Lattice reduction algorithms

Damien Stehlé
(Some slides courtesy of Shi Bai)
(Bibliography: see proceedings)

ENS de Lyon

July 25th 2017
Goal and roadmap

An overview of the algorithmic aspects of lattice reduction

1. **Background on lattices**
2. Solving the Shortest Vector Problem
3. The dynamics of lattice reduction
4. Blocking techniques
5. Approximations
Euclidean lattices

Lattice \( \equiv \left\{ \sum_{i \leq n} x_i b_i : x_i \in \mathbb{Z} \right\} \),
for linearly indep. \( b_i \)'s in \( \mathbb{R}^n \),
referred to as \textbf{lattice basis}

Bases are \textbf{not unique}, but can be obtained from one another by integer transforms of determinant \( \pm 1 \):

\[
\begin{bmatrix}
-2 & 1 \\
10 & 6
\end{bmatrix} =
\begin{bmatrix}
4 & -3 \\
2 & 4
\end{bmatrix} \cdot
\begin{bmatrix}
1 & 1 \\
2 & 1
\end{bmatrix}
\]

Lattice reduction

Find a short basis, given an arbitrary one
Euclidean lattices

Lattice \( \equiv \left\{ \sum_{i \leq n} x_i b_i : x_i \in \mathbb{Z} \right\} \),
for linearly indep. \( b_i \)'s in \( \mathbb{R}^n \),
referred to as **lattice basis**

Bases are **not unique**, but can be obtained from one another by integer transforms of determinant \( \pm 1 \):

\[
\begin{bmatrix}
-2 & 1 \\
10 & 6
\end{bmatrix} = \begin{bmatrix}
4 & -3 \\
2 & 4
\end{bmatrix} \cdot \begin{bmatrix}
1 & 1 \\
2 & 1
\end{bmatrix}
\]

Lattice reduction

Find a short basis, given an arbitrary one
Introduction

Background on lattices

SVP Dynamics

Blocking

Approximations

Conclusion

Euclidean lattices

Lattice \( \equiv \{ \sum_{i \leq n} x_i b_i : x_i \in \mathbb{Z} \} \), for linearly indep. \( b_i \)'s in \( \mathbb{R}^n \), referred to as **lattice basis**

Bases are **not unique**, but can be obtained from one another by integer transforms of determinant \( \pm 1 \):

\[
\begin{bmatrix}
-2 & 1 \\
10 & 6
\end{bmatrix} = \begin{bmatrix}
4 & -3 \\
2 & 4
\end{bmatrix} \cdot \begin{bmatrix}
1 & 1 \\
2 & 1
\end{bmatrix}
\]

Lattice reduction

Find a short basis, given an arbitrary one
**Minimum**

\[ \lambda(L) = \min \{ \|b\| : b \in L \setminus 0 \} \]

**Determinant**

\[ \det L = |\det(b_i)_i|, \text{ for any basis} \]

**Minkowski theorem**

\[ \lambda(L) \leq \sqrt{n} \cdot (\det L)^{\frac{1}{n}}, \text{ for any } L \text{ of dim } n \]

**Lattice reduction**

Find a basis that is short compared to \( \lambda(L) \) and/or \( (\det L)^{\frac{1}{n}} \)
Lattice invariants

**Minimum**
\[ \lambda(L) = \min \{ \|b\| : b \in L \setminus 0 \} \]

**Determinant**
\[ \det L = |\det(b_i)|, \text{ for any basis} \]

**Minkowski theorem**
\[ \lambda(L) \leq \sqrt{n} \cdot (\det L)^{\frac{1}{n}}, \text{ for any } L \text{ of dim } n \]

**Lattice reduction**
Find a basis that is short compared to \( \lambda(L) \) and/or \( (\det L)^{\frac{1}{n}} \)
**Lattice invariants**

**Minimum**
\[ \lambda(L) = \min \{ \|b\| : b \in L \setminus 0 \} \]

**Determinant**
\[ \det L = |\det(b_i)_i|, \text{ for any basis} \]

**Minkowski theorem**
\[ \lambda(L) \leq \sqrt{n} \cdot (\det L)^{\frac{1}{n}}, \text{ for any } L \text{ of dim } n \]

**Lattice reduction**
Find a basis that is short compared to \( \lambda(L) \) and/or \( (\det L)^{\frac{1}{n}} \)
Lattice invariants

Minimum
\[ \lambda(L) = \min \{ \| b \| : b \in L \setminus 0 \} \]

Determinant
\[ \det L = | \det(b_i)_i |, \text{ for any basis} \]

Minkowski theorem
\[ \lambda(L) \leq \sqrt{n} \cdot (\det L)^{\frac{1}{n}}, \text{ for any } L \text{ of dim } n \]

Lattice reduction
Find a basis that is short compared to \( \lambda(L) \) and/or \( (\det L)^{\frac{1}{n}} \)
The Shortest Vector Problem

SVP\(\gamma\), \(\gamma \geq 1\)

Given as input a basis matrix \(B\) of a lattice \(L\), find \(x \in \mathbb{Z}^n\) s.t.

\[
0 < \|Bx\| \leq \gamma \cdot \lambda(L)
\]

HSVP\(\gamma\), \(\gamma \geq 1\) (Hermite-SVP)

Given as input a basis matrix \(B\) of a lattice \(L\), find \(x \in \mathbb{Z}^n\) s.t.

\[
0 < \|Bx\| \leq \gamma \cdot (\det L)^{1/(\dim L)}
\]

- The dimension drives computational hardness
- SVP is NP-hard under prob. reductions for \(\gamma \leq O(1)\)
- The problems get easier when \(\gamma\) increases
- HSVP and SVP reduce to one another (up to increases of \(\gamma\))
The Shortest Vector Problem

\[ \text{SVP}_\gamma, \; \gamma \geq 1 \]

Given as input a basis matrix \( B \) of a lattice \( L \), find \( x \in \mathbb{Z}^n \) s.t.
\[
0 < \|Bx\| \leq \gamma \cdot \lambda(L)
\]

\[ \text{HSVP}_\gamma, \; \gamma \geq 1 \quad (\text{Hermite-SVP}) \]

Given as input a basis matrix \( B \) of a lattice \( L \), find \( x \in \mathbb{Z}^n \) s.t.
\[
0 < \|Bx\| \leq \gamma \cdot (\det L)^{1/(\dim L)}
\]

- The dimension drives computational hardness
- SVP is NP-hard under prob. reductions for \( \gamma \leq O(1) \)
- The problems get easier when \( \gamma \) increases
- HSVP and SVP reduce to one another (up to increases of \( \gamma \))
Why do we care?

Lots of computational problems can be cast as finding a short vector in a lattice.

- **Communication theory:** white Gaussian noise channel
- **Combinatorial optimization:** integer linear programming
- **Number theory:** invariants of number fields
- **Cryptanalysis:** knapsacks, RSA variants, lattice-based crypto
- **Computer algebra:** factorisation of integer polynomials
Example 1: Reconstructing an algebraic number

Let $\alpha \in \mathbb{R}$ algebraic, and $P_\alpha$ its minimal polynomial. How to recover $P_\alpha$ from an approximation $\overline{\alpha}$ of $\alpha$?

$$L := \begin{bmatrix} 1 & \overline{\alpha} & \overline{\alpha}^2 & \ldots & \overline{\alpha}^d \\ \varepsilon & 0 & 0 & \ldots & 0 \\ 0 & \varepsilon & 0 & \ldots & 0 \\ 0 & 0 & \varepsilon & \ldots & 0 \\ \vdots \\ 0 & 0 & 0 & \ldots & \varepsilon \end{bmatrix} \cdot \mathbb{Z}^{d+1}$$

For $d = \deg P_\alpha$, $\varepsilon$ small and $|\alpha - \overline{\alpha}|$ small, any short enough vector in $L$ leads to $P_\alpha$.

We want to be able to do that!
Example 1: Reconstructing an algebraic number

Let $\alpha \in \mathbb{R}$ algebraic, and $P_\alpha$ its minimal polynomial. How to recover $P_\alpha$ from an approximation $\overline{\alpha}$ of $\alpha$?

Let $
abla := \begin{bmatrix}
1 & \overline{\alpha} & \overline{\alpha}^2 & \ldots & \overline{\alpha}^d \\
\varepsilon & 0 & 0 & \ldots & 0 \\
0 & \varepsilon & 0 & \ldots & 0 \\
0 & 0 & \varepsilon & \ldots & 0 \\
\vdots \\
0 & 0 & 0 & \ldots & \varepsilon 
\end{bmatrix} \cdot \mathbb{Z}^{d+1}$

For $d = \deg P_\alpha$, $\varepsilon$ small and $|\alpha - \overline{\alpha}|$ small, any short enough vector in $L$ leads to $P_\alpha$.

We want to be able to do that!
Example 1: Reconstructing an algebraic number

Let $\alpha \in \mathbb{R}$ algebraic, and $P_\alpha$ its minimal polynomial. How to recover $P_\alpha$ from an approximation $\bar{\alpha}$ of $\alpha$?

$$L := \begin{bmatrix}
1 & \bar{\alpha} & \bar{\alpha}^2 & \ldots & \bar{\alpha}^d \\
\varepsilon & 0 & 0 & \ldots & 0 \\
0 & \varepsilon & 0 & \ldots & 0 \\
0 & 0 & \varepsilon & \ldots & 0 \\
& \ddots & & & \varepsilon \\
0 & 0 & 0 & \ldots & \varepsilon
\end{bmatrix} \cdot \mathbb{Z}^{d+1}$$

For $d = \deg P_\alpha$, $\varepsilon$ small and $|\alpha - \bar{\alpha}|$ small, any short enough vector in $L$ leads to $P_\alpha$.

We want to be able to do that!
Example 2: Collisions in Ajtai’s hash function

Ajtai’s hash function

Let $n \ll m$, $t \ll q$ and $A \in (\mathbb{Z}/q\mathbb{Z})^{n \times m}$. We define:

$$h_A : \{-t, \ldots, +t\}^m \rightarrow (\mathbb{Z}/q\mathbb{Z})^n$$

$$x \mapsto A \cdot x \mod q$$

Finding a collision is finding $x \neq 0$ small in

$$L := \{x \in \mathbb{Z}^m : A \cdot x = 0 \mod q\}.$$ 

We want this to be intractable!
Example 2: Collisions in Ajtai’s hash function

Ajtai’s hash function

Let \( n \ll m, t \ll q \) and \( A \in (\mathbb{Z}/q\mathbb{Z})^{n \times m} \). We define:

\[
h_A : \{-t, \ldots, +t\}^m \rightarrow (\mathbb{Z}/q\mathbb{Z})^n \quad \text{x} \mapsto A \cdot x \pmod q
\]

Finding a collision is finding \( x \neq 0 \) small in

\[
L := \{x \in \mathbb{Z}^m : A \cdot x = 0 \pmod q\}.
\]

We want this to be intractable!
Roadmap

1. Background on lattices
2. **Solving the Shortest Vector Problem**
3. The dynamics of lattice reduction
4. Blocking techniques
5. Approximations

Solving $\text{SVP}_\gamma$ with $\gamma = 1$
“Optimal reduction”: HKZ
QR factorization

- $Q^TQ = I$
- $R$ up-triangular with positive diagonal entries
- $\|Bx\| = \|Rx\|$
- $Rx = \left( \sum_{i \geq 1} r_1i x_i, \sum_{i \geq 2} r_2i x_i, \ldots, r_{n-1,n-1} x_{n-1} + r_{n-1,n} x_n, r_{nn} x_n \right)$
QR factorization

- $Q^TQ = I$
- $R$ up-triangular with positive diagonal entries
- $\|Bx\| = \|Rx\|$
- $Rx = \left( \sum_{i \geq 1} r_{1i}x_i, \sum_{i \geq 2} r_{2i}x_i, \ldots, r_{n-1,n-1}x_{n-1} + r_{n-1,n}x_n, r_{nn}x_n \right)$
QR factorization

- $Q^T Q = I$
- $R$ up-triangular with positive diagonal entries
- $\|Bx\| = \|Rx\|$
- $Rx =$

\[
\begin{pmatrix}
\sum_{i \geq 1} r_{1i}x_i, & \sum_{i \geq 2} r_{2i}x_i, & \ldots, & r_{n-1,n-1}x_{n-1} + r_{n-1,n}x_n, & r_{nn}x_n
\end{pmatrix}
\]
Solving SVP by enumeration

\[
\left( \sum_{i \geq 1} r_{1i} x_i, \sum_{i \geq 2} r_{2i} x_i, \ldots, r_{n-1,n-1} x_{n-1} + r_{n-1,n} x_n, \ r_{nn} x_n \right)
\]

- Set a norm bound \( S \)
- List all \( x_n \in \mathbb{Z} \) s.t. \( |r_{nn} \cdot x_n| \leq S \)
- For each \( x_n \), list all \( x_{n-1} \in \mathbb{Z} \) s.t. the partial vector \( (r_{n-1,n-1} x_{n-1} + r_{n-1,n} x_n, \ r_{nn} x_n) \) has norm \( \leq S \)
- etc
- For each \( (x_n, x_{n-1}, \ldots, x_2) \), list all possible \( x_1 \in \mathbb{Z} \)
Enumeration

- This is a search of leaves in a big tree:
  - Depth-first search $\Rightarrow Poly(n)$ memory
  - Zig-zag around the center
  - Well-suited for parallelization
  - Can do (heuristic) tree pruning

- Huge cost, growing with $S, 1/r_{nn}, 1/(r_{nn}r_{n-1,n-1}), \ldots$
  - $S$ can be chosen tightly in many cases
  - The last $r_{ii}$'s can be increased with lattice reduction
## Enumeration versus other SVP solvers

<table>
<thead>
<tr>
<th>Method</th>
<th>Time upper bound</th>
<th>Space upper bound</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>via enumeration</td>
<td>$n^n/(2e)+o(n)$</td>
<td>$\mathcal{P}oly(n)$</td>
<td>Deterministic</td>
</tr>
<tr>
<td>[FiPo'83, Kan'83, HaSt'07]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>via sieving</td>
<td>$2^{2.47n+o(n)}$</td>
<td>$2^{1.325n+o(n)}$</td>
<td>Probabilistic</td>
</tr>
<tr>
<td>[AjKuSi'01, PuSt'09]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>via heuristic sieving</td>
<td>$2^{0.292n+o(n)}$</td>
<td>$2^{0.292n+o(n)}$</td>
<td>Heuristic</td>
</tr>
<tr>
<td>[MiVo'10, BDGL'16]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>via Voronoi cell</td>
<td>$2^{2n+o(n)}$</td>
<td>$2^n+o(n)$</td>
<td>Deterministic</td>
</tr>
<tr>
<td>[MiVo'10]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>via Gaussians</td>
<td>$2^n+o(n)$</td>
<td>$2^n+o(n)$</td>
<td>Probabilistic</td>
</tr>
<tr>
<td>[ADRS'16]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In practice, enumeration wins

\[ \log_2(\text{time}) \]

![Graph showing the performance of different algorithms](image)
Interlude: implementations of lattice algorithms

MAGMA, PARI-GP, NTL, Maple, Mathematica, SAGE, etc

Reference implementation: fplll
- C++, with a Python interface (fpylll)
- GNU LGPL
- hosted on github
- enumeration, sieving, LLL, BKZ
What if we want a full short basis?

Enumeration gives a single vector...

**HKZ-reduction** (Hermite Korkine Zolotarev)

\( \mathbf{R} \) up-triangular is **HKZ-reduced** if

- \( r_{11} = \lambda(L(\mathbf{R})) \)
- and \( (r_{ij})_{i,j>1} \) is HKZ-reduced

Minkowski’s theorem implies that for all \( i \leq n \):

\[
r_{ii} \leq \sqrt{n - i + 1} \cdot \left( \prod_{j=i}^{n} r_{jj} \right)^{\frac{1}{n-i+1}}
\]

If these are equalities, then fixing the last one fixes them all.
What if we want a full short basis?

Enumeration gives a single vector...

HKZ-reduction (Hermite Korkine Zolotarev)

$R$ up-triangular is **HKZ-reduced** if

1. $r_{11} = \lambda(L(R))$
2. and $(r_{ij})_{i,j>1}$ is HKZ-reduced

Minkowski’s theorem implies that for all $i \leq n$: 

$$r_{ii} \leq \sqrt{n-i+1} \cdot \left( \prod_{j=i}^{n} r_{jj} \right)^{\frac{1}{n-i+1}}$$

If these are equalities, then fixing the last one fixes them all.
Shape of HKZ-reduced bases

\[ \rho_i = \log r_{ii} \sim \log^2(n - i + 1) \]
Roadmap

- Background on lattices
- **Solving the Shortest Vector Problem**
- The dynamics of lattice reduction
- Blocking techniques
- Approximations

**Open problems:**

1. When does sieving beat enumeration?
2. Can we make heuristic sieving less heuristic?
3. Is HKZ-reduction “optimal”?
Roadmap

1. Background on lattices
2. Solving the Shortest Vector Problem
3. The dynamics of lattice reduction
4. Blocking techniques
5. Approximations
From $n(n + 1)/2$ to $n$ variables: size-reduction

\[
\begin{bmatrix}
\ldots & \ldots & r_{ij} & \ldots \\
r_{ii} & \ldots & \ldots & \ldots \\
\ldots & \ldots & \ldots & \ldots \\
r_{jj} & \ldots & \ldots & \ldots \\
\end{bmatrix}
\begin{bmatrix}
1 \\
-\frac{r_{ij}}{r_{ii}} \\
\ldots \\
1 \\
\end{bmatrix}
\Rightarrow |r_{ij}^{(new)}| \leq \frac{r_{ii}}{2}
\]

( Go from left to right, and bottom to top )

- Triangular linear system solving, with roundings
- Number of arithmetic steps: $O(n^2)$ per column
- The magnitudes of the rationals can grow by a factor $2^{O(n)}$ during the computation
From $n(n+1)/2$ to $n$ variables: size-reduction

\[
\begin{pmatrix}
\ddots & \cdots & \cdots & \cdots \\
\cdot & r_{ii} & \cdots & r_{ij} \\
\cdot & \ddots & \ddots & \cdots \\
\cdot & \cdot & r_{jj} & \cdots \\
\end{pmatrix}
\begin{pmatrix}
\ddots & \cdot & \cdot & \cdot \\
1 & -\frac{r_{ij}}{r_{ii}} & \cdot & \cdot \\
\cdot & \ddots & 1 & \cdot \\
\cdot & \cdot & \cdot & \cdot \\
\end{pmatrix}
\Rightarrow |r_{ij}^{(new)}| \leq \frac{r_{ii}}{2}
\]

( Go from left to right, and bottom to top )

- Triangular linear system solving, with roundings
- Number of arithmetic steps: $O(n^2)$ per column
- The magnitudes of the rationals can grow by a factor $2^{O(n)}$ during the computation
From $n(n + 1)/2$ to $n$ variables: size-reduction

\[
\begin{bmatrix}
\vdots & \vdots & \cdots & \vdots \\
\vdots & \ddots & \cdots & \vdots \\
\vdots & \cdots & \ddots & \vdots \\
\vdots & \cdots & \cdots & \ddots
\end{bmatrix}
\begin{bmatrix}
\vdots \\
1 \\
\vdots \\
1
\end{bmatrix}
\Rightarrow |r_{ij}^{(new)}| \leq \frac{r_{ii}}{2}
\]

( Go from left to right, and bottom to top )

- Triangular linear system solving, with roundings
- Number of arithmetic steps: $O(n^2)$ per column
- The magnitudes of the rationals can grow by a factor $2^{O(n)}$ during the computation
From \( n(n+1)/2 \) to \( n \) variables: size-reduction

\[
\begin{bmatrix}
\ddots & \cdots & r_{ij} & \cdots \\
\vdots & \ddots & \vdots & \vdots \\
r_{ij} & \ddots & \ddots & \vdots \\
& \ddots & \ddots & \ddots \\
\end{bmatrix}
\begin{bmatrix}
\ddots & 1 & -\frac{r_{ij}}{r_{ii}} & \cdots \\
\vdots & \ddots & 1 & \vdots \\
\vdots & \ddots & \ddots & \vdots \\
\vdots & \ddots & \ddots & \ddots \\
\end{bmatrix}
\Rightarrow |r_{ij}^{(new)}| \leq \frac{r_{ii}}{2}
\]

( Go from left to right, and bottom to top )

- Triangular linear system solving, with roundings
- Number of arithmetic steps: \( O(n^2) \) per column
- The magnitudes of the rationals can grow by a factor \( 2^{O(n)} \) during the computation
Size-reduction: updating $B$
Where are we now?

Goal of lattice reduction

Given $R \in \mathbb{R}^{n \times n}$ up-triangular, find $U \in \text{GL}_n(\mathbb{Z})$ s.t. the $R$-factor of $R \cdot U$ has small diagonal coeffs.

Constraint: the product of the $r_{ii}$'s is constant.

We want to

- make the first $r_{ii}$'s small
- prevent the $r_{ii}$'s from decreasing fast

HKZ is too costly
Goal of lattice reduction

Given $\mathbf{R} \in \mathbb{R}^{n \times n}$ up-triangular, find $\mathbf{U} \in \text{GL}_n(\mathbb{Z})$ s.t. the $R$-factor of $\mathbf{R} \cdot \mathbf{U}$ has small diagonal coeffs

Constraint: the product of the $r_{ii}$'s is constant.

We want to

- make the first $r_{ii}$'s small
- prevent the $r_{ii}$'s from decreasing fast

HKZ is too costly
Where are we now?

Goal of lattice reduction

Given $R \in \mathbb{R}^{n \times n}$ up-triangular, find $U \in \text{GL}_n(\mathbb{Z})$ s.t. the R-factor of $R \cdot U$ has small diagonal coeffs

Constraint: the product of the $r_{ii}$'s is constant.

We want to

- make the first $r_{ii}$'s small
- prevent the $r_{ii}$'s from decreasing fast

HKZ is too costly
The LLL strategy

(Lenstra Lenstra Lovász)

Take any $i$ such that $r_{i+1,i+1} \ll r_{ii}$, and swap $b_i$ and $b_{i+1}$.
The LLL strategy (Lenstra Lenstra Lovász)

Take any $i$ such that $r_{i+1,i+1} \ll r_{i,i}$, and swap $b_i$ and $b_{i+1}$

$$(r_{i,i}^{new})^2 = r_{i+1,i+1}^2 + r_{i,i+1}^2 \leq r_{i+1,i+1}^2 + r_{i,i}/4$$

$$r_{i+1,i+1}^2 \leq (3/4) \cdot r_{i,i}^2 \Rightarrow (r_{i,i}^{new})^2 \leq r_{i,i}^2$$

$r_{i+1,i+1}$ cannot go wild, as $r_{i+1,i+1}^{new} \cdot r_{i,i}^{new} = r_{i+1,i+1} \cdot r_{i,i}$. 
The LLL strategy  

(Lenstra Lenstra Lovász)

Take any $i$ such that $r_{i+1,i+1} \ll r_{i,i}$, and swap $b_i$ and $b_{i+1}$

$$(r_{i,i}^{\text{new}})^2 = r_{i+1,i+1}^2 + r_{i,i+1}^2 \leq r_{i+1,i+1}^2 + r_{i,i}^2 / 4$$

$$r_{i+1,i+1}^2 \leq (3/4) \cdot r_{i,i}^2 \Rightarrow (r_{i,i}^{\text{new}})^2 \leq r_{i,i}^2$$

$r_{i+1,i+1}$ cannot go wild, as $r_{i+1,i+1}^{\text{new}} \cdot r_{i,i}^{\text{new}} = r_{i+1,i+1} \cdot r_{i,i}$. 
The LLL strategy (Lenstra Lenstra Lovász)

Take any \( i \) such that \( r_{i+1,i+1} \ll r_{i,i} \), and swap \( b_i \) and \( b_{i+1} \)

\[
(r_{i,i}^{\text{new}})^2 = r_{i+1,i+1}^2 + r_{i,i+1}^2 \leq r_{i+1,i+1}^2 + r_{i,i}/4
\]

\[
r_{i+1,i+1}^2 \leq (3/4) \cdot r_{i,i}^2 \quad \Rightarrow \quad (r_{i,i}^{\text{new}})^2 \leq r_{i,i}^2
\]

\( r_{i+1,i+1} \) cannot go wild, as \( r_{i+1,i+1}^{\text{new}} \cdot r_{i,i}^{\text{new}} = r_{i+1,i+1} \cdot r_{i,i} \).
LLL as sandpile flattening

If $r_{i\rightarrow i} \gg r_{i+1, i+1}$, do $b_i \leftrightarrow b_{i+1}$.

If $\rho_i \gg \rho_{i+1}$, decrease $\rho_i$ by $C$ and increase $\rho_{i+1}$ by $C$. 
If $r_{ii} \gg r_{i+1,i+1}$, do $b_i \leftrightarrow b_{i+1}$.

If $\rho_i \gg \rho_{i+1}$, decrease $\rho_i$ by $C$ and increase $\rho_{i+1}$ by $C$. 

LLL as sandpile flattening

$\rho_i = \log r_{ii}$
LLL as sandpile flattening

If $r_{ii} \gg r_{i+1,i+1}$, do $b_i \leftrightarrow b_{i+1}$.

If $\rho_i \gg \rho_{i+1}$, decrease $\rho_i$ by $C$ and increase $\rho_{i+1}$ by $C$. 

\[ \rho_i = \log n_{ii} \]
LLL as sandpile flattening

If \( r_{ij} \gg r_{i+1,j+1} \), do \( b_i \leftrightarrow b_{i+1} \).

If \( \rho_i \gg \rho_{i+1} \), decrease \( \rho_i \) by \( C \) and increase \( \rho_{i+1} \) by \( C \).
If $r_{ii} \gg r_{i+1,i+1}$, do $b_i \leftrightarrow b_{i+1}$.

If $\rho_i \gg \rho_{i+1}$, decrease $\rho_i$ by $C$ and increase $\rho_{i+1}$ by $C$. 
LLL as sandpile flattening

If \( r_{ii} \gg r_{i+1,i+1} \), do \( b_i \leftrightarrow b_{i+1} \).

If \( \rho_i \gg \rho_{i+1} \), decrease \( \rho_i \) by \( C \) and increase \( \rho_{i+1} \) by \( C \).
On a real example

\[ \log r_i \]

\[ \log r_i \]

\[ i \]

25/07/2017
On a real example
On a real example
On a real example
On a real example

\[ \log r_i \]

\[ i \]

D. Stehlé

Lattice reduction algorithms
On a real example

\[ \log r_i \]

\[ r_i \]

\[ i \]
On a real example

\[ \log r_i \]

\[ 500 \]
\[ 400 \]
\[ 300 \]
\[ 200 \]
\[ 100 \]

\[ i \]
On a real example
On a real example
On a real example

\[ \log r_i \]

\[ i \]

![Graph showing \( \log r_i \) versus \( i \)]
## Convergence of LLL

### The LLL potential

\[ \Pi := \sum_{i \leq n} (n - i + 1) \cdot \rho_i \]

- Weighted amount of sand to be moved to the right
- For each swap, it decreases by at least a constant

### Number of loop iterations of LLL

\[ O(n^2 \log \| \mathbf{B} \|) \] loop iterations, if the input basis \( \mathbf{B} \) is integral.
Convergence of LLL

The LLL potential

$$\Pi := \sum_{i \leq n} (n - i + 1) \cdot \rho_i$$

- Weighted amount of sand to be moved to the right
- For each swap, it decreases by at least a constant

Number of loop iterations of LLL

$$O(n^2 \log \|B\|)$$ loop iterations, if the input basis $B$ is integral.
Convergence of LLL

The LLL potential

\[ \Pi := \sum_{i \leq n} (n - i + 1) \cdot \rho_i \]

- Weighted amount of sand to be moved to the right
- For each swap, it decreases by at least a constant

Number of loop iterations of LLL

\[ O(n^2 \log \|B\|) \] loop iterations, if the input basis \( B \) is integral.
The BKZ-LLL strategy

For $i = 1, 2, \ldots, n - 1$ and over and over again,

$\text{HKZ-reduce } \begin{pmatrix} r_{i,i} & r_{i+1,i} \\ 0 & r_{i+1,i+1} \end{pmatrix}$
The BKZ-LLL strategy

For \( i = 1, 2, \ldots, n - 1 \) and over and over again,

\[
\text{HKZ-reduce } \begin{pmatrix} r_{i,i} & r_{i+1,i} \\ 0 & r_{i+1,i+1} \end{pmatrix}
\]

By Minkowski’s theorem:

\[
r_{i,i}^{\text{new}} \leq \sqrt{4/3} \cdot (r_{i,i} \cdot r_{i+1,i+1})^{1/2}
\]

\( r_{i+1,i+1} \) cannot go wild, as \( r_{i+1,i+1}^{\text{new}} \cdot r_{i,i}^{\text{new}} = r_{i+1,i+1} \cdot r_{i,i} \).
The BKZ-LLL strategy

For $i = 1, 2, \ldots, n - 1$ and over and over again,

HKZ-reduce \[ \begin{pmatrix} r_{i,i} & r_{i+1,i} \\ 0 & r_{i+1,i+1} \end{pmatrix} \]

By Minkowski’s theorem:

\[ r_{i,i}^{\text{new}} \leq \sqrt{4/3} \cdot (r_{i,i} \cdot r_{i+1,i+1})^{1/2} \]

$r_{i+1,i+1}$ cannot go wild, as $r_{i+1,i+1}^{\text{new}} \cdot r_{i,i}^{\text{new}} = r_{i+1,i+1} \cdot r_{i,i}$. 
A sandpile model for BKZ-LLL

Regularity assumption: Each HKZ-reduction gives

\[ r_{i,i}^{new} = \sqrt{\frac{4}{3}} \cdot (r_{i,i} \cdot r_{i+1,i+1})^{1/2} \]

\[ \rho \leftarrow \begin{bmatrix} 1/2 & 1/2 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \cdot \rho \]

\[ \rho \leftarrow \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1/2 & 1/2 & 0 \\ 0 & 1/2 & 1/2 & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \cdot \rho \]

A full tour: \( \rho \leftarrow A \cdot \rho + \Gamma \)
Discrete-time affine dynamical system

A full tour: $\rho \leftarrow A \cdot \rho + \Gamma$

- Output quality $\leftarrow$ Fix-point
- Speed of convergence $\leftarrow$ Second largest eigenvalue

Neumaier's potential

$$\nu := \max_{i < n} \frac{1}{n - i} \left( \frac{\sum_{j \leq i} \rho_j}{i} - \frac{\sum_{j \leq n} \rho_j}{n} \right).$$

- $(\sum_{j \leq i} \rho_j)/i$ is a smoothed proxy for $\rho_i$.
- The definition is justified by the fact we expect the $\rho_i$'s to decrease linearly after reduction.
Analyses of BKZ-LLL

Discrete-time affine dynamical system

\[ \rho \leftarrow A \cdot \rho + \Gamma \]

- Output quality \leftarrow \text{Fix-point}
- Speed of convergence \leftarrow \text{Second largest eigenvalue}

Neumaier’s potential

\[ \nu := \max_{i < n} \frac{1}{n - i} \left( \frac{\sum_{j \leq i} \rho_j}{i} - \frac{\sum_{j \leq n} \rho_j}{n} \right) \]

- \((\sum_{j \leq i} \rho_j)/i\) is a smoothed proxy for \(\rho_i\).
- The definition is justified by the fact we expect the \(\rho_i\)’s to decrease linearly after reduction.
Cost/quality of BKZ-LLL vs LLL

<table>
<thead>
<tr>
<th></th>
<th>LLL</th>
<th>BKZ-LLL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVP’s $\gamma$</td>
<td>$(\sqrt{4/3} + \varepsilon)^{n-1}$</td>
<td>?</td>
</tr>
<tr>
<td>HSVP’s $\gamma$</td>
<td>$(\sqrt{4/3} + \varepsilon)^{(n-1)/2}$</td>
<td>$(\sqrt{4/3})^{(n-1)/2}(1 + \varepsilon)$</td>
</tr>
<tr>
<td>Iterations</td>
<td>$n^2 \cdot \log |B|$</td>
<td>$n^3 \cdot \log \log |B|$</td>
</tr>
</tbody>
</table>

(SVP: $\gamma = r_{11}/\lambda_1$, HSVP: $\gamma = r_{11}/\det^{1/n}$)
Roadmap

1. Background on lattices
2. Solving the Shortest Vector Problem
3. **The dynamics of lattice reduction**
4. Blocking techniques
5. Approximations

**Open problems:**

1. SVP rather than HSVP for BKZ-LLL?
2. Can we prove a lower bound on the speed of convergence?
3. Algorithms that do not belong to this framework?
Roadmap

1. Background on lattices
2. Solving the Shortest Vector Problem
3. The dynamics of lattice reduction
4. **Blocking techniques**
5. Approximations

Blocking to improve efficiency
Blocking to improve reducedness
Cost of LLL

Text-book LLL:
- $O(n^2 \log \|B\|)$ loop iterations
- $O(n^2)$ arithmetic operations per iteration
- $\mathbb{R}$ represented with rationals of bit-lengths $O(n \log \|B\|)$

$\Rightarrow$ Cost is $\tilde{O}(n^5 \log^2 \|B\|)$

Using BKZ-LLL:

$\tilde{O}(n^3) \cdot O(n^2) \cdot \tilde{O}(n \log \|B\|) = \tilde{O}(n^6 \log \|B\|)$
Cost of LLL

Text-book LLL:
- $O(n^2 \log \|B\|)$ loop iterations
- $O(n^2)$ arithmetic operations per iteration
- $\mathbb{R}$ represented with rationals of bit-lengths $O(n \log \|B\|)$

$\Rightarrow$ Cost is $\tilde{O}(n^5 \log^2 \|B\|)$

Using BKZ-LLL:

$$\tilde{O}(n^3) \cdot O(n^2) \cdot \tilde{O}(n \log \|B\|) = \tilde{O}(n^6 \log \|B\|)$$
Blocking allows to stay local

Local is more efficient

As long as \( i \) stays in a length \( k \) interval, then R-updates and size-reductions have costs that depend on \( k \) and not on \( n \).

Used to decrease the impact of \( n \) on the cost

- Cost depends on \( n \) only when \( i \) enters or exits the interval
- May be combined with fast linear algebra
Blocking allows to stay local

Local is more efficient

As long as $i$ stays in a length $k$ interval, then R-updates and size-reductions have costs that depend on $k$ and not on $n$.

\[
\begin{pmatrix}
\ddots & \ddots & \ddots & \ddots \\
& r_{i,i} & \ddots & \ddots \\
& \vdots & \ddots & r_{i,i+k-1} \\
& \vdots & \vdots & \vdots & \ddots \\
& \vdots & \vdots & \vdots & \ddots \\
\end{pmatrix}
\]

Used to decrease the impact of $n$ on the cost

- Cost depends on $n$ only when $i$ enters or exits the interval
- May be combined with fast linear algebra
Cost of updating a block

- To minimize cost, stay within a block for long
- May use recursive blocking to reduce the cost further
- Locality seems incompatible with fast convergence
Cost of updating a block

To minimize cost, stay within a block for long
May use recursive blocking to reduce the cost further
Locality seems incompatible with fast convergence
Neumaier-S.’16: both global and local

- Global to limit the impact of $\log \| B \|$ on the cost
- Local to limit the impact of $n$ on the cost

Recursive calls in dim. $k$, with blocks that overlap by half

At the bottom of the recursion, use 2-dim. reduction

2-dim. reduction costs $\tilde{O}(n \log \| B \|)$

Total cost: $\tilde{O}(n^4 \log \| B \|)$
Neumaier-S.’16: both global and local

- Global to limit the impact of $\log \|B\|$ on the cost
- Local to limit the impact of $n$ on the cost

Recursive calls in dim. $k$, with blocks that overlap by half

- At the bottom of the recursion, use 2-dim. reduction
- 2-dim. reduction costs $\tilde{O}(n \log \|B\|)$
- Total cost: $\tilde{O}(n^4 \log \|B\|)$
Neumaier-S.'16: both global and local

- Global to limit the impact of $\log \|B\|$ on the cost
- Local to limit the impact of $n$ on the cost

- Recursive calls in dim. $k$, with blocks that overlap by half
- At the bottom of the recursion, use 2-dim. reduction

- 2-dim. reduction costs $\tilde{O}(n \log \|B\|)$
- Total cost: $\tilde{O}(n^4 \log \|B\|)$
Neumaier-S.’16: both global and local

- Global to limit the impact of $\log ||B||$ on the cost
- Local to limit the impact of $n$ on the cost

- Recursive calls in dim. $k$, with blocks that overlap by half
- At the bottom of the recursion, use 2-dim. reduction
- 2-dim. reduction costs $\tilde{O}(n \log ||B||)$

- Total cost: $\tilde{O}(n^4 \log ||B||)$
Neumaier-S.’16: both global and local

- Global to limit the impact of $\log \| B \|$ on the cost
- Local to limit the impact of $n$ on the cost

Recursive calls in dim. $k$, with blocks that overlap by half

At the bottom of the recursion, use 2-dim. reduction

2-dim. reduction costs $\tilde{O}(n \log \| B \|)$

Total cost: $\tilde{O}(n^4 \log \| B \|)$
Blocking to improve reducedness

- LLL and its variants achieve exponential (H)SVP approximation factors in polynomial-time
- Can we do better by paying more?

**BKZ** (Block Korkine-Zolotarev)

- HKZ calls in dim. $k$, with blocks that overlap by $k-1$
- Quality improves as blocks are more reduced
- Cost grows as $2^{O(k^3)}$
Blocking to improve reducedness

- LLL and its variants achieve exponential (H)SVP approximation factors in polynomial-time
- Can we do better by paying more?

BKZ  (Block Korkine-Zolotarev)
- HKZ calls in dim. $k$, with blocks that overlap by $k - 1$
  - Quality improves as blocks are more reduced
  - Cost grows as $2^{O(k)}$
Blocking to improve reducedness

- LLL and its variants achieve exponential (H)SVP approximation factors in polynomial-time
- Can we do better by paying more?

**BKZ (Block Korkine-Zolotarev)**

- HKZ calls in dim. $k$, with blocks that overlap by $k - 1$
- Quality improves as blocks are more reduced
- Cost grows as $2^{O(k)}$
Blocking to improve reducedness

- LLL and its variants achieve exponential (H)SVP approximation factors in polynomial-time
- Can we do better by paying more?

BKZ (Block Korkine-Zolotarev)

- HKZ calls in dim. $k$, with blocks that overlap by $k - 1$
- Quality improves as blocks are more reduced
- Cost grows as $2^{O(k)}$
On a real example: BKZ40
On a real example: BKZ40
On a real example: BKZ40
On a real example: BKZ40
On a real example: BKZ40
On a real example: BKZ40
On a real example: BKZ40
On a real example: BKZ40
On a real example: BKZ40

\[
\log r_{i,i}
\]

\begin{figure}
\centering
\includegraphics[width=\textwidth]{image}
\end{figure}
On a real example: BKZ40
On a real example: BKZ40
On a real example: BKZ40
On a real example: BKZ40

![Graph showing \( \log r_{i,i} \) vs. \( i \)]
On a real example: BKZ40
On a real example: BKZ40
On a real example: BKZ40
On a real example: BKZ40
On a real example: BKZ40
On a real example: BKZ40
On a real example: BKZ40

\[ \log r_{i,i} \]

![Graph showing logarithmic decrease](image_url)
On a real example: BKZ40
On a real example: BKZ40

\[ \log r_{i,i} \]

\[ 0 \leq i \leq 100 \]
On a real example: BKZ40
On a real example: BKZ40
On a real example: BKZ40
On a real example: BKZ40
On a real example: BKZ40
On a real example: BKZ40
On a real example: BKZ70
On a real example: BKZ70
On a real example: BKZ70
On a real example: BKZ70

\[ \log r_{i,i} \]

- \( i \) ranges from 0 to 100.
On a real example: BKZ70

\[ \log r_{i,i} \]

\[ 0 \quad 20 \quad 40 \quad 60 \quad 80 \quad 100 \]

\[ 205 \quad 200 \quad 195 \]
On a real example: BKZ70
On a real example: BKZ70
On a real example: BKZ70
On a real example: BKZ70
On a real example: BKZ70
On a real example: BKZ70

\[ \log r_{i,i} \]

![Graph showing the logarithm of \( r_{i,i} \) over the index \( i \). The graph depicts a decreasing trend.]
On a real example: BKZ70

\[ \log r_{i,i} \]

\[ i \]

\[ 0 \]

\[ 20 \]

\[ 40 \]

\[ 60 \]

\[ 80 \]

\[ 100 \]

\[ 195 \]

\[ 200 \]

\[ 205 \]
On a real example: BKZ70
On a real example: BKZ70

\[ \log r_{i,i} \]

\[ r_{i,i} \]

\[ i \]

D. Stehlé

Lattice reduction algorithms

25/07/2017
On a real example: BKZ70
On a real example: BKZ70
On a real example: BKZ70

\[ \log r_{i,i} \]
On a real example: BKZ70

\[
\log r_{i,i}
\]

\(0\quad 20\quad 40\quad 60\quad 80\quad 100\)

\(195\quad 200\quad 205\)
On a real example: BKZ70
On a real example: BKZ70
On a real example: BKZ70

\[ \log r_{i,i} \]

where \( i \) ranges from 0 to 100.
On a real example: BKZ70
On a real example: BKZ70

\[ \log r_{i, i} \]

\[ \begin{align*}
0 & \quad 20 & \quad 40 & \quad 60 & \quad 80 & \quad 100 \\
0 & \quad 195 & \quad 200 & \quad 205
\end{align*} \]
On a real example: BKZ70

\[ \log r_{i,i} \]

\[ \begin{align*}
0 & \quad 20 \quad 40 \quad 60 \quad 80 \quad 100 \\
195 & \quad 200 & \quad 205
\end{align*} \]
On a real example: BKZ70

\[
\log r_{i,i}
\]
On a real example: BKZ70
On a real example: BKZ70
**BKZ, asymptotically**

<table>
<thead>
<tr>
<th></th>
<th>HKZ</th>
<th>BKZ$_k$</th>
<th>LLL $\approx$ BKZ$_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$|b_1|/(\det L)^{1/n}$</td>
<td>$\sqrt{n}$</td>
<td>$\sim k^{\frac{n}{2k}}$</td>
<td>$2^{O(n)}$</td>
</tr>
<tr>
<td>Time*</td>
<td>$2^{O(n)}$</td>
<td>$2^{O(k)} \times \text{Poly}(n)$</td>
<td>$\text{Poly}(n)$</td>
</tr>
</tbody>
</table>

*Omitting arithmetic costs

---

Lattice reduction rule of thumb (neglecting poly. factors)

<table>
<thead>
<tr>
<th>Time $2^{O(k)}$</th>
<th>approx. factor $\gamma = k^{O(n/k)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>or, equivalently</td>
<td>Approx. factor $\gamma$ costs time $\left(1 + \frac{n}{\log \gamma}\right)^{O\left(1 + \frac{n}{\log \gamma}\right)}$.</td>
</tr>
</tbody>
</table>
### BKZ, asymptotically

<table>
<thead>
<tr>
<th></th>
<th>HKZ</th>
<th>BKZ$_k$</th>
<th>LLL $\approx$ BKZ$_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$|b_1|/(\det L)^{\frac{1}{n}}$</td>
<td>$\sqrt{n}$</td>
<td>$\approx k^{\frac{n}{2k}}$</td>
<td>$2^{O(n)}$</td>
</tr>
<tr>
<td>Time*</td>
<td>$2^{O(n)}$</td>
<td>$2^{O(k)} \times \text{Poly}(n)$</td>
<td>$\text{Poly}(n)$</td>
</tr>
</tbody>
</table>

*Omitting arithmetic costs

### Lattice reduction rule of thumb (neglecting poly. factors)

Time $2^{O(k)} \implies$ approx. factor $\gamma = k^{O(n/k)}$

or, equivalently

Approx. factor $\gamma$ costs time $\left(1 + \frac{n}{\log \gamma}\right)^{O\left(1 + \frac{n}{\log \gamma}\right)}$. 

\[\begin{array}{|c|c|c|}
\hline
\text{HKZ} & \text{BKZ}_k & \text{LLL} \approx \text{BKZ}_2 \\
\hline
\|b_1\|/(\det L)^{\frac{1}{n}} & \sqrt{n} & \approx k^{\frac{n}{2k}} \\
\hline
\text{Time}* & 2^{O(n)} & 2^{O(k)} \times \text{Poly}(n) \\
\hline
\end{array}\]
Roadmap

1. Background on lattices
2. Solving the Shortest Vector Problem
3. The dynamics of lattice reduction
4. **Blocking techniques**
5. Approximations

**Open problems:**

1. Faster LLL-type reduction than $\tilde{O}(n^4 \log \|B\|)$
2. Accurate predictive model for BKZ$_k$ with large $k$
3. Good code for BKZ$_k$ with large $k$
Roadmap

1. Background on lattices
2. Solving the Shortest Vector Problem
3. The dynamics of lattice reduction
4. Blocking techniques
5. Approximations
Bit-complexity of LLL and practical run-time

Text-book LLL terminates in \( \tilde{O}(n^4 \log^2 \|B\|) \) bit operations

With MAGMA V2.19:

```maple
> n := 35; B := RMatrixSpace(Integers(),n,n)!0;
> for i:=1 to n do
> B[i][i]:=1; B[i][1]:=RandomBits(5000);
> end for;
> time C := LLL(B:Method:='Integral');
Time: 70.380
> time C := LLL(B);
Time: 1.560
```
The exact and approximate approaches

The exact approach

Integral Basis → Rational QR
↓ ↓ ↓ ↓ ↓ ↓ ...
↓ ↓ ↓ ↓ ↓ ↓ ...
↓ ↓ ↓ ↓ ↓ ↓ ...
↓ ↓ ↓ ↓ ↓ ↓ ...
↓ ↓ ↓ ↓ ↓ ↓ ...
... ...
...
...

We get a reduced basis... but
QR dominates the cost

The approximate approach

Integral Basis → Floating-pt QR
↓ ↓ ↓ ↓ ↓ ↓ ...
↓ ↓ ↓ ↓ ↓ ↓ ...
↓ ↓ ↓ ↓ ↓ ↓ ...
↓ ↓ ↓ ↓ ↓ ↓ ...
↓ ↓ ↓ ↓ ↓ ↓ ...
... ...
...
...

This is faster... but
This is highly unstable
The exact and approximate approaches

The exact approach

Integral Basis $\rightarrow$ Rational QR
$\downarrow$ $\downarrow$
$\downarrow$ $\downarrow$
$\downarrow$ $\downarrow$
$\downarrow$ $\downarrow$
$\downarrow$ $\downarrow$
$\vdots$ $\vdots$
$\vdots$ $\vdots$

We get a reduced basis... but

QR dominates the cost

The approximate approach

Integral Basis $\rightarrow$ Floating-pt QR
$\downarrow$ $\downarrow$
$\downarrow$ $\downarrow$
$\downarrow$ $\downarrow$
$\downarrow$ $\downarrow$
$\downarrow$ $\downarrow$
$\vdots$ $\vdots$
$\vdots$ $\vdots$

This is faster... but

This is highly unstable
The exact and approximate approaches

**The exact approach**

- Integral Basis $\rightarrow$ Rational QR
  - $\downarrow$
  - $\downarrow$
  - $\downarrow$ $\mathbb{Z}$-operations
  - $\downarrow$
  - $\downarrow$
  - $\downarrow$
  - $\vdots$
  - $\vdots$

We get a reduced basis... but

QR dominates the cost

**The approximate approach**

- Integral Basis $\rightarrow$ Floating-pt QR
  - $\downarrow$
  - $\downarrow$
  - $\downarrow$ $\mathbb{Z}$-operations
  - $\downarrow$
  - $\downarrow$
  - $\downarrow$
  - $\vdots$
  - $\vdots$

This is faster... but

This is highly unstable
The exact and approximate approaches

**The exact approach**

**Integral** Basis $\rightarrow$ **Rational** QR

$\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$

Z-operations $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$

$\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$

We get a reduced basis... but

QR dominates the cost

**The approximate approach**

**Integral** Basis $\rightarrow$ **Floating-pt** QR

$\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$

Z-operations $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$ $\rightarrow$

$\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$

This is faster... but

This is highly unstable
Odlyzko’s hybrid approach

<table>
<thead>
<tr>
<th>Integral</th>
<th>Floating-point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basis</td>
<td>QR</td>
</tr>
<tr>
<td>↓</td>
<td>↓</td>
</tr>
<tr>
<td>↓</td>
<td>↓</td>
</tr>
<tr>
<td>↓</td>
<td>↓</td>
</tr>
<tr>
<td>↓</td>
<td>↓</td>
</tr>
<tr>
<td>↓</td>
<td>↓</td>
</tr>
<tr>
<td>↓</td>
<td>↓</td>
</tr>
<tr>
<td>↓</td>
<td>↓</td>
</tr>
<tr>
<td>↓</td>
<td>↓</td>
</tr>
</tbody>
</table>
| ...
| ↓        | ↓             |

\[ \mathbb{Z}-\text{operations} \]

Numerical refreshing
Numerical QR-factorization

A rather well-studied topic:

- Many **backward stable** algorithms:
  Modified GS, Givens, Householder, ...

\[
B \rightarrow \bar{R}, \quad \text{the R-factor of } B + \Delta B
\]

- Backward stability may be combined with perturbation analysis, if the inputs are well-conditioned

\[
B = QR \quad \Rightarrow \quad B + \Delta B = (Q + \Delta Q)(R + \Delta R),
\]

where \( \Delta R \) grows as \( \text{“cond”}(R) \cdot \Delta B \).
Numerical QR-factorization

A rather well-studied topic:

- Many backward stable algorithms:
  Modified GS, Givens, Householder, ...

\[ B \rightarrow \bar{R}, \text{ the } R\text{-factor of } B + \Delta B \]

- Backward stability may be combined with perturbation analysis, if the inputs are well-conditioned

\[ B = QR \implies B + \Delta B = (Q + \Delta Q)(R + \Delta R), \]

where \( \Delta R \) grows as “cond”(R) \( \cdot \) \( \Delta B \).
Conditioning of $\mathbf{R}$

**Backward stability & sensitivity analysis**

⇒ approximation bounds

- Householder & co are backward stable for column-wise perturbations:
  \[
  \overline{\mathbf{R}} \text{ is the R-factor of } \mathbf{B} + \Delta \mathbf{B}, \\
  \max_i \frac{\|\Delta \mathbf{b}_i\|}{\|\mathbf{b}_i\|} \leq \text{Poly}(n) \cdot 2^{-p}.
  \]

- Perturbation analysis for columnwise perturbations
  \[
  \max_i \frac{\|\Delta \mathbf{r}_i\|}{\|\mathbf{r}_i\|} \leq \|\mathbf{R}\|\mathbf{R}^{-1}\| \cdot \max_i \frac{\|\Delta \mathbf{b}_i\|}{\|\mathbf{b}_i\|}.
  \]

If computing with precision $p \gg \log \|\mathbf{R}\|\mathbf{R}^{-1}\|$, then the computed $\overline{\mathbf{R}}$ is meaningful.
Backward stability & sensitivity analysis

⇒ approximation bounds

Householder & co are backward stable for column-wise perturbations:

\[ \overline{R} \text{ is the R-factor of } B + \Delta B, \]
\[ \max_i \frac{\|\Delta b_i\|}{\|b_i\|} \leq Poly(n) \cdot 2^{-p}. \]

Perturbation analysis for columnwise perturbations

\[ \max_i \frac{\|\Delta r_i\|}{\|r_i\|} \leq \|R\|\|R^{-1}\| \cdot \max_i \frac{\|\Delta b_i\|}{\|b_i\|}. \]

If computing with precision \( p \gg \log \|R\|\|R^{-1}\| \),
then the computed \( \overline{R} \) is meaningful.


**Conditioning of \( R \)**

Backward stability & sensitivity analysis

\[ \Rightarrow \text{approximation bounds} \]

- Householder & co are backward stable for column-wise perturbations:

\[
\overline{R} \text{ is the R-factor of } B + \Delta B, \\
\max_i \frac{\|\Delta b_i\|}{\|b_i\|} \leq \text{Poly}(n) \cdot 2^{-p}.
\]

- Perturbation analysis for columnwise perturbations

\[
\max_i \frac{\|\Delta r_i\|}{\|r_i\|} \leq \|R\|R^{-1}\| \cdot \max_i \frac{\|\Delta b_i\|}{\|b_i\|}.
\]

If computing with precision \( p \gg \log \|R\|R^{-1}\| \),
then the computed \( \overline{R} \) is meaningful.
 Conditioning of $R$

Backward stability & sensitivity analysis

$\Rightarrow$ approximation bounds

- Householder & co are backward stable for column-wise perturbations:

  $\overline{R}$ is the R-factor of $B + \Delta B$,

  $\max_i \|\Delta b_i\|/\|b_i\| \leq Poly(n) \cdot 2^{-p}$.

- Perturbation analysis for columnwise perturbations

  $\max_i \|\Delta r_i\|/\|r_i\| \leq \|R\|\|R^{-1}\| \cdot \max_i \|\Delta b_i\|/\|b_i\|$. 

If computing with precision $p \gg \log \|\|R\|\|R^{-1}\|$,
then the computed $\overline{R}$ is meaningful.
Conditioning and reducedness

\[
\text{“cond”}(\mathbf{R}) \leq \|\mathbf{R}\|\mathbf{R}^{-1}\|
\]

We are lucky! If \( \mathbf{B} \) is LLL-reduced, then \( \text{cond}(\mathbf{R}) \leq 2^{O(n)} \) and computing with \( p = O(n) \) suffices.

Need LLL-reducedness to LLL-reduce numerically...

Use a greedy LLL algorithm:

- Take the first \( i \) s.t. \((\mathbf{b}_1, \ldots, \mathbf{b}_i)\) is not LLL-reduced
  \(\Rightarrow (\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})\) is well-conditioned
- Work on \( \mathbf{b}_i \) until \((\mathbf{b}_1, \ldots, \mathbf{b}_i)\) is LLL-reduced or until we can decide to swap \( \mathbf{b}_i \) and \( \mathbf{b}_{i-1} \)
We are lucky! If \( B \) is LLL-reduced, then \( \text{cond}(R) \leq 2^{O(n)} \) and computing with \( p = O(n) \) suffices.

Need LLL-reducedness to LLL-reduce numerically...

Use a **greedy LLL algorithm**:

- Take the first \( i \) s.t. \( (b_1, \ldots, b_i) \) is not LLL-reduced
  \[ \Rightarrow (b_1, \ldots, b_{i-1}) \text{ is well-conditioned} \]
- Work on \( b_i \) until \( (b_1, \ldots, b_i) \) is LLL-reduced
  or until we can decide to swap \( b_i \) and \( b_{i-1} \)
Conditioning and reducedness

\[ \text{“cond”}(\mathbf{R}) \leq \| \mathbf{R} \| \| \mathbf{R}^{-1} \| \]

We are lucky! If \( \mathbf{B} \) is LLL-reduced, then \( \text{cond}(\mathbf{R}) \leq 2^{O(n)} \) and computing with \( p = O(n) \) suffices.

Need LLL-reducedness to LLL-reduce numerically...

Use a greedy LLL algorithm:

- Take the first \( i \) s.t. \( (\mathbf{b}_1, \ldots, \mathbf{b}_i) \) is not LLL-reduced
  \[ \Rightarrow (\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}) \] is well-conditioned
- Work on \( \mathbf{b}_i \) until \( (\mathbf{b}_1, \ldots, \mathbf{b}_i) \) is LLL-reduced or until we can decide to swap \( \mathbf{b}_i \) and \( \mathbf{b}_{i-1} \)
Bit complexity of hybrid LLL

Bit-complexity (left hand side is correct only in an amortized sense)

\[ O(n^2 \beta) \cdot O(n^2) \cdot \left[ O(n\beta) + O(n^2) \right] = O(n^5 \beta (n + \beta)) \]

1- loop iterations
2- arithmetic operations per loop iteration
3- integer arithmetic (on the basis)
4- floating-point arithmetic (on QR)

- Asymptotically: not much better than textbook LLL and worse than blocking
- In practice: \( p = 53 \) for \( n \) up to 150-200.

Lengthy rationals \( \rightarrow \) low-precision floating-points
Bit complexity of hybrid LLL

Bit-complexity

\[ O(n^2\beta) \cdot O(n^2) \cdot \left[ O(n\beta) + O(n^2) \right] = O \left( n^5 \beta(n + \beta) \right) \]

1- loop iterations
2- arithmetic operations per loop iteration
3- integer arithmetic (on the basis)
4- floating-point arithmetic (on QR)

- Asymptotically: not much better than text-book LLL and worse than blocking
- In practice: \( p = 53 \) for \( n \) up to 150-200.

Lengthy rationals \( \rightarrow \) low-precision floating-points
In hybrid-LLL, basis operations are dominating the cost.

- The sensitivity bounds imply that we can work with an approximation of $B$.
  ⇒ take the most significant bits!

- The transformation matrix is also bounded by $\|R\| R^{-1} \|$. 

- Round-reduce-update, recursively
  $\tilde{L}^1$ algorithm: $\tilde{O}(n^5 \log \|B\|)$ bit operations
Going further with approximations

In hybrid-LLL, basis operations are dominating the cost.

- The sensitivity bounds imply that we can work with an approximation of $B$.

  $\Rightarrow$ take the most significant bits!

- The transformation matrix is also bounded by $\|R\|\|R^{-1}\|$.

- Round-reduce-update, recursively
  \[ \tilde{\mathbb{L}}^1 \]
  algorithm: $\tilde{O}(n^5 \log \|B\|)$ bit operations
Going further with approximations

In hybrid-LLL, basis operations are dominating the cost.

- The sensitivity bounds imply that we can work with an approximation of $\mathbf{B}$.
  - $\implies$ take the most significant bits!
- The transformation matrix is also bounded by $||\mathbf{R}||\mathbf{R}^{-1}||$.

- Round-reduce-update, recursively
  - $\tilde{L}^1$ algorithm: $\tilde{O}(n^5 \log ||\mathbf{B}||)$ bit operations
# LLL-reduction: state of the art

<table>
<thead>
<tr>
<th></th>
<th>[Stor96]</th>
<th>[KoSc01]</th>
<th>[NoStVi11]</th>
<th>[NeSt16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow dynamics</td>
<td>slow dynamics</td>
<td>slow dynamics</td>
<td>slow dynamics</td>
<td>fast dynamics</td>
</tr>
<tr>
<td>Blocking</td>
<td>blocking</td>
<td>blocking</td>
<td>no blocking</td>
<td>blocking</td>
</tr>
<tr>
<td>Exact R</td>
<td>exact R</td>
<td>exact R</td>
<td>approximate B and R</td>
<td>exact R</td>
</tr>
<tr>
<td>Cost</td>
<td>$\tilde{O}(n^{3.39}\beta^2)$</td>
<td>$\tilde{O}(n^3\beta^2)$</td>
<td>$\tilde{O}(n^5\beta)$</td>
<td>$\tilde{O}(n^4\beta)$</td>
</tr>
<tr>
<td>HSVP</td>
<td>$(4/3 + \varepsilon)^{n-1/4}$</td>
<td>$2^{O(n \log n)}$</td>
<td>$(4/3 + \varepsilon)^{n-1/4}$</td>
<td>$(1 + \varepsilon)(4/3)^{n-1/4}$</td>
</tr>
</tbody>
</table>

$\beta := \log \|B\|$
Roadmap

1. Background on lattices
2. Solving the Shortest Vector Problem
3. The dynamics of lattice reduction
4. Blocking techniques
5. Approximations

Open problems:
1. Combine fast dynamics, blocking and approximations
2. Use less precision, in theory and in practice
Concluding remarks

Lattice reduction comes in two main flavours:
- Fast, with exponential approximation factors
- Slow, with shorter vectors

Both cases are very relevant for applications.

The set of algorithmic techniques is limited
- Dynamics
- Blocking
- Approximations

This is in contrast to, e.g., SVP algorithms.
Concluding remarks

Lattice reduction comes in two main flavours:

- Fast, with exponential approximation factors
- Slow, with shorter vectors

Both cases are very relevant for applications.

The set of algorithmic techniques is limited

- Dynamics
- Blocking
- Approximations

This is in contrast to, e.g., SVP algorithms.
My favourite open problems

- SVP solvers: make theory, heuristics and practice match
  - Prove lower bounds on the necessary number of steps towards reducedness
  - Combine fast dynamics, blocking and approximations to obtain a faster LLL-type algorithm
My favourite open problems

- SVP solvers: make theory, heuristics and practice match
- Prove lower bounds on the necessary number of steps towards reducedness
- Combine fast dynamics, blocking and approximations to obtain a faster LLL-type algorithm
My favourite open problems

- SVP solvers: make theory, heuristics and practice match
- Prove lower bounds on the necessary number of steps towards reducedness
- Combine fast dynamics, blocking and approximations to obtain a faster LLL-type algorithm