x} \in R^d\}$, with $B = (\vec{b}_1, \cdots, \vec{b}_k)$ linearly independent

Canonical embedding

$$\begin{aligned} \sigma : K &\to \mathbb{C}^d \\ x &\mapsto (x(\alpha_1), \cdots, x(\alpha_d)) \end{aligned} \qquad \begin{aligned} \sigma : K^k &\to \mathbb{C}^{kd} \\ (x_1, \cdots, x_k) &\mapsto (\sigma(x_1) \| \cdots \| \sigma(x_k)) \end{aligned}$$

$$\boxed{\mathcal{L}(M) = \{\sigma(\vec{x}) \,:\, \vec{x} \in M\} \subset \mathbb{C}^{kd}}$$

$\mathcal{L}(M)$ is a lattice of rank kd, spanned by $(\sigma(\vec{b}_1), \sigma(x\vec{b}_1), \cdots, \sigma(x^{d-1}\vec{b}_k))$

Motivation

Module lattices

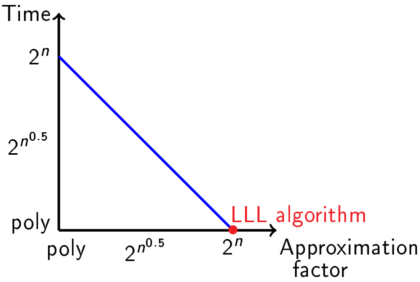
- dimension $n = kd$ over $\mathbb{Z}$
- dimension k over R

Motivation

Module lattices

- dimension $n = kd$ over $\mathbb{Z}$
  typically $500 \leq kd \leq 1500$
- dimension k over R
  typically $k \leq 10$

Motivation



Lattice reduction over $\mathbb{Z}$
(in blue: BKZ trade-offs [Sch87, SE94])

Module lattices

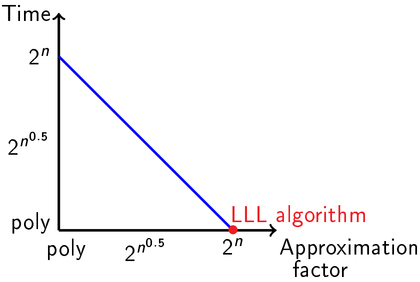
- dimension $n = kd$ over $\mathbb{Z}$
  typically $500 \leq kd \leq 1500$
- dimension k over R
  typically $k \leq 10$

[Sch87] C.-P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. TCS.

[SE94] C.-P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. Mathematical programming.

[LLL82] A. K. Lenstra, H. W. Lenstra, L. Lovász. Factoring polynomials with rational coefficients. Mathematische

Motivation



Lattice reduction over $\mathbb{Z}$
(in blue: BKZ trade-offs [Sch87, SE94])

Module lattices

- dimension $n = kd$ over $\mathbb{Z}$
  typically $500 \leq kd \leq 1500$
- dimension k over R
  typically $k \leq 10$

Can we extend the LLL algorithm to lattices over R?

[Sch87] C.-P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. TCS.

[SE94] C.-P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. Mathematical programming.

[LLL82] A. K. Lenstra, H. W. Lenstra, L. Lovász. Factoring polynomials with rational coefficients. Mathematische

What is small in K?

Remember

$$\begin{aligned}\sigma : K &\to \mathbb{C}^d \\ x &\mapsto (x(\alpha_1), \cdots, x(\alpha_d))\end{aligned}$$

What is small in K?

Remember

$$\begin{aligned} \sigma : K &\to \mathbb{C}^d \\ x &\mapsto (x(\alpha_1), \cdots, x(\alpha_d)) \end{aligned}$$

Two generalization of $|\cdot|$:

- $\|x\| := \|\sigma(x)\|$ (Euclidean norm)
- $\mathcal{N}(x) := \prod_i |x(\alpha_i)|$ (algebraic norm)

What is small in K?

Remember

$$\begin{aligned} \sigma : K &\to \mathbb{C}^d \\ x &\mapsto (x(\alpha_1), \cdots, x(\alpha_d)) \end{aligned}$$

Two generalization of $|\cdot|$:

- $\|x\| := \|\sigma(x)\|$ (Euclidean norm)
- $\mathcal{N}(x) := \prod_i |x(\alpha_i)|$ (algebraic norm)

$$\|x\| \text{ small } \overset{\Rightarrow}{\underset{\not\Leftarrow}{}} \mathcal{N}(x) \text{ small}$$

Previous works and result

	Bound on quality	Bound on runtime
Napias [Nap96]: ▸ specific number fields	✗	✗

[Nap96] H. Napias. A generalization of the LLL-algorithm over Euclidean rings or orders. Journal de théorie des nombres de Bordeaux.

Previous works and result

	Bound on quality	Bound on runtime
Napias [Nap96]: ▶ specific number fields	✗	✗
Fieker-Pohst [FP96]: ▶ all number fields	✗	✗
▶ totally real fields	✗	✓

[FP96] C. Fieker, M. E. Pohst. Lattices over number fields. ANTS.

Previous works and result

	Bound on quality	Bound on runtime
Napias [Nap96]:		
▸ specific number fields	✗	✗
Fieker-Pohst [FP96]:		
▸ all number fields	✗	✗
▸ totally real fields	✗	✓
Kim-Lee [KL17]:		
▸ norm Euclidean fields	✗	✓
▸ biquadratic norm euclidean	✓	✓

[KL17] T. Kim, C. Lee. Lattice reductions over euclidean rings with applications to cryptanalysis. IMACC.

Previous works and result

	Bound on quality	Bound on runtime
Napias [Nap96]:		
▸ specific number fields	✗	✗
Fieker-Pohst [FP96]:		
▸ all number fields	✗	✗
▸ totally real fields	✗	✓
Kim-Lee [KL17]:		
▸ norm Euclidean fields	✗	✓
▸ biquadratic norm euclidean	✓	✓
This work [LPSW19]:		
▸ any number field	✓	$\approx$

[LPSW19] C. Lee, A. Pellet-Mary, D. Stehlé, A. Wallet. An LLL algorithm for module lattices.

Previous works and result

	Bound on quality	Bound on runtime
Napias [Nap96]:		
▸ specific number fields	✗	✗
Fieker-Pohst [FP96]:		
▸ all number fields	✗	✗
▸ totally real fields	✗	✓
Kim-Lee [KL17]:		
▸ norm Euclidean fields	✗	✓
▸ biquadratic norm euclidean	✓	✓
This work [LPSW19]:		
▸ any number field	✓	$\approx$ (✓ if we have an oracle for CVP in a fixed lattice depending only on R)

[LPSW19] C. Lee, A. Pellet-Mary, D. Stehlé, A. Wallet. An LLL algorithm for module lattices.

Outline of the talk

1. The LLL algorithm
2. The Lagrange-Gauss algorithm
3. Computing the relaxed Euclidean division

Outline of the talk

1. The LLL algorithm

2. The Lagrange-Gauss algorithm

3. Computing the relaxed Euclidean division

High-level overview of LLL

LLL over $\mathbb{Z}$

- γ'-SVP in dim k
  $\leq$ 1-SVP in dim 2
  - $\gamma' = 2^{O(k)}$
  - poly time

[LLL82] A. K. Lenstra, H. W. Lenstra, L. Lovász. Factoring polynomials with rational coefficients. Mathematische Annalen.

High-level overview of LLL

LLL over $\mathbb{Z}$

- γ'-SVP in dim k
  $\leq$ 1-SVP in dim 2
  - $\gamma' = 2^{O(k)}$
  - poly time
- Lagrange-Gauss algo
  for SVP in dim 2
  - poly time

[LLL82] A. K. Lenstra, H. W. Lenstra, L. Lovász. Factoring polynomials with rational coefficients. Mathematische Annalen.

High-level overview of LLL

LLL over $\mathbb{Z}$

- γ'-SVP in dim k
  $\leq$ 1-SVP in dim 2
  - $\gamma' = 2^{O(k)}$
  - poly time
- Lagrange-Gauss algo for SVP in dim 2
  - poly time

LLL over $R = \mathbb{Z}[X]/(X^d + 1)$

- γ'-SVP in rank-k
  $\leq \gamma$-SVP in rank-2
  - $\gamma' = (\gamma d)^{O(k)}$
  - poly time

[LLL82] A. K. Lenstra, H. W. Lenstra, L. Lovász. Factoring polynomials with rational coefficients. Mathematische Annalen.

High-level overview of LLL

LLL over $\mathbb{Z}$

- γ'-SVP in dim k $\leq$ 1-SVP in dim 2
  - $\gamma' = 2^{O(k)}$
  - poly time
- Lagrange-Gauss algo for SVP in dim 2
  - poly time

LLL over $R = \mathbb{Z}[X]/(X^d + 1)$

- γ'-SVP in rank-k $\leq$ γ-SVP in rank-2
  - $\gamma' = (\gamma d)^{O(k)}$
  - poly time
- Algorithm for γ-SVP in rank-2
  - $\gamma = 2^{(\log d)^{O(1)}}$
  - heuristic, quantum
  - poly time if oracle solving CVP in a fixed lattice (depending only on R)

[LLL82] A. K. Lenstra, H. W. Lenstra, L. Lovász. Factoring polynomials with rational coefficients. Mathematische Annalen.

High-level overview of LLL

LLL over $\mathbb{Z}$

- γ'-SVP in dim k $\leq$ 1-SVP in dim 2
  - $\gamma' = 2^{O(k)}$
  - poly time
- Lagrange-Gauss algo for SVP in dim 2
  - poly time

LLL over $R = \mathbb{Z}[X]/(X^d + 1)$

- γ'-SVP in rank-k
  $\leq \gamma$-SVP i
  -
  - time

needs QR-factorisation

- Algorithm for γ-SVP in rank-2
  - $\gamma = 2^{(\log d)^{O(1)}}$
  - heu
  - poly oracle solving CVP in a fixed lattice (depending only on R)

next section

[LLL82] A. K. Lenstra, H. W. Lenstra, L. Lovász. Factoring polynomials with rational coefficients. Mathematische Annalen.

Inner product over R

For $\vec{a} = (a_1, \cdots, a_k) \in K^k$ and $\vec{b} = (b_1, \cdots, b_k) \in K^k$,

$$\boxed{\langle \vec{a}, \vec{b} \rangle_K = \sum_i a_i \overline{b_i} \in K \quad (\text{or } K_{\mathbb{R}})}$$

Inner product over R

For $\vec{a} = (a_1, \cdots, a_k) \in K^k$ and $\vec{b} = (b_1, \cdots, b_k) \in K^k$,

$$\boxed{\langle \vec{a}, \vec{b} \rangle_K = \sum_i a_i \overline{b_i} \in K \quad (\text{or } K_{\mathbb{R}})}$$

$\|\vec{a}\|_K := \sqrt{\langle \vec{a}, \vec{a} \rangle_K} \in K$ (or $K_{\mathbb{R}}$)

Inner product over R

For $\vec{a} = (a_1, \cdots, a_k) \in K^k$ and $\vec{b} = (b_1, \cdots, b_k) \in K^k$,

$$\boxed{\langle \vec{a}, \vec{b} \rangle_K = \sum_i a_i \overline{b_i} \in K \quad (\text{or } K_{\mathbb{R}})}$$

$\|\vec{a}\|_K := \sqrt{\langle \vec{a}, \vec{a} \rangle_K} \in K$ (or $K_{\mathbb{R}}$)

Properties

- $\| \, \|\vec{a}\|_K \, \| = \|\sigma(\vec{a})\|$

$\|x\| = \|\sigma(x)\|$

Inner product over R

For $\vec{a} = (a_1, \cdots, a_k) \in K^k$ and $\vec{b} = (b_1, \cdots, b_k) \in K^k$,

$$\boxed{\langle \vec{a}, \vec{b} \rangle_K = \sum_i a_i \overline{b_i} \in K \quad (\text{or } K_{\mathbb{R}})}$$

$\|\vec{a}\|_K := \sqrt{\langle \vec{a}, \vec{a} \rangle_K} \in K$ (or $K_{\mathbb{R}}$)

Properties

- $\| \; \|\vec{a}\|_K \; \| = \|\sigma(\vec{a})\|$
- $\mathcal{N}(\|\vec{a}\|_K) = \Delta_K^{-1/2} \cdot \det(\mathcal{L}(\vec{a}))$

$\mathcal{N}(x) = \prod_{i=1}^d |\sigma(x)_i|$

QR factorization over R

Let $B = (\vec{b}_1, \cdots, \vec{b}_k) \in K^k$, define

$$\vec{b}_i^* = \vec{b}_i - \sum_{j<i} \mu_{ij} \vec{b}_j^*, \quad \text{with } \mu_{ij} = \frac{\langle \vec{b}_i, \vec{b}_j^* \rangle_K}{\langle \vec{b}_j^*, \vec{b}_j^* \rangle_K}$$

QR factorization over R

Let $B = (\vec{b}_1, \cdots, \vec{b}_k) \in K^k$, define

$$\vec{b}_i^* = \vec{b}_i - \sum_{j<i} \mu_{ij} \vec{b}_j^*, \quad \text{with } \mu_{ij} = \frac{\langle \vec{b}_i, \vec{b}_j^* \rangle_K}{\langle \vec{b}_j^*, \vec{b}_j^* \rangle_K}$$

QR-factorisation: $B = QR$, with

- $r_{ii} = \|\vec{b}_i^*\|_K$, $r_{ij} = \mu_{ji} r_{ii}$ for $i < j$ and $r_{ij} = 0$ otherwise
- columns of Q are $\vec{b}_i^* / \|\vec{b}_i^*\|_K$

QR factorization over R

Let $B = (\vec{b}_1, \cdots, \vec{b}_k) \in K^k$, define

$$\vec{b}_i^* = \vec{b}_i - \sum_{j<i} \mu_{ij} \vec{b}_j^*, \text{ with } \mu_{ij} = \frac{\langle \vec{b}_i, \vec{b}_j^* \rangle_K}{\langle \vec{b}_j^*, \vec{b}_j^* \rangle_K}$$

QR-factorisation: $B = QR$, with

- $r_{ii} = \|\vec{b}_i^*\|_K$, $r_{ij} = \mu_{ji} r_{ii}$ for $i < j$ and $r_{ij} = 0$ otherwise
- columns of Q are $\vec{b}_i^* / \|\vec{b}_i^*\|_K$

Properties

- $\forall \vec{x}, \vec{y}$, $\boxed{\langle B\vec{x}, B\vec{y} \rangle_K = \langle R\vec{x}, R\vec{y} \rangle_K}$

QR factorization over R

Let $B = (\vec{b}_1, \cdots, \vec{b}_k) \in K^k$, define

$$\vec{b}_i^* = \vec{b}_i - \sum_{j<i} \mu_{ij} \vec{b}_j^*, \ \text{ with } \mu_{ij} = \frac{\langle \vec{b}_i, \vec{b}_j^* \rangle_K}{\langle \vec{b}_j^*, \vec{b}_j^* \rangle_K}$$

QR-factorisation: $B = QR$, with

- $r_{ii} = \|\vec{b}_i^*\|_K$, $r_{ij} = \mu_{ji} r_{ii}$ for $i < j$ and $r_{ij} = 0$ otherwise
- columns of Q are $\vec{b}_i^* / \|\vec{b}_i^*\|_K$

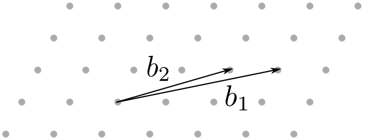
Properties

- $\forall \vec{x}, \vec{y}$, $\boxed{\langle B\vec{x}, B\vec{y} \rangle_K = \langle R\vec{x}, R\vec{y} \rangle_K}$
- $\forall \vec{v} \in \mathcal{L}(B)$, $\mathcal{N}(\|\vec{v}\|_K) \geq \min \mathcal{N}(r_{ii})$

Outline of the talk

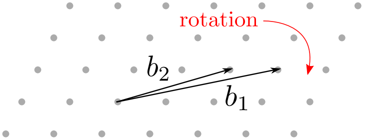
1. The LLL algorithm
2. The Lagrange-Gauss algorithm
3. Computing the relaxed Euclidean division

Lagrange-Gauss algorithm (over $\mathbb{Z}$)



$$M = \begin{pmatrix} 10 & 7 \\ 2 & 2 \end{pmatrix}$$

Lagrange-Gauss algorithm (over $\mathbb{Z}$)

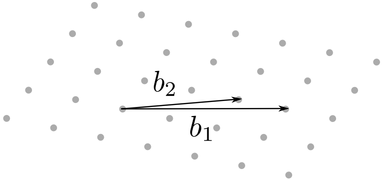


rotation

$$M = \begin{pmatrix} 10 & 7 \\ 2 & 2 \end{pmatrix}$$

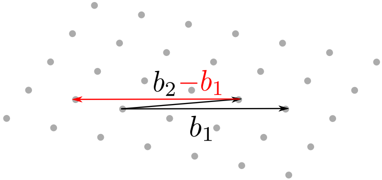
Compute QR factorization

Lagrange-Gauss algorithm (over $\mathbb{Z}$)



$$M = \begin{pmatrix} 10.2 & 7.3 \\ 0 & 0.6 \end{pmatrix}$$

Lagrange-Gauss algorithm (over $\mathbb{Z}$)

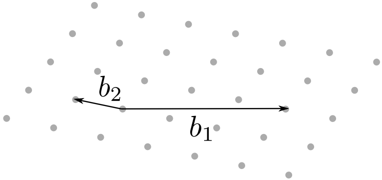


reduce b_2 with b_1

$$M = \begin{pmatrix} 10.2 & 7.3 \\ 0 & 0.6 \end{pmatrix}$$

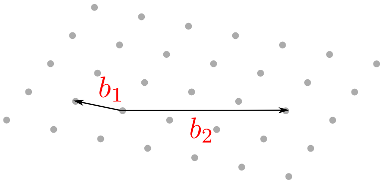
"Euclidean division" (over $\mathbb{R}$) of 7.3 by 10.2

Lagrange-Gauss algorithm (over $\mathbb{Z}$)



$$M = \begin{pmatrix} 10.2 & -2.9 \\ 0 & 0.6 \end{pmatrix}$$

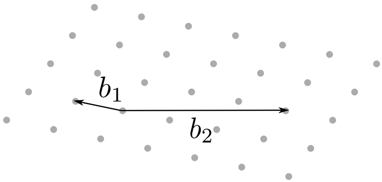
Lagrange-Gauss algorithm (over $\mathbb{Z}$)



swap

$$M = \begin{pmatrix} -2.9 & 10.2 \\ 0.6 & 0 \end{pmatrix}$$

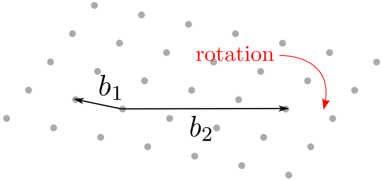
Lagrange-Gauss algorithm (over $\mathbb{Z}$)



start again

$$M = \begin{pmatrix} -2.9 & 10.2 \\ 0.6 & 0 \end{pmatrix}$$

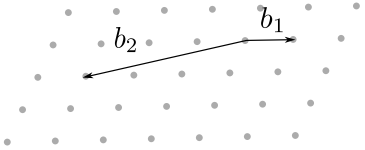
Lagrange-Gauss algorithm (over $\mathbb{Z}$)



rotation

$$M = \begin{pmatrix} -2.9 & 10.2 \\ 0.6 & 0 \end{pmatrix}$$

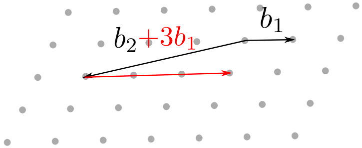
Lagrange-Gauss algorithm (over $\mathbb{Z}$)



rotation

$$M = \begin{pmatrix} 3 & -10 \\ 0 & -2 \end{pmatrix}$$

Lagrange-Gauss algorithm (over $\mathbb{Z}$)

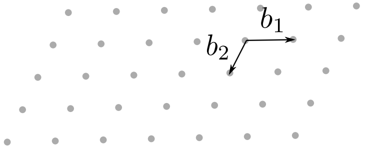


reduce b_2 with b_1

$$M = \begin{pmatrix} 3 & -10 \\ 0 & -2 \end{pmatrix}$$

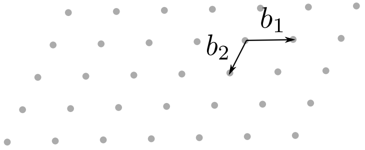
"Euclidean division" (over $\mathbb{R}$) of -10 by 3

Lagrange-Gauss algorithm (over $\mathbb{Z}$)



$$M = \begin{pmatrix} 3 & -1 \\ 0 & -2 \end{pmatrix}$$

Lagrange-Gauss algorithm (over $\mathbb{Z}$)

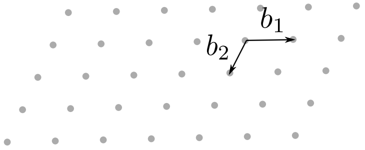


$$M = \begin{pmatrix} 3 & -1 \\ 0 & -2 \end{pmatrix}$$

For the Lagrange-Gauss algorithm over R, we need

- Rotation
- Euclidean division

Lagrange-Gauss algorithm (over $\mathbb{Z}$)



$$M = \begin{pmatrix} 3 & -1 \\ 0 & -2 \end{pmatrix}$$

For the Lagrange-Gauss algorithm over R, we need

- Rotation $\Rightarrow$ ok
- Euclidean division $\Rightarrow$?

Euclidean division

Over $\mathbb{Z}$

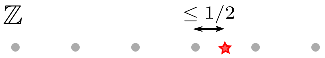
Input: $a, b \in \mathbb{Z}$, $a \neq 0$
Output: $r \in \mathbb{Z}$
such that $|b + ra| \leq |a|/2$

Euclidean division

Over $\mathbb{Z}$

Input: $a, b \in \mathbb{Z}$, $a \neq 0$
Output: $r \in \mathbb{Z}$
such that $|b + ra| \leq |a|/2$

CVP in $\mathbb{Z}$ with target $-b/a$.

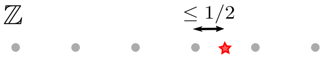


Euclidean division

Over $\mathbb{Z}$

Input: $a, b \in \mathbb{Z}$, $a \neq 0$
Output: $r \in \mathbb{Z}$
such that $|b + ra| \leq |a|/2$

CVP in $\mathbb{Z}$ with target $-b/a$.



Over R

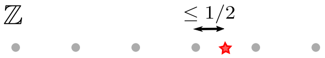
CVP in R with target $-b/a$
$\Rightarrow$ output $r \in R$

Euclidean division

Over $\mathbb{Z}$

Input: $a, b \in \mathbb{Z}$, $a \neq 0$
Output: $r \in \mathbb{Z}$
such that $|b + ra| \leq |a|/2$

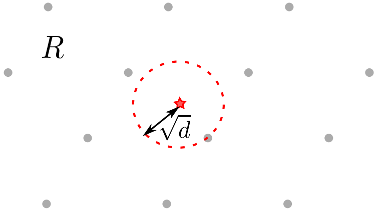
CVP in $\mathbb{Z}$ with target $-b/a$.



Over R

CVP in R with target $-b/a$
$\Rightarrow$ output $r \in R$

Difficulty: Typically
$\|b + ra\| \approx \sqrt{d} \cdot \|a\| \gg \|a\|$.



Euclidean division

Over $\mathbb{Z}$

Input: $a, b \in \mathbb{Z}$, $a \neq 0$
Output: $r \in \mathbb{Z}$
such that $|b + ra| \leq |a|/2$

CVP in $\mathbb{Z}$ with target $-b/a$.

Over R

CVP in R with target $-b/a$
$\Rightarrow$ output $r \in R$

Difficulty: Typically
$\|b + ra\| \approx \sqrt{d} \cdot \|a\| \gg \|a\|$.

Relax the requirement

Find $x, y \in R$ such that

- $\|xa + yb\| \leq \|a\|/2$
- $\|y\| \leq \mathrm{poly}(d)$

Euclidean division

Over $\mathbb{Z}$

Input: $a, b \in \mathbb{Z}$, $a \neq 0$
Output: $r \in \mathbb{Z}$
such that $|b + ra| \leq |a|/2$

CVP in $\mathbb{Z}$ with target $-b/a$.

Over R

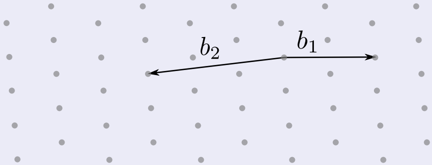
CVP in R with target $-b/a$
$\Rightarrow$ output $r \in R$

Difficulty: Typically
$\|b + ra\| \approx \sqrt{d} \cdot \|a\| \gg \|a\|$.

Relax the requirement

Find $x, y \in R$ such that

- $\|xa + yb\| \leq \|a\|/2$
- $\|y\| \leq \mathrm{poly}(d)$



Euclidean division

Over $\mathbb{Z}$

Input: $a, b \in \mathbb{Z}$, $a \neq 0$
Output: $r \in \mathbb{Z}$
such that $|b + ra| \leq |a|/2$

CVP in $\mathbb{Z}$ with target $-b/a$.

Over R

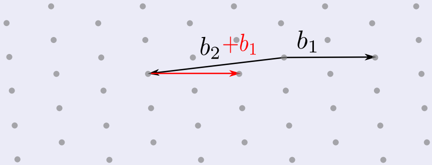
CVP in R with target $-b/a$
$\Rightarrow$ output $r \in R$

Difficulty: Typically
$\|b + ra\| \approx \sqrt{d} \cdot \|a\| \gg \|a\|$.

Relax the requirement

Find $x, y \in R$ such that

- $\|xa + yb\| \leq \|a\|/2$
- $\|y\| \leq \mathrm{poly}(d)$



Euclidean division

Over $\mathbb{Z}$

Input: $a, b \in \mathbb{Z}$, $a \neq 0$
Output: $r \in \mathbb{Z}$
such that $|b + ra| \leq |a|/2$

CVP in $\mathbb{Z}$ with target $-b/a$.

Over R

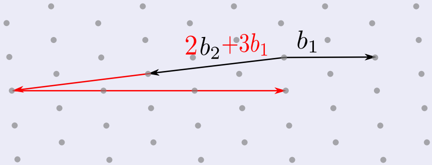
CVP in R with target $-b/a$
$\Rightarrow$ output $r \in R$

Difficulty: Typically
$\|b + ra\| \approx \sqrt{d} \cdot \|a\| \gg \|a\|$.

Relax the requirement

Find $x, y \in R$ such that

- $\|xa + yb\| \leq \|a\|/2$
- $\|y\| \leq \mathrm{poly}(d)$



Euclidean division

Over $\mathbb{Z}$

Input: $a, b \in \mathbb{Z}$, $a \neq 0$
Output: $r \in \mathbb{Z}$
such that $|b + ra| \leq |a|/2$

CVP in $\mathbb{Z}$ with target $-b/a$.

Over R

CVP in R with target $-b/a$
$\Rightarrow$ output $r \in R$

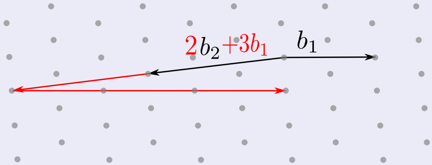
Difficulty: Typically
$\|b + ra\| \approx \sqrt{d} \cdot \|a\| \gg \|a\|$.

Relax the requirement

Find $x, y \in R$ such that

- $\|xa + yb\| \leq \|a\|/2$
- $\|y\| \leq \mathrm{poly}(d)$

$\Rightarrow$ sufficient for Gauss' algo



Outline of the talk

1. The LLL algorithm
2. The Lagrange-Gauss algorithm
3. Computing the relaxed Euclidean division

The Log space

Reminder: $\sigma(r) = (r(\alpha_1), \cdots, r(\alpha_d))^T$

(multiplication is coefficient-wise)

The Log space

Reminder: $\sigma(r) = (r(\alpha_1), \cdots, r(\alpha_d))^T$
(multiplication is coefficient-wise)

$$\boxed{\mathrm{Log}(r) = (\log|r(\alpha_1)|, \cdots, \log|r(\alpha_d)|)^T}$$

The Log space

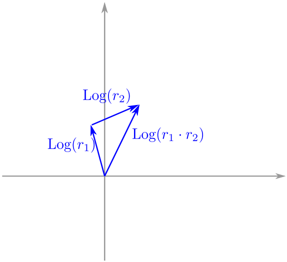
Reminder: $\sigma(r) = (r(\alpha_1), \cdots, r(\alpha_d))^T$

(multiplication is coefficient-wise)

$$\boxed{\mathrm{Log}(r) = (\log|r(\alpha_1)|, \cdots, \log|r(\alpha_d)|)^T}$$

Properties of Log

- $\mathrm{Log}(r_1 \cdot r_2) = \mathrm{Log}(r_1) + \mathrm{Log}(r_2)$



The Log space

Reminder: $\sigma(r) = (r(\alpha_1), \cdots, r(\alpha_d))^T$

(multiplication is coefficient-wise)

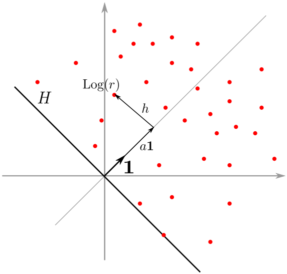
$$\boxed{\mathrm{Log}(r) = (\log|r(\alpha_1)|, \cdots, \log|r(\alpha_d)|)^T}$$

Let $\mathbf{1} = (1, \cdots, 1)$ and $H = \mathbf{1}^\perp$

Properties of Log

$\mathrm{Log}\, r = h + a\mathbf{1}$, with $h \in H$

- $\mathrm{Log}(r_1 \cdot r_2) = \mathrm{Log}(r_1) + \mathrm{Log}(r_2)$
- $a \geq 0$ if $r \in R$



The Log space

Reminder: $\sigma(r) = (r(\alpha_1), \cdots, r(\alpha_d))^T$
(multiplication is coefficient-wise)

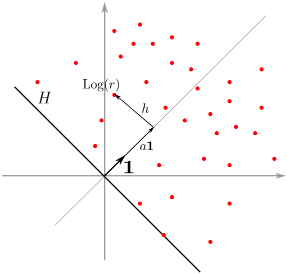
$$\boxed{\mathrm{Log}(r) = (\log|r(\alpha_1)|, \cdots, \log|r(\alpha_d)|)^T}$$

Let $\mathbf{1} = (1, \cdots, 1)$ and $H = \mathbf{1}^\perp$

Properties of Log

$\mathrm{Log}\, r = h + a\mathbf{1}$, with $h \in H$

- $\mathrm{Log}(r_1 \cdot r_2) = \mathrm{Log}(r_1) + \mathrm{Log}(r_2)$
- $a \geq 0$ if $r \in R$
- $\|r\| \simeq 2^{\|\mathrm{Log}\, r\|_\infty}$



Using the Log space

Objective: find $x, y \in R$ such that

- $\|xa + yb\| \leq \|a\|/2$
- $\|y\| \leq \mathrm{poly}(d)$

Using the Log space

Objective: find $x, y \in R$ such that

- $\|xa + yb\| \leq \|a\|/2$
- $\|y\| \leq \mathrm{poly}(d)$

Difficulty: Log works well with $\times$, but not with $+$

Using the Log space

Objective: find $x, y \in R$ such that

- $\|xa - yb\| \leq \|a\|/2$
- $\|y\| \leq \mathrm{poly}(d)$

Difficulty: Log works well with $\times$, but not with $+$

Using the Log space

Objective: find $x, y \in R$ such that

- $\|xa - yb\| \leq \|a\|/2$
- $\|y\| \leq \mathrm{poly}(d)$

Difficulty: Log works well with $\times$, but not with $+$

Solution: If $\|\operatorname{Log}(u) - \operatorname{Log}(v)\| \leq \varepsilon$
then $\|u - v\| \lesssim \varepsilon \cdot \min(\|u\|, \|v\|)$
(requires to extend Log to take arguments into account)

Extended logarithm

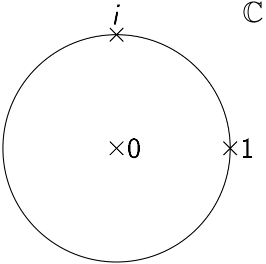
Let's say $d = 1$

$$\mathrm{Log}(x) = \log |x(\alpha_1)| \qquad (\alpha_1 \in \mathbb{C})$$

Extended logarithm

Let's say $d = 1$

$$\mathrm{Log}(x) = \log |x(\alpha_1)| \qquad (\alpha_1 \in \mathbb{C})$$

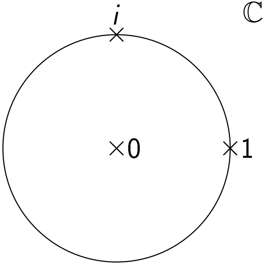


$$\mathrm{Log}\, 1 = \mathrm{Log}\, i$$

Extended logarithm

Let's say $d = 1$

$$\mathrm{Log}(x) = \log|x(\alpha_1)| \qquad (\alpha_1 \in \mathbb{C})$$



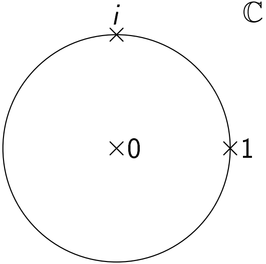
$$\mathrm{Log}\, 1 = \mathrm{Log}\, i$$

$$\boxed{\begin{array}{l} |\mathrm{Log}(u) - \mathrm{Log}(v)| \leq \varepsilon \\ \not\Rightarrow |u - v| \leq \varepsilon' \end{array}}$$

Extended logarithm

Let's say $d = 1$

$$\overline{\mathrm{Log}}(x) = (\log r, \theta) \in \mathbb{R} \times \mathbb{Z}/(2\pi\mathbb{Z}) \qquad x(\alpha_1) = r \cdot e^{i\theta} \in \mathbb{C}$$



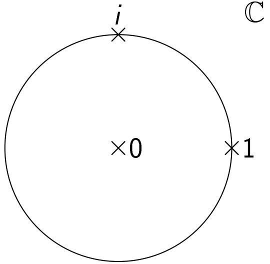
$$\overline{\mathrm{Log}}\, 1 \neq \overline{\mathrm{Log}}\, i$$

$$\boxed{\begin{array}{l} |\,\overline{\mathrm{Log}}(u) - \overline{\mathrm{Log}}(v)| \leq \varepsilon \\ \Rightarrow |u - v| \leq \varepsilon' \end{array}}$$

Extended logarithm

Let's say $d = 1$

$$\overline{\mathrm{Log}}(x) = (\log r, \theta) \in \mathbb{R} \times \mathbb{Z}/(2\pi\mathbb{Z}) \qquad x(\alpha_1) = r \cdot e^{i\theta} \in \mathbb{C}$$



$$\overline{\mathrm{Log}}\, 1 \neq \overline{\mathrm{Log}}\, i$$

$$\boxed{\begin{array}{l} |\overline{\mathrm{Log}}(u) - \overline{\mathrm{Log}}(v)| \leq \varepsilon \\ \Rightarrow |u - v| \lesssim \varepsilon \cdot \min(\|u\|, \|v\|) \end{array}}$$

Using the Log space

Objective: find $x, y \in R$ such that

- $\|xa - yb\| \leq \|a\|/2$
- $\|y\| \leq \mathrm{poly}(d)$

Difficulty: Log works well with $\times$, but not with $+$

Solution: If $\| \mathrm{Log}(u) - \mathrm{Log}(v)\| \leq \varepsilon$
then $\|u - v\| \lesssim \varepsilon \cdot \min(\|u\|, \|v\|)$
(requires to extend Log to take arguments into account)

Using the Log space

Objective: find $x, y \in R$ such that

- $\|xa - yb\| \leq \|a\|/2$
- $\|y\| \leq \mathrm{poly}(d)$

Difficulty: Log works well with $\times$, but not with $+$

Solution: If $\|\operatorname{Log}(u) - \operatorname{Log}(v)\| \leq \varepsilon$
then $\|u - v\| \lesssim \varepsilon \cdot \min(\|u\|, \|v\|)$
(requires to extend Log to take arguments into account)

New objective

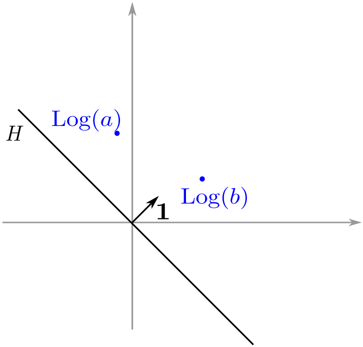
Find $x, y \in R$ such that

- $\|\operatorname{Log}(xa) - \operatorname{Log}(yb)\| \leq \varepsilon$
- $\|\operatorname{Log}(y)\|_\infty \leq O(\log d)$

Idea of the algorithm

Objective: find $x, y \in R$ s.t.

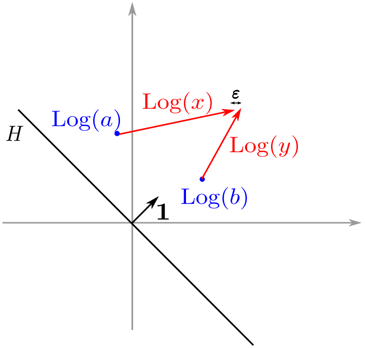
- $\| \mathrm{Log}(xa) - \mathrm{Log}(yb) \| \leq \varepsilon$
- $\| \mathrm{Log}(y) \|_\infty \leq O(\log d)$



Idea of the algorithm

Objective: find $x, y \in R$ s.t.

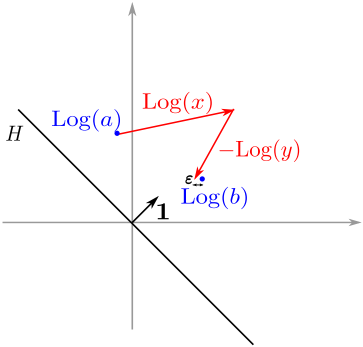
- $\| \mathrm{Log}(xa) - \mathrm{Log}(yb) \| \leq \varepsilon$
- $\| \mathrm{Log}(y) \|_\infty \leq O(\log d)$



Idea of the algorithm

Objective: find $x, y \in R$ s.t.

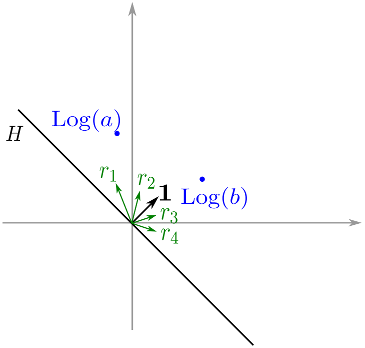
- $\|(\mathrm{Log}(x) - \mathrm{Log}(y)) - \mathrm{Log}(b/a)\| \leq \varepsilon$
- $\|\mathrm{Log}(y)\|_\infty \leq O(\log d)$



Idea of the algorithm

Objective: find $x, y \in R$ s.t.

- $\|(\mathrm{Log}(x) - \mathrm{Log}(y)) - \mathrm{Log}(b/a)\| \leq \varepsilon$
- $\|\mathrm{Log}(y)\|_\infty \leq O(\log d)$

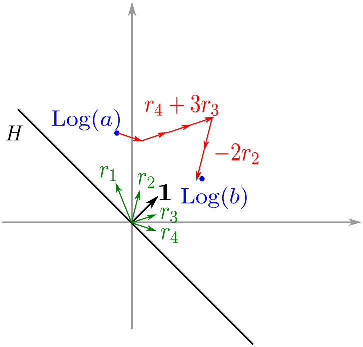


$$r_i = \mathrm{Log}(x_i)\,,\ \ x_i \in R$$

Idea of the algorithm

Objective: find $x, y \in R$ s.t.

- $\|(\mathrm{Log}(x) - \mathrm{Log}(y)) - \mathrm{Log}(b/a)\| \leq \varepsilon$
- $\|\mathrm{Log}(y)\|_\infty \leq O(\log d)$

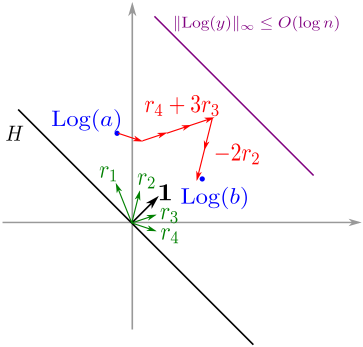


$$r_i = \mathrm{Log}(x_i)\,,\ \ x_i \in R$$

Idea of the algorithm

Objective: find $x, y \in R$ s.t.

- $\|(\mathrm{Log}(x) - \mathrm{Log}(y)) - \mathrm{Log}(b/a)\| \leq \varepsilon$
- $\|\mathrm{Log}(y)\|_\infty \leq O(\log d)$

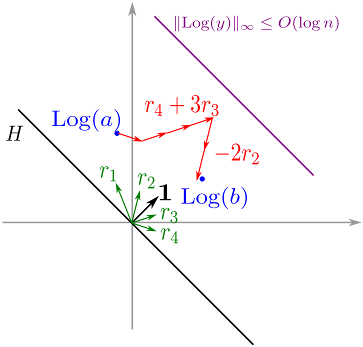


$$r_i = \mathrm{Log}(x_i)\,,\ \ x_i \in R$$

Idea of the algorithm

Objective: find $x, y \in R$ s.t.

- $\|(\mathrm{Log}(x) - \mathrm{Log}(y)) - \mathrm{Log}(b/a)\| \leq \varepsilon$
- $\|\mathrm{Log}(y)\|_\infty \leq O(\log d)$



Solve **exact** CVP in L with target t

$$L = \begin{pmatrix} r_1 & \cdots & r_{d^2} \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix}, \quad t = \begin{pmatrix} \mathrm{Log}(b/a) \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

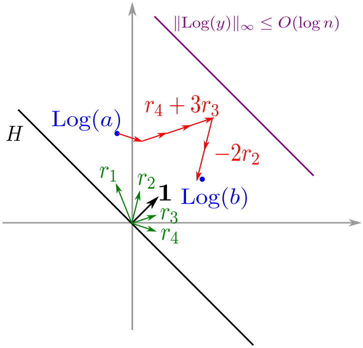
$$r_i = \mathrm{Log}(x_i)\,, \quad x_i \in R$$

(L is fixed and independent of a and b)

Idea of the algorithm

Objective: find $x, y \in R$ s.t.

- $\|(\mathrm{Log}(x) - \mathrm{Log}(y)) - \mathrm{Log}(b/a)\| \leq \varepsilon$
- $\|\mathrm{Log}(y)\|_\infty \leq O(\log d)$



Solve **exact** CVP in L with target t with an oracle

$$L = \begin{pmatrix} r_1 & \cdots & r_{d^2} \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix}, \quad t = \begin{pmatrix} \mathrm{Log}(b/a) \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

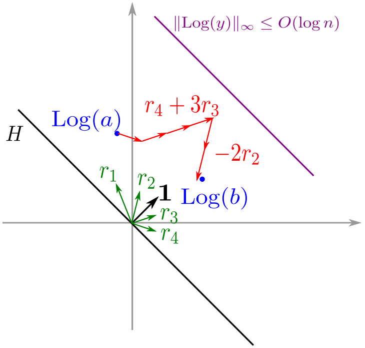
$$r_i = \mathrm{Log}(x_i)\,, \ \ x_i \in R$$

(L is fixed and independent of a and b)

Idea of the algorithm

Objective: find $x, y \in R$ s.t.

- $\|(\mathrm{Log}(x) - \mathrm{Log}(y)) - \mathrm{Log}(b/a)\| \leq \varepsilon$
- $\|\mathrm{Log}(y)\|_\infty \leq O(\log d)$



Solve **exact** CVP in L with target t with an oracle

$$L = \begin{pmatrix} r_1 & \cdots & r_{d^2} \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix}, \quad t = \begin{pmatrix} \mathrm{Log}(b/a) \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

(L is fixed and independent of a and b)

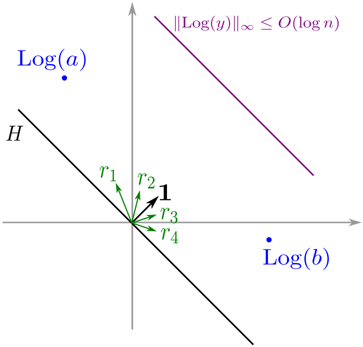
Complexity

Quantum poly time (with the oracle)

Idea of the algorithm

Objective: find $x, y \in R$ s.t.

- $\|(\mathrm{Log}(x) - \mathrm{Log}(y)) - \mathrm{Log}(b/a)\| \leq \varepsilon$
- $\|\mathrm{Log}(y)\|_\infty \leq O(\log d)$



Solve **exact** CVP in L with target t with an oracle

$$L = \begin{pmatrix} r_1 & \cdots & r_{d^2} \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix}, \quad t = \begin{pmatrix} \mathrm{Log}(b/a) \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

(L is fixed and independent of a and b)

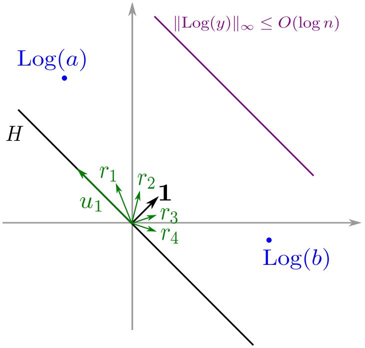
Complexity

Quantum poly time (with the oracle)

Idea of the algorithm

Objective: find $x, y \in R$ s.t.

- $\|(\mathrm{Log}(x) - \mathrm{Log}(y)) - \mathrm{Log}(b/a)\| \leq \varepsilon$
- $\|\mathrm{Log}(y)\|_\infty \leq O(\log d)$



Solve **exact** CVP in L with target t with an oracle

$$L = \begin{pmatrix} U & r_1 & \cdots & r_{d^2} \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}, \quad t = \begin{pmatrix} \mathrm{Log}(b/a) \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

(L is fixed and independent of a and b)

Complexity

Quantum poly time (with the oracle)

Heuristic

$$L = \begin{pmatrix} U & r_1 & \cdots & r_{d^2} \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}, \qquad t = \begin{pmatrix} \mathrm{Log}(b/a) \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Heuristic

$$L = \begin{pmatrix} \beta U & \beta r_1 & \cdots & \beta r_{d^2} \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}, \qquad t = \begin{pmatrix} \beta \operatorname{Log}(b/a) \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Heuristic

$$L = \begin{pmatrix} \beta U & \beta r_1 & \cdots & \beta r_{d^2} \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}, \qquad t = \begin{pmatrix} \beta \operatorname{Log}(b/a) \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Heuristic assumption

For any target $t = (\beta \cdot x, 0, \cdots, 0)^T$, we have

$$\operatorname{dist}(L, t) \leq \sqrt{d}.$$

Heuristic

$$L = \begin{pmatrix} \beta U & \beta r_1 & \cdots & \beta r_{d^2} \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}, \qquad t = \begin{pmatrix} \beta \operatorname{Log}(b/a) \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Heuristic assumption

For any target $t = (\beta \cdot x, 0, \cdots, 0)^T$, we have

$$\mathrm{dist}(L, t) \leq \sqrt{d}.$$

- Numerical experiments (up to dimension $d = 8$)

Heuristic

$$L = \begin{pmatrix} \beta U & \beta r_1 & \cdots & \beta r_{d^2} \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}, \qquad t = \begin{pmatrix} \beta \operatorname{Log}(b/a) \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Heuristic assumption

For any target $t = (\beta \cdot x, 0, \cdots, 0)^T$, we have

$$\mathrm{dist}(L, t) \leq \sqrt{d}.$$

- Numerical experiments (up to dimension $d = 8$)
- Hand waving counting argument
  - assuming $(\beta \cdot x \| \vec{v})$ with $v \xleftarrow{\$} \{\vec{w} \in \{0,1\}^{d^2} \mid \|\vec{w}\|_1 = d\}$ implies x (almost) uniform mod U.

Heuristic

$$L = \begin{pmatrix} \beta U & \beta r_1 & \cdots & \beta r_{d^2} \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}, \qquad t = \begin{pmatrix} \beta \operatorname{Log}(b/a) \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Heuristic assumption

For any target $t = (\beta \cdot x, 0, \cdots, 0)^T$, we have

$$\operatorname{dist}(L, t) \leq \sqrt{d}.$$

- Numerical experiments (up to dimension $d = 8$)
- Hand waving counting argument
  - assuming $(\beta \cdot x \| \vec{v})$ with $v \xleftarrow{\$} \{\vec{w} \in \{0,1\}^{d^2} \mid \|\vec{w}\|_1 = d\}$ implies x (almost) uniform mod U.
  - related to Arakelov random walk $\Rightarrow$ see Léo and Benjamin's talk

Under the carpet

- Any module
  - use pseudo-basis
  - add class group to L (cf [Buc88])
- Proving correctness
  - Lovász' swap condition
  - switch between $\mathcal{N}(\cdot)$ and $\|\cdot\|$
  - handling bit sizes

[Buc88] J. Buchmann. A subexponential algorithm for the determination of class groups and regulators of algebraic number fields. Séminaire de théorie des nombres.

Summary and impact

LLL algorithm for cyclotomic fields

- Approx: quasi-poly(d)$^{O(k)} = 2^{(\log d)^{O(1)} \cdot k}$
- Time: quantum polynomial time
  if oracle solving CVP in L (of dim $O(d^{2+\varepsilon})$)

Summary and impact

LLL algorithm for cyclotomic fields

- Approx: quasi-poly(d)$^{O(k)} = 2^{(\log d)^{O(1)} \cdot k}$
- Time: quantum polynomial time
  if oracle solving CVP in L (of dim $O(d^{2+\varepsilon})$)

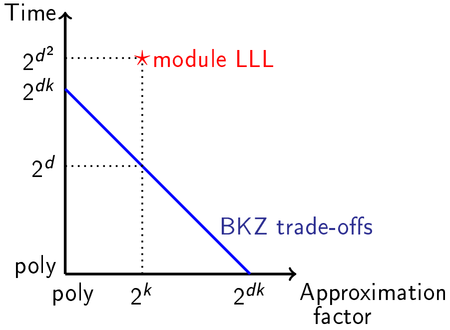
In practice? $\Rightarrow$ replace the oracle by a CVP solver

Summary and impact

LLL algorithm for cyclotomic fields

- Approx: quasi-poly(d)$^{O(k)} = 2^{(\log d)^{O(1)} \cdot k}$
- Time: quantum polynomial time
  if oracle solving CVP in L (of dim $O(d^{2+\varepsilon})$)

In practice? $\Rightarrow$ replace the oracle by a CVP solver

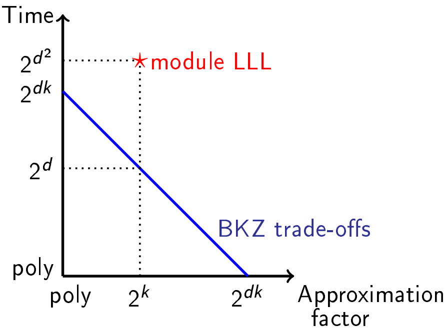


Summary and impact

LLL algorithm for cyclotomic fields

- Approx: quasi-poly(d)$^{O(k)} = 2^{(\log d)^{O(1)} \cdot k}$
- Time: quantum polynomial time
  if oracle solving CVP in L (of dim $O(d^{2+\varepsilon})$)

In practice? $\Rightarrow$ replace the oracle by a CVP solver



$\Rightarrow$ theoretical result
(not practical)

Open problems

(1) Improving CVP in L

- reduce its dimension to $\widetilde{O}(d)$?
- use the structure of L to improve CVP solvers?

Open problems

(1) Improving CVP in L

- reduce its dimension to $\widetilde{O}(d)$?
- use the structure of L to improve CVP solvers?
- CVP with pre-processing is NP-hard [Mic01]

"CVPP in NP-hard": $\forall n$, $\exists L$, $\forall \phi$ (SAT instance with n variables), $\exists \vec{t}$ and α $s.t.$

$$\phi \text{ is satisfiable } \Leftrightarrow \mathrm{dist}(L, \vec{t}) \leq \alpha.$$

[Mic01] D. Micciancio. The hardness of the closest vector problem with preprocessing. Transaction on Information Theory.

Open problems

(1) Improving CVP in L

- reduce its dimension to $\widetilde{O}(d)$?
- use the structure of L to improve CVP solvers?
- CVP with pre-processing is NP-hard [Mic01]

(2) Generalizing LLL to all the BKZ trade-offs?

- reduction rank k to rank β (cf [MS19])
- sieving/enumeration in modules?

[MS19] T. Mukherjee, N. Stephens-Davidowitz. Lattice Reduction for Modules, or How to Reduce ModuleSVP to ModuleSVP, ePrint.

Open problems

(1) Improving CVP in L

- reduce its dimension to $\widetilde{O}(d)$?
- use the structure of L to improve CVP solvers?
- CVP with pre-processing is NP-hard [Mic01]

(2) Generalizing LLL to all the BKZ trade-offs?

- reduction rank k to rank β (cf [MS19])
- sieving/enumeration in modules?

Thank you