Lattice Reduction Algorithms:

EUCLID, GAUSS, LLL

Description and Probabilistic Analysis

Brigitte Vallée
(CNRS and Université de Caen, France)

Mauritanie, February 2016

The general problem of lattice reduction

A lattice of $\mathbb{R}^p$ = a discrete additive subgroup of $\mathbb{R}^p$.

A lattice $\mathcal{L}$ possesses a basis $B := (b_1, b_2, \ldots, b_n)$ with $n \le p$,

$$\mathcal{L} := \{x \in \mathbb{R}^p; \quad x = \sum_{i=1}^{n} x_i b_i, \qquad x_i \in \mathbb{Z}\}$$

... and in fact, an infinite number of bases....

If now $\mathbb{R}^p$ is endowed with its (canonical) Euclidean structure,
there exist bases (called reduced) with good Euclidean properties:
their vectors are short enough and almost orthogonal.

Lattice reduction Problem : From a lattice $\mathcal{L}$ given by a basis B,
construct from B a reduced basis $\widehat{B}$ of $\mathcal{L}$.

Many applications of this problem in various domains:
number theory, arithmetics, discrete geometry..... and cryptology.

The general problem of lattice reduction

A lattice of $\mathbb{R}^p$ = a discrete additive subgroup of $\mathbb{R}^p$.

A lattice $\mathcal{L}$ possesses a basis $B := (b_1, b_2, \ldots, b_n)$ with $n \le p$,

$$\mathcal{L} := \{x \in \mathbb{R}^p; \quad x = \sum_{i=1}^{n} x_i b_i, \qquad x_i \in \mathbb{Z}\}$$

... and in fact, an infinite number of bases....

If now $\mathbb{R}^p$ is endowed with its (canonical) Euclidean structure,
there exist bases (called reduced) with good Euclidean properties:
their vectors are short enough and almost orthogonal.

Lattice reduction Problem : From a lattice $\mathcal{L}$ given by a basis B,
construct from B a reduced basis $\widehat{B}$ of $\mathcal{L}$.

Many applications of this problem in various domains:
number theory, arithmetics, discrete geometry..... and cryptology.

The general problem of lattice reduction

A lattice of $\mathbb{R}^p$ = a discrete additive subgroup of $\mathbb{R}^p$.

A lattice $\mathcal{L}$ possesses a basis $B := (b_1, b_2, \ldots, b_n)$ with $n \leq p$,

$$\mathcal{L} := \{x \in \mathbb{R}^p; \quad x = \sum_{i=1}^{n} x_i b_i, \qquad x_i \in \mathbb{Z}\}$$

... and in fact, an infinite number of bases....

If now $\mathbb{R}^p$ is endowed with its (canonical) Euclidean structure,
there exist bases (called reduced) with good Euclidean properties:
their vectors are short enough and almost orthogonal.

Lattice reduction Problem : From a lattice $\mathcal{L}$ given by a basis B,
construct from B a reduced basis $\widehat{B}$ of $\mathcal{L}$.

Many applications of this problem in various domains:
number theory, arithmetics, discrete geometry..... and cryptology.

The general problem of lattice reduction

A lattice of $\mathbb{R}^p$ = a discrete additive subgroup of $\mathbb{R}^p$.

A lattice $\mathcal{L}$ possesses a basis $B := (b_1, b_2, \ldots, b_n)$ with $n \le p$,

$$\mathcal{L} := \{x \in \mathbb{R}^p; \quad x = \sum_{i=1}^{n} x_i b_i, \qquad x_i \in \mathbb{Z}\}$$

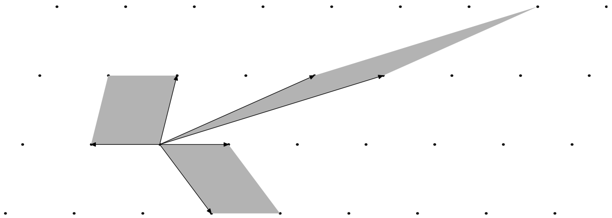
... and in fact, an infinite number of bases....

If now $\mathbb{R}^p$ is endowed with its (canonical) Euclidean structure,
there exist bases (called reduced) with good Euclidean properties:
their vectors are short enough and almost orthogonal.

Lattice reduction Problem : From a lattice $\mathcal{L}$ given by a basis B,
construct from B a reduced basis $\widehat{B}$ of $\mathcal{L}$.

Many applications of this problem in various domains:
number theory, arithmetics, discrete geometry..... and cryptology.

Lattice reduction algorithms in the two dimensional case.



III- The LLL algorithm.

III-1. The main objects of a lattice.

III-2. Principles of the LLL algorithm

III-3. Description and analysis of the algorithm

III-4. Properties of the output

III-5. Examples of problems solved by the LLL algorithm

III-6. Simplified models.

Main objects of a lattice (I)

Two reference parameters :

– the determinant and the successive minima.

– The first minimum $\lambda(\mathcal{L})$ is the norm of a shortest non-zero vector.

– The determinant $\det \mathcal{L} := \det G(\boldsymbol{b})$ with $G(\boldsymbol{b}) := ((b_i, b_j))_{i,j}$.

independent of the basis $\boldsymbol{b}$.

When the lattice is given by a basis $\boldsymbol{b}$, it is

– easy to compute the determinant.

– (probably) difficult to compute a shortest non zero vector.

Minkowski's Theorem relates $\lambda(\mathcal{L})$ and $\det \mathcal{L}$:

For any n, there is a constant γ_n, s.t, for any $\mathcal{L}$ of dimension n,

$$\lambda(\mathcal{L})^2 \leq \gamma_n \left[\det(\mathcal{L})\right]^{1/n}$$

γ_n has a polynomial growth wrt n.

Main objects of a lattice (I)

Two reference parameters :

– the determinant and the successive minima.

– The first minimum $\lambda(\mathcal{L})$ is the norm of a shortest non-zero vector.

– The determinant $\det \mathcal{L} := \det G(\boldsymbol{b})$ with $G(\boldsymbol{b}) := ((b_i, b_j))_{i,j}$.

independent of the basis $\boldsymbol{b}$.

When the lattice is given by a basis $\boldsymbol{b}$, it is

– easy to compute the determinant.

– (probably) difficult to compute a shortest non zero vector.

Minkowski's Theorem relates $\lambda(\mathcal{L})$ and $\det \mathcal{L}$:

For any n, there is a constant γ_n, s.t, for any $\mathcal{L}$ of dimension n,

$$\lambda(\mathcal{L})^2 \leq \gamma_n \left[\det(\mathcal{L})\right]^{1/n}$$

γ_n has a polynomial growth wrt n.

Main objects of a lattice (I)

Two reference parameters :

– the determinant and the successive minima.

– The first minimum $\lambda(\mathcal{L})$ is the norm of a shortest non-zero vector.

– The determinant $\det \mathcal{L} := \det G(\boldsymbol{b})$ with $G(\boldsymbol{b}) := ((b_i, b_j))_{i,j}$.

independent of the basis $\boldsymbol{b}$.

When the lattice is given by a basis $\boldsymbol{b}$, it is

– easy to compute the determinant.

– (probably) difficult to compute a shortest non zero vector.

Minkowski's Theorem relates $\lambda(\mathcal{L})$ and $\det \mathcal{L}$:

For any n, there is a constant γ_n, s.t, for any $\mathcal{L}$ of dimension n,

$$\lambda(\mathcal{L})^2 \le \gamma_n \left[\det(\mathcal{L})\right]^{1/n}$$

γ_n has a polynomial growth wrt n.

Main objects of a lattice (II)
Successive minima

$\lambda_i(\mathcal{L}) := \min\{r \mid$ the ball $B(0,r)$ contains at least i independent vectors of $\mathcal{L}$ $\}$

Second theorem of Minkowski:

For any n, there is a constant γ_n, s.t, for any $\mathcal{L}$ of dimension n,

$$\prod_{i=1}^{n} \lambda_i(\mathcal{L})^2 \leq \gamma_n^n \det(\mathcal{L})$$

γ_n has a polynomial growth wrt n.

For $n \geq 4$, a lattice $\mathcal{L}$ does not always admit a minimal basis.

Example of such a lattice

$$\mathcal{L} := \{(x_i)_{1\leq i\leq 5} \mid x_1 \equiv x_2 \equiv x_3 \equiv x_4 \equiv x_5 \mod 2\}$$

Main objects of a lattice (II)
Successive minima

$\lambda_i(\mathcal{L}) := \min\{r \mid \text{the ball } B(0,r) \text{ contains at least } i \text{ independent vectors of } \mathcal{L}\ \}$

Second theorem of Minkowski:

For any n, there is a constant γ_n, s.t, for any $\mathcal{L}$ of dimension n,

$$\prod_{i=1}^{n} \lambda_i(\mathcal{L})^2 \leq \gamma_n^n \det(\mathcal{L})$$

γ_n has a polynomial growth wrt n.

For $n \geq 4$, a lattice $\mathcal{L}$ does not always admit a minimal basis.

Example of such a lattice

$$\mathcal{L} := \{(x_i)_{1 \leq i \leq 5} \mid x_1 \equiv x_2 \equiv x_3 \equiv x_4 \equiv x_5 \mod 2\}$$

Main objects of a lattice (II)
Successive minima

$\lambda_i(\mathcal{L}) := \min\{r \mid \text{the ball } B(0,r) \text{ contains at least } i \text{ independent vectors of } \mathcal{L}\ \}$

Second theorem of Minkowski:

For any n, there is a constant γ_n, s.t, for any $\mathcal{L}$ of dimension n,

$$\prod_{i=1}^{n} \lambda_i(\mathcal{L})^2 \leq \gamma_n^n \det(\mathcal{L})$$

γ_n has a polynomial growth wrt n.

For $n \geq 4$, a lattice $\mathcal{L}$ does not always admit a minimal basis.

Example of such a lattice

$$\mathcal{L} := \{(x_i)_{1\leq i \leq 5} \mid x_1 \equiv x_2 \equiv x_3 \equiv x_4 \equiv x_5 \mod 2\}$$

Main objects of a lattice (II)
Successive minima

$\lambda_i(\mathcal{L}) := \min\{r \mid \text{the ball } B(0,r) \text{ contains at least } i \text{ independent vectors of } \mathcal{L}\ \}$

Second theorem of Minkowski:

For any n, there is a constant γ_n, s.t, for any $\mathcal{L}$ of dimension n,

$$\prod_{i=1}^{n} \lambda_i(\mathcal{L})^2 \leq \gamma_n^n \det(\mathcal{L})$$

γ_n has a polynomial growth wrt n.

For $n \geq 4$, a lattice $\mathcal{L}$ does not always admit a minimal basis.

Example of such a lattice

$$\mathcal{L} := \{(x_i)_{1 \leq i \leq 5} \mid x_1 \equiv x_2 \equiv x_3 \equiv x_4 \equiv x_5 \mod 2\}$$

Algorithmic difficulty of main lattice problems.

Algorithmic framework:

– A lattice $\mathcal{L}$ of dimension n is given by an integer basis $\boldsymbol{b}$ of length $M := \max \|b_i\|^2$. The input size is $O(n \log M)$.

– It is easy to compute $\det \mathcal{L}$ in polynomial-time in $O(n \log M)$.

– However, it is probably difficult to compute a shortest non zero vector.

Shortest Vector Problem [SVP]. *Given a basis* $\boldsymbol{b}$ *of a lattice* $\mathcal{L}$*, find a non-zero vector* v *of* $\mathcal{L}$ *that satisfies* $\|v\| = \lambda(\mathcal{L})$.

– This problem is only known to be NP–hard for randomized reductions

– It is closely surrounded by problems that are proven to be NP–hard

This leads to consider approximate versions of the SVP Problem:

Problem γ–SVP. *Given a basis* $\boldsymbol{b}$ *of a lattice* $\mathcal{L}$*, find a short enough vector* v *that satisfies* $\|v\| \le \gamma\lambda(\mathcal{L})$.

There exist algorithms that solve this problem in polynomial-time when the approximation factor is $\gamma = 2^{O(n)}$.

The LLL algorithm is such an approximation algorithm.

Algorithmic difficulty of main lattice problems.

Algorithmic framework:

– A lattice $\mathcal{L}$ of dimension n is given by an integer basis $\boldsymbol{b}$ of length $M := \max \|b_i\|^2$. The input size is $O(n \log M)$.

– It is easy to compute $\det \mathcal{L}$ in polynomial-time in $O(n \log M)$.

– However, it is probably difficult to compute a shortest non zero vector.

Shortest Vector Problem [SVP]. *Given a basis $\boldsymbol{b}$ of a lattice $\mathcal{L}$, find a non-zero vector v of $\mathcal{L}$ that satisfies $\|v\| = \lambda(\mathcal{L})$.*

– This problem is only known to be NP–hard for randomized reductions

– It is closely surrounded by problems that are proven to be NP–hard

This leads to consider approximate versions of the SVP Problem:

Problem γ–SVP. *Given a basis $\boldsymbol{b}$ of a lattice $\mathcal{L}$, find a short enough vector v that satisfies $\|v\| \leq \gamma\lambda(\mathcal{L})$.*

There exist algorithms that solve this problem in polynomial-time when the approximation factor is $\gamma = 2^{O(n)}$.

The LLL algorithm is such an approximation algorithm.

Algorithmic difficulty of main lattice problems.

Algorithmic framework:

– A lattice $\mathcal{L}$ of dimension n is given by an integer basis $\boldsymbol{b}$ of length $M := \max \|b_i\|^2$. The input size is $O(n \log M)$.

– It is easy to compute $\det \mathcal{L}$ in polynomial-time in $O(n \log M)$.

– However, it is probably difficult to compute a shortest non zero vector.

Shortest Vector Problem [SVP]. *Given a basis* $\boldsymbol{b}$ *of a lattice* $\mathcal{L}$*, find a non-zero vector* v *of* $\mathcal{L}$ *that satisfies* $\|v\| = \lambda(\mathcal{L})$.

– This problem is only known to be NP–hard for randomized reductions

– It is closely surrounded by problems that are proven to be NP–hard

This leads to consider approximate versions of the SVP Problem:

Problem γ–SVP. *Given a basis* $\boldsymbol{b}$ *of a lattice* $\mathcal{L}$*, find a short enough vector* v *that satisfies* $\|v\| \le \gamma\lambda(\mathcal{L})$.

There exist algorithms that solve this problem in polynomial-time when the approximation factor is $\gamma = 2^{O(n)}$.

The LLL algorithm is such an approximation algorithm.

Algorithmic difficulty of main lattice problems.

Algorithmic framework:

– A lattice $\mathcal{L}$ of dimension n is given by an integer basis $\boldsymbol{b}$ of length $M := \max \|b_i\|^2$. The input size is $O(n \log M)$.

– It is easy to compute $\det \mathcal{L}$ in polynomial-time in $O(n \log M)$.

– However, it is probably difficult to compute a shortest non zero vector.

Shortest Vector Problem [SVP]. *Given a basis* $\boldsymbol{b}$ *of a lattice* $\mathcal{L}$*, find a non-zero vector* v *of* $\mathcal{L}$ *that satisfies* $\|v\| = \lambda(\mathcal{L})$.

– This problem is only known to be NP–hard for randomized reductions

– It is closely surrounded by problems that are proven to be NP–hard

This leads to consider approximate versions of the SVP Problem:

Problem γ–SVP. *Given a basis* $\boldsymbol{b}$ *of a lattice* $\mathcal{L}$*, find a short enough vector* v *that satisfies* $\|v\| \leq \gamma\lambda(\mathcal{L})$.

There exist algorithms that solve this problem in polynomial-time when the approximation factor is $\gamma = 2^{O(n)}$.

The LLL algorithm is such an approximation algorithm.

Algorithmic difficulty of main lattice problems.

Algorithmic framework:

– A lattice $\mathcal{L}$ of dimension n is given by an integer basis $\boldsymbol{b}$ of length $M := \max \|b_i\|^2$. The input size is $O(n \log M)$.

– It is easy to compute $\det \mathcal{L}$ in polynomial-time in $O(n \log M)$.

– However, it is probably difficult to compute a shortest non zero vector.

Shortest Vector Problem [SVP]. *Given a basis $\boldsymbol{b}$ of a lattice $\mathcal{L}$, find a non-zero vector v of $\mathcal{L}$ that satisfies $\|v\| = \lambda(\mathcal{L})$.*

– This problem is only known to be NP–hard for randomized reductions

– It is closely surrounded by problems that are proven to be NP–hard

This leads to consider approximate versions of the SVP Problem:

Problem γ–SVP. *Given a basis $\boldsymbol{b}$ of a lattice $\mathcal{L}$, find a short enough vector v that satisfies $\|v\| \le \gamma\lambda(\mathcal{L})$.*

There exist algorithms that solve this problem in polynomial-time when the approximation factor is $\gamma = 2^{O(n)}$.

The LLL algorithm is such an approximation algorithm.

Algorithmic difficulty of main lattice problems.

Algorithmic framework:

– A lattice $\mathcal{L}$ of dimension n is given by an integer basis $\boldsymbol{b}$ of length $M := \max \|b_i\|^2$. The input size is $O(n \log M)$.

– It is easy to compute $\det \mathcal{L}$ in polynomial-time in $O(n \log M)$.

– However, it is probably difficult to compute a shortest non zero vector.

> Shortest Vector Problem [SVP]. *Given a basis $\boldsymbol{b}$ of a lattice $\mathcal{L}$, find a non-zero vector v of $\mathcal{L}$ that satisfies $\|v\| = \lambda(\mathcal{L})$.*

– This problem is only known to be NP–hard for randomized reductions

– It is closely surrounded by problems that are proven to be NP–hard

This leads to consider approximate versions of the SVP Problem:

> Problem γ–SVP. *Given a basis $\boldsymbol{b}$ of a lattice $\mathcal{L}$, find a short enough vector v that satisfies $\|v\| \leq \gamma\lambda(\mathcal{L})$.*

There exist algorithms that solve this problem in polynomial-time when the approximation factor is $\gamma = 2^{O(n)}$.

The LLL algorithm is such an approximation algorithm.

What can be expected about a good basis?

Important role played by the Gram-Schmidt orthogonalized system :

$B^\star = (b_1^\star, b_2^\star, \dots, b_n^\star)$ with $b_i^\star :=$ proj. of b_i orth. to $< b_1, b_2, \dots b_{i-1} >$

– together with the matrix $\mathcal{P}$ which expresses B as a function of $B^\star$

– its coefficients $m_{i,j} := \dfrac{(b_i, b_j^\star)}{\|b_j^\star\|}$

$$\mathcal{P} := \begin{array}{cc} & \begin{array}{cccccccc} b_1^\star & b_2^\star & \dots & b_{i-1}^\star & b_i^\star & b_{i+1}^\star & \dots & b_n^\star \end{array} \\ \begin{array}{c} b_1 \\ b_2 \\ \vdots \\ b_{i-1} \\ b_i \\ b_{i+1} \\ \vdots \\ b_n \end{array} & \begin{pmatrix} 1 & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\ m_{2,1} & 1 & \dots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ m_{i-1,1} & m_{i-1,2} & \dots & 1 & 0 & 0 & 0 & 0 \\ m_{i,1} & m_{i,2} & \dots & m_{i,i-1} & 1 & 0 & 0 & 0 \\ m_{i+1,1} & m_{i+1,2} & \dots & m_{i+1,i-1} & m_{i+1,i} & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{n,1} & m_{n,2} & \dots & m_{n,i-1} & m_{n,i} & m_{n,i+1} & \dots & 1 \end{pmatrix} \end{array}$$

What can be expected about a good basis?

Important role played by the Gram-Schmidt orthogonalized system :

$B^\star = (b_1^\star, b_2^\star, \ldots, b_n^\star)$ with $b_i^\star :=$ proj. of b_i orth. to $< b_1, b_2, \ldots b_{i-1} >$

– together with the matrix $\mathcal{P}$ which expresses B as a function of $B^\star$

– its coefficients $m_{i,j} := \dfrac{(b_i, b_j^\star)}{\|b_j^\star\|}$

$$\mathcal{P} := \begin{array}{cc}
 & \begin{array}{cccccccc} b_1^\star & b_2^\star & \ldots & b_{i-1}^\star & b_i^\star & b_{i+1}^\star & \ldots & b_n^\star \end{array} \\
\begin{array}{c} b_1 \\ b_2 \\ \vdots \\ b_{i-1} \\ b_i \\ b_{i+1} \\ \vdots \\ b_n \end{array} &
\begin{pmatrix}
1 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 \\
m_{2,1} & 1 & \ldots & 0 & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
m_{i-1,1} & m_{i-1,2} & \ldots & 1 & 0 & 0 & 0 & 0 \\
m_{i,1} & m_{i,2} & \ldots & m_{i,i-1} & 1 & 0 & 0 & 0 \\
m_{i+1,1} & m_{i+1,2} & \ldots & m_{i+1,i-1} & m_{i+1,i} & 1 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
m_{n,1} & m_{n,2} & \ldots & m_{n,i-1} & m_{n,i} & m_{n,i+1} & \ldots & 1
\end{pmatrix}
\end{array}$$

What can be expected about a good basis? (III)

The lengths $\ell_i := ||b_i^\star||$, the ratios $y_i := \dfrac{\ell_{i+1}}{\ell_i}$, the interval $[a := \min \ell_i, A := \max \ell_i]$.

For any basis, the interval $[a, A]$ provides an approximation of $\lambda(\mathcal{L})$ and $\det \mathcal{L}$:

$$\lambda(\mathcal{L}) \geq a, \qquad \lambda(\mathcal{L}) \leq A\sqrt{n}, \qquad a^n \leq \det \mathcal{L} \leq A^n$$

Two actions performed by the algorithm.

– It size-reduces the basis $\mathcal{P}$: the final coefficients satisfies $|\widehat{m}_{i,j}| \leq 1/2$

– It narrows the interval $[a, A]$ and increases the Siegel ratios :
the final ratios $\widehat{y}_i$ satisfy $\widehat{y}_i \geq \sigma$ with $\sigma \leq \sqrt{3}/2$

Two properties for the output basis.

Length defect: $\quad ||\widehat{b}_1|| \leq \left(\dfrac{1}{\sigma}\right)^{n-1} \lambda(\mathcal{L})$

Orthogonality defect: $\quad \displaystyle\prod_{i=1}^{n-1} \frac{||\widehat{b}_i||}{||\widehat{b}_i^\star||} \leq \left(\frac{1}{\sigma}\right)^{n(n-1)}$

What can be expected about a good basis? (III)

The lengths $\ell_i := ||b_i^\star||$, the ratios $y_i := \dfrac{\ell_{i+1}}{\ell_i}$, the interval $[a := \min \ell_i, A := \max \ell_i]$.

For any basis, the interval $[a, A]$ provides an approximation of $\lambda(\mathcal{L})$ and $\det \mathcal{L}$:

$$\lambda(\mathcal{L}) \geq a, \qquad \lambda(\mathcal{L}) \leq A\sqrt{n}, \qquad a^n \leq \det \mathcal{L} \leq A^n$$

Two actions performed by the algorithm.

– It size-reduces the basis $\mathcal{P}$: the final coefficients satisfies $|\widehat{m}_{i,j}| \leq 1/2$

– It narrows the interval $[a, A]$ and increases the Siegel ratios :

the final ratios $\widehat{y}_i$ satisfy $\widehat{y}_i \geq \sigma$ with $\sigma \leq \sqrt{3}/2$

Two properties for the output basis.

Length defect: $\quad ||\widehat{b}_1|| \leq \left(\dfrac{1}{\sigma}\right)^{n-1} \lambda(\mathcal{L})$

Orthogonality defect: $\quad \displaystyle\prod_{i=1}^{n-1} \frac{||\widehat{b}_i||}{||\widehat{b}_i^\star||} \leq \left(\frac{1}{\sigma}\right)^{n(n-1)}$

What can be expected about a good basis? (III)

The lengths $\ell_i := ||b_i^\star||$, the ratios $y_i := \dfrac{\ell_{i+1}}{\ell_i}$, the interval $[a := \min \ell_i, A := \max \ell_i]$.

For any basis, the interval $[a, A]$ provides an approximation of $\lambda(\mathcal{L})$ and $\det \mathcal{L}$:

$$\lambda(\mathcal{L}) \geq a, \qquad \lambda(\mathcal{L}) \leq A\sqrt{n}, \qquad a^n \leq \det \mathcal{L} \leq A^n$$

Two actions performed by the algorithm.

– It size-reduces the basis $\mathcal{P}$: the final coefficients satisfies $|\widehat{m}_{i,j}| \leq 1/2$

– It narrows the interval $[a, A]$ and increases the Siegel ratios :

the final ratios $\widehat{y}_i$ satisfy $\widehat{y}_i \geq \sigma$ with $\sigma \leq \sqrt{3}/2$

Two properties for the output basis.

Length defect: $\quad ||\widehat{b}_1|| \leq \left(\dfrac{1}{\sigma}\right)^{n-1} \lambda(\mathcal{L})$

Orthogonality defect: $\quad \displaystyle\prod_{i=1}^{n-1} \frac{||\widehat{b}_i||}{||\widehat{b}_i^\star||} \leq \left(\frac{1}{\sigma}\right)^{n(n-1)}$

What can be expected about a good basis? (III)

The lengths $\ell_i := ||b_i^\star||$, the ratios $y_i := \dfrac{\ell_{i+1}}{\ell_i}$, the interval $[a := \min \ell_i, A := \max \ell_i]$.

For any basis, the interval $[a, A]$ provides an approximation of $\lambda(\mathcal{L})$ and $\det \mathcal{L}$:

$$\lambda(\mathcal{L}) \geq a, \qquad \lambda(\mathcal{L}) \leq A\sqrt{n}, \qquad a^n \leq \det \mathcal{L} \leq A^n$$

Two actions performed by the algorithm.

– It size-reduces the basis $\mathcal{P}$: the final coefficients satisfies $|\widehat{m}_{i,j}| \leq 1/2$

– It narrows the interval $[a, A]$ and increases the Siegel ratios :

the final ratios $\widehat{y}_i$ satisfy $\widehat{y}_i \geq \sigma$ with $\sigma \leq \sqrt{3}/2$

Two properties for the output basis.

Length defect: $\qquad ||\widehat{b}_1|| \leq \left(\dfrac{1}{\sigma}\right)^{n-1} \lambda(\mathcal{L})$

Orthogonality defect: $\qquad \displaystyle\prod_{i=1}^{n-1} \frac{||\widehat{b}_i||}{||\widehat{b}_i^\star||} \leq \left(\frac{1}{\sigma}\right)^{n(n-1)}$

What can be expected about a good basis? (III)

The lengths $\ell_i := ||b_i^\star||$, the ratios $y_i := \dfrac{\ell_{i+1}}{\ell_i}$, the interval $[a := \min \ell_i, A := \max \ell_i]$.

For any basis, the interval $[a, A]$ provides an approximation of $\lambda(\mathcal{L})$ and $\det \mathcal{L}$:

$$\lambda(\mathcal{L}) \geq a, \qquad \lambda(\mathcal{L}) \leq A\sqrt{n}, \qquad a^n \leq \det \mathcal{L} \leq A^n$$

Two actions performed by the algorithm.

– It size-reduces the basis $\mathcal{P}$: the final coefficients satisfies $|\widehat{m}_{i,j}| \leq 1/2$

– It narrows the interval $[a, A]$ and increases the Siegel ratios :
  the final ratios $\widehat{y}_i$ satisfy $\widehat{y}_i \geq \sigma$ with $\sigma \leq \sqrt{3}/2$

Two properties for the output basis.

Length defect: $\quad ||\widehat{b}_1|| \leq \left(\dfrac{1}{\sigma}\right)^{n-1} \lambda(\mathcal{L})$

Orthogonality defect: $\quad \displaystyle\prod_{i=1}^{n-1} \frac{||\widehat{b}_i||}{||\widehat{b}_i^\star||} \leq \left(\frac{1}{\sigma}\right)^{n(n-1)}$

What can be expected about a good basis? (III)

The lengths $\ell_i := ||b_i^\star||$, the ratios $y_i := \dfrac{\ell_{i+1}}{\ell_i}$, the interval $[a := \min \ell_i, A := \max \ell_i]$.

For any basis, the interval $[a, A]$ provides an approximation of $\lambda(\mathcal{L})$ and $\det \mathcal{L}$:

$$\lambda(\mathcal{L}) \geq a, \qquad \lambda(\mathcal{L}) \leq A\sqrt{n}, \qquad a^n \leq \det \mathcal{L} \leq A^n$$

Two actions performed by the algorithm.

– It size-reduces the basis $\mathcal{P}$: the final coefficients satisfies $|\widehat{m}_{i,j}| \leq 1/2$

– It narrows the interval $[a, A]$ and increases the Siegel ratios :

the final ratios $\widehat{y}_i$ satisfy $\widehat{y}_i \geq \sigma$ with $\sigma \leq \sqrt{3}/2$

Two properties for the output basis.

$$\text{Length defect:} \qquad ||\widehat{b}_1|| \leq \left(\frac{1}{\sigma}\right)^{n-1} \lambda(\mathcal{L})$$

$$\text{Orthogonality defect:} \qquad \prod_{i=1}^{n-1} \frac{||\widehat{b}_i||}{||\widehat{b}_i^\star||} \leq \left(\frac{1}{\sigma}\right)^{n(n-1)}$$

The LLL algorithm

Input : A lattice $\mathcal{L}$ given by a basis $B = (b_1, b_2, \ldots, b_n)$

The algorithm deals with the Gram–Schmidt orthogonalized system $B^\star$ and the matrix $\mathcal{P} := (m_{i,j})$ which expresses B as a function of $B^\star$.

$$\mathcal{P} := \begin{array}{c} \\ b_1 \\ b_2 \\ \vdots \\ b_{i-1} \\ b_i \\ b_{i+1} \\ \vdots \\ b_p \end{array} \begin{array}{c} \begin{array}{cccccccc} b_1^\star & b_2^\star & \ldots & b_{i-1}^\star & b_i^\star & b_{i+1}^\star & \ldots & b_p^\star \end{array} \\ \begin{pmatrix} 1 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 \\ m_{2,1} & 1 & \ldots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ m_{i-1,1} & m_{i-1,2} & \ldots & 1 & 0 & 0 & 0 & 0 \\ m_{i,1} & m_{i,2} & \ldots & m_{i,i-1} & 1 & 0 & 0 & 0 \\ m_{i+1,1} & m_{i+1,2} & \ldots & m_{i+1,i-1} & m_{i+1,i} & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{p,1} & m_{p,2} & \ldots & m_{p,i-1} & m_{p,i} & m_{p,i+1} & \ldots & 1 \end{pmatrix} \end{array}$$

LLL algorithm =

Gauss' reduction steps on local bases

$$U_i := \begin{array}{c} \\ u_i \\ v_i \end{array} \begin{array}{c} \begin{array}{cc} b_i^\star & b_{i+1}^\star \end{array} \\ \begin{pmatrix} 1 & 0 \\ m_{i+1,i} & 1 \end{pmatrix} \end{array}$$

The LLL algorithm

Input : A lattice $\mathcal{L}$ given by a basis $B = (b_1, b_2, \ldots, b_n)$

The algorithm deals with the Gram–Schmidt orthogonalized system $B^\star$ and the matrix $\mathcal{P} := (m_{i,j})$ which expresses B as a function of $B^\star$.

$$\mathcal{P} := \begin{array}{c} \\ b_1 \\ b_2 \\ \vdots \\ b_{i-1} \\ b_i \\ b_{i+1} \\ \vdots \\ b_p \end{array} \begin{array}{c} \begin{array}{cccccccc} b_1^\star & b_2^\star & \ldots & b_{i-1}^\star & b_i^\star & b_{i+1}^\star & \ldots & b_p^\star \end{array} \\ \begin{pmatrix} 1 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 \\ m_{2,1} & 1 & \ldots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ m_{i-1,1} & m_{i-1,2} & \ldots & 1 & 0 & 0 & 0 & 0 \\ m_{i,1} & m_{i,2} & \ldots & m_{i,i-1} & 1 & 0 & 0 & 0 \\ m_{i+1,1} & m_{i+1,2} & \ldots & m_{i+1,i-1} & m_{i+1,i} & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{p,1} & m_{p,2} & \ldots & m_{p,i-1} & m_{p,i} & m_{p,i+1} & \ldots & 1 \end{pmatrix} \end{array}$$

LLL algorithm =
Gauss' reduction steps on local bases

$$U_i := \begin{array}{c} \\ u_i \\ v_i \end{array} \begin{array}{c} \begin{array}{cc} b_i^\star & b_{i+1}^\star \end{array} \\ \begin{pmatrix} 1 & 0 \\ m_{i+1,i} & 1 \end{pmatrix} \end{array}$$

The LLL algorithm

Input : A lattice $\mathcal{L}$ given by a basis $B = (b_1, b_2, \ldots, b_n)$

The algorithm deals with the Gram–Schmidt orthogonalized system $B^\star$ and the matrix $\mathcal{P} := (m_{i,j})$ which expresses B as a function of $B^\star$.

$$\mathcal{P} := \begin{array}{c} \\ b_1 \\ b_2 \\ \vdots \\ b_{i-1} \\ b_i \\ b_{i+1} \\ \vdots \\ b_p \end{array} \begin{array}{c} \begin{array}{cccccccc} b_1^\star & b_2^\star & \ldots & b_{i-1}^\star & b_i^\star & b_{i+1}^\star & \ldots & b_p^\star \end{array} \\ \begin{pmatrix} 1 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 \\ m_{2,1} & 1 & \ldots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ m_{i-1,1} & m_{i-1,2} & \ldots & 1 & 0 & 0 & 0 & 0 \\ m_{i,1} & m_{i,2} & \ldots & m_{i,i-1} & 1 & 0 & 0 & 0 \\ m_{i+1,1} & m_{i+1,2} & \ldots & m_{i+1,i-1} & m_{i+1,i} & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{p,1} & m_{p,2} & \ldots & m_{p,i-1} & m_{p,i} & m_{p,i+1} & \ldots & 1 \end{pmatrix} \end{array}$$

LLL algorithm =
Gauss' reduction steps on local bases

$$U_i := \begin{array}{c} \\ u_i \\ v_i \end{array} \begin{array}{c} \begin{array}{cc} b_i^\star & b_{i+1}^\star \end{array} \\ \begin{pmatrix} 1 & 0 \\ m_{i+1,i} & 1 \end{pmatrix} \end{array}$$

The LLL algorithm

Input : A lattice $\mathcal{L}$ given by a basis $B = (b_1, b_2, \ldots, b_n)$

The algorithm deals with the Gram–Schmidt orthogonalized system $B^\star$ and the matrix $\mathcal{P} := (m_{i,j})$ which expresses B as a function of $B^\star$.

$$\mathcal{P} := \begin{array}{c} \\ b_1 \\ b_2 \\ \vdots \\ b_{i-1} \\ b_i \\ b_{i+1} \\ \vdots \\ b_p \end{array} \begin{array}{c} \begin{array}{cccccccc} b_1^\star & b_2^\star & \ldots & b_{i-1}^\star & b_i^\star & b_{i+1}^\star & \ldots & b_p^\star \end{array} \\ \begin{pmatrix} 1 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 \\ m_{2,1} & 1 & \ldots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ m_{i-1,1} & m_{i-1,2} & \ldots & 1 & 0 & 0 & 0 & 0 \\ m_{i,1} & m_{i,2} & \ldots & m_{i,i-1} & 1 & 0 & 0 & 0 \\ m_{i+1,1} & m_{i+1,2} & \ldots & m_{i+1,i-1} & m_{i+1,i} & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{p,1} & m_{p,2} & \ldots & m_{p,i-1} & m_{p,i} & m_{p,i+1} & \ldots & 1 \end{pmatrix} \end{array}$$

LLL algorithm =
Gauss' reduction steps on local bases

$$U_i := \begin{array}{c} \\ u_i \\ v_i \end{array} \begin{array}{c} \begin{array}{cc} b_i^\star & b_{i+1}^\star \end{array} \\ \begin{pmatrix} 1 & 0 \\ m_{i+1,i} & 1 \end{pmatrix} \end{array}$$

The LLL algorithm

Input : A lattice $\mathcal{L}$ given by a basis $B = (b_1, b_2, \ldots, b_n)$

The algorithm deals with the Gram–Schmidt orthogonalized system $B^\star$ and the matrix $\mathcal{P} := (m_{i,j})$ which expresses B as a function of $B^\star$.

$$\mathcal{P} := \begin{array}{c} \\ b_1 \\ b_2 \\ \vdots \\ b_{i-1} \\ b_i \\ b_{i+1} \\ \vdots \\ b_p \end{array} \begin{array}{c} \begin{array}{cccccccc} b_1^\star & b_2^\star & \ldots & b_{i-1}^\star & b_i^\star & b_{i+1}^\star & \ldots & b_p^\star \end{array} \\ \begin{pmatrix} 1 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 \\ m_{2,1} & 1 & \ldots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ m_{i-1,1} & m_{i-1,2} & \ldots & 1 & 0 & 0 & 0 & 0 \\ m_{i,1} & m_{i,2} & \ldots & m_{i,i-1} & 1 & 0 & 0 & 0 \\ m_{i+1,1} & m_{i+1,2} & \ldots & m_{i+1,i-1} & m_{i+1,i} & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{p,1} & m_{p,2} & \ldots & m_{p,i-1} & m_{p,i} & m_{p,i+1} & \ldots & 1 \end{pmatrix} \end{array}$$

LLL algorithm =
Gauss' reduction steps on local bases

$$U_i := \begin{array}{c} \\ u_i \\ v_i \end{array} \begin{array}{c} \begin{array}{cc} b_i^\star & b_{i+1}^\star \end{array} \\ \begin{pmatrix} 1 & 0 \\ m_{i+1,i} & 1 \end{pmatrix} \end{array}$$

Main principles for the LLL Algorithm

The LLL algorithm performs the `A-Gauss` algorithm on local bases $U_i = (u_i, v_i)$ associated with a complex z_i with three differences

(a) The output test is weaker: with a fixed $\tau \leq 1$

the test $|v_i| > |u_i|$ is replaced by the test $|v_i| > \tau |u_i|$.

Then the output domain for z_i is the pseudo fondamental domain

$$\mathcal{F}_\tau := \{z \mid 0 \leq \Re z \leq 1/2, \quad |z| \geq \tau\},$$

(b) The operations are decided on the system (u_i, v_i)

– then reflected on the system (b_i, b_{i+1}).

(c) The algorithm is performed on the local basis U_i step by step.

More precisely, the algorithm computes a basis $\widehat{B}$ that

(i) is size–reduced: $|\widehat{m}_{i,j}| \leq 1/2$

(ii) for which the complex z_i fulfill the Lovász conditions $\mathcal{L}_\tau(i)$: $|\widehat{z}_i| \geq \tau$

Main principles for the LLL Algorithm

The LLL algorithm performs the `A-Gauss` algorithm on local bases $U_i = (u_i, v_i)$ associated with a complex z_i with three differences

(a) The output test is weaker: with a fixed $\tau \le 1$

the test $|v_i| > |u_i|$ is replaced by the test $|v_i| > \tau |u_i|$.

Then the output domain for z_i is the pseudo fondamental domain

$$\mathcal{F}_\tau := \{z \mid 0 \le \Re z \le 1/2, \quad |z| \ge \tau\},$$

(b) The operations are decided on the system (u_i, v_i)

– then reflected on the system (b_i, b_{i+1}).

(c) The algorithm is performed on the local basis U_i step by step.

More precisely, the algorithm computes a basis $\widehat{B}$ that

(i) is size–reduced: $|\widehat{m}_{i,j}| \le 1/2$

(ii) for which the complex z_i fulfill the Lovász conditions $\mathcal{L}_\tau(i)$: $|\widehat{z}_i| \ge \tau$

Main principles for the LLL Algorithm

The LLL algorithm performs the `A-Gauss` algorithm on local bases $U_i = (u_i, v_i)$ associated with a complex z_i with three differences

(a) The output test is weaker: with a fixed $\tau \le 1$

the test $|v_i| > |u_i|$ is replaced by the test $|v_i| > \tau |u_i|$.

Then the output domain for z_i is the pseudo fondamental domain

$$\mathcal{F}_\tau := \{z \mid 0 \le \Re z \le 1/2, \quad |z| \ge \tau\},$$

(b) The operations are decided on the system (u_i, v_i)

– then reflected on the system (b_i, b_{i+1}).

(c) The algorithm is performed on the local basis U_i step by step.

More precisely, the algorithm computes a basis $\widehat{B}$ that

(i) is size–reduced: $|\widehat{m}_{i,j}| \le 1/2$

(ii) for which the complex z_i fulfill the Lovász conditions $\mathcal{L}_\tau(i)$: $|\widehat{z}_i| \ge \tau$

Main principles for the LLL Algorithm

The LLL algorithm performs the `A-Gauss` algorithm on local bases $U_i = (u_i, v_i)$ associated with a complex z_i with three differences

(a) The output test is weaker: with a fixed $\tau \le 1$

the test $|v_i| > |u_i|$ is replaced by the test $|v_i| > \tau |u_i|$.

Then the output domain for z_i is the pseudo fondamental domain

$$\mathcal{F}_\tau := \{ z \mid 0 \le \Re z \le 1/2, \quad |z| \ge \tau \} ,$$

(b) The operations are decided on the system (u_i, v_i)

– then reflected on the system (b_i, b_{i+1}).

(c) The algorithm is performed on the local basis U_i step by step.

More precisely, the algorithm computes a basis $\widehat{B}$ that

(i) is size–reduced: $|\widehat{m}_{i,j}| \le 1/2$

(ii) for which the complex z_i fulfill the Lovász conditions $\mathcal{L}_\tau(i)$: $|\widehat{z}_i| \ge \tau$

Main principles for the LLL Algorithm

The LLL algorithm performs the `A-Gauss` algorithm on local bases $U_i = (u_i, v_i)$ associated with a complex z_i with three differences

(a) The output test is weaker: with a fixed $\tau \leq 1$

the test $|v_i| > |u_i|$ is replaced by the test $|v_i| > \tau |u_i|$.

Then the output domain for z_i is the pseudo fondamental domain

$$\mathcal{F}_\tau := \{z \mid 0 \leq \Re z \leq 1/2, \quad |z| \geq \tau\},$$

(b) The operations are decided on the system (u_i, v_i)

– then reflected on the system (b_i, b_{i+1}).

(c) The algorithm is performed on the local basis U_i step by step.

More precisely, the algorithm computes a basis $\widehat{B}$ that

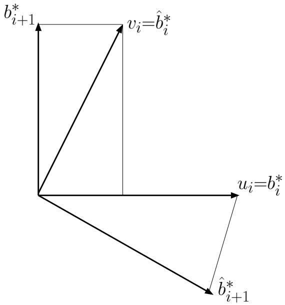
(i) is size–reduced: $|\widehat{m}_{i,j}| \leq 1/2$

(ii) for which the complex z_i fulfill the Lovász conditions $\mathcal{L}_\tau(i)$: $|\widehat{z}_i| \geq \tau$

Two main types of operations performed on the system U_i

(i) Translation $b_{i+1} := b_{i+1} - \lfloor m_{i+1,i} \rceil b_i$. also called size-reduction
This does not change ℓ_{i+1}, and entails the inequality $|m_{i+1,i}| \le (1/2)$.

(ii) Exchange between b_i and b_{i+1} when $\mathcal{L}_\tau(i)$ is not satisfied



$$\rho^2 := |z_i|^2 = \frac{|v_i|^2}{|u_i|^2} = \frac{\ell_{i+1}^2}{\ell_i^2} + m_{i+1,i}^2 \le \tau^2 < 1$$

The exchange modifies the lengths ℓ_i, ℓ_{i+1}.
The new values $\check{\ell}_i, \check{\ell}_{i+1}, \check{y}_i$ satisfy

$$\check{\ell}_i = \rho\, \ell_i \quad \check{\ell}_{i+1} = \left(\frac{1}{\rho}\right) \ell_{i+1}$$

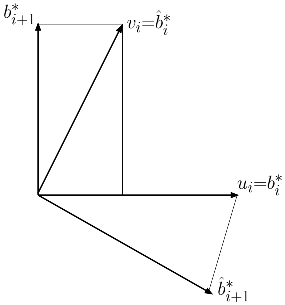
During the exchange: ℓ_i is decreasing, both ℓ_{i+1} and y_i are increasing.

Two main types of operations performed on the system U_i

(i) Translation $b_{i+1} := b_{i+1} - \lfloor m_{i+1,i} \rceil b_i$. also called size-reduction

This does not change ℓ_{i+1}, and entails the inequality $|m_{i+1,i}| \le (1/2)$.

(ii) Exchange between b_i and b_{i+1} when $\mathcal{L}_\tau(i)$ is not satisfied



$$\rho^2 := |z_i|^2 = \frac{|v_i|^2}{|u_i|^2} = \frac{\ell^2_{i+1}}{\ell^2_i} + m^2_{i+1,i} \le \tau^2 < 1$$

The exchange modifies the lengths ℓ_i, ℓ_{i+1}.

The new values $\check{\ell}_i, \check{\ell}_{i+1}, \check{y}_i$ satisfy

$$\check{\ell}_i = \rho\, \ell_i \quad \check{\ell}_{i+1} = \left(\frac{1}{\rho}\right) \ell_{i+1}$$

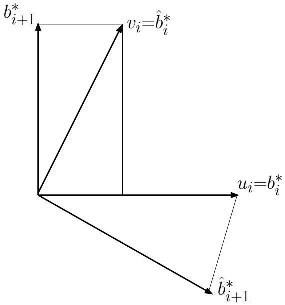
During the exchange: ℓ_i is decreasing, both ℓ_{i+1} and y_i are increasing.

Two main types of operations performed on the system U_i

(i) Translation $b_{i+1} := b_{i+1} - \lfloor m_{i+1,i} \rceil b_i$. also called size-reduction

This does not change ℓ_{i+1}, and entails the inequality $|m_{i+1,i}| \le (1/2)$.

(ii) Exchange between b_i and b_{i+1} when $\mathcal{L}_\tau(i)$ is not satisfied



$$\rho^2 := |z_i|^2 = \frac{|v_i|^2}{|u_i|^2} = \frac{\ell^2_{i+1}}{\ell^2_i} + m^2_{i+1,i} \le \tau^2 < 1$$

The exchange modifies the lengths ℓ_i, ℓ_{i+1}.

The new values $\check{\ell}_i, \check{\ell}_{i+1}, \check{y}_i$ satisfy

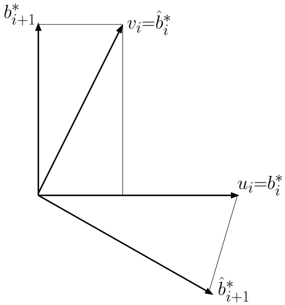
$$\check{\ell}_i = \rho\, \ell_i \quad \check{\ell}_{i+1} = \left(\frac{1}{\rho}\right) \ell_{i+1}$$

During the exchange: ℓ_i is decreasing, both ℓ_{i+1} and y_i are increasing.

Two main types of operations performed on the system U_i

(i) Translation $b_{i+1} := b_{i+1} - \lfloor m_{i+1,i} \rceil b_i$. also called size-reduction
This does not change ℓ_{i+1}, and entails the inequality $|m_{i+1,i}| \le (1/2)$.

(ii) Exchange between b_i and b_{i+1} when $\mathcal{L}_\tau(i)$ is not satisfied



$$\rho^2 := |z_i|^2 = \frac{|v_i|^2}{|u_i|^2} = \frac{\ell^2_{i+1}}{\ell^2_i} + m^2_{i+1,i} \le \tau^2 < 1$$

The exchange modifies the lengths ℓ_i, ℓ_{i+1}.
The new values $\check{\ell}_i, \check{\ell}_{i+1}, \check{y}_i$ satisfy

$$\check{\ell}_i = \rho\, \ell_i \quad \check{\ell}_{i+1} = \left(\frac{1}{\rho}\right) \ell_{i+1}$$

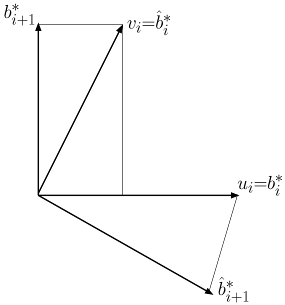
During the[1] exchange: ℓ_i is decreasing, both ℓ_{i+1} and y_i are increasing.

Two main types of operations performed on the system U_i

(i) Translation $b_{i+1} := b_{i+1} - \lfloor m_{i+1,i} \rceil b_i$. also called size-reduction

This does not change ℓ_{i+1}, and entails the inequality $|m_{i+1,i}| \le (1/2)$.

(ii) Exchange between b_i and b_{i+1} when $\mathcal{L}_\tau(i)$ is not satisfied



$$\rho^2 := |z_i|^2 = \frac{|v_i|^2}{|u_i|^2} = \frac{\ell^2_{i+1}}{\ell^2_i} + m^2_{i+1,i} \le \tau^2 < 1$$

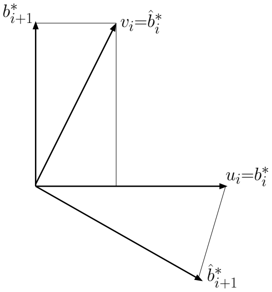
The exchange modifies the lengths ℓ_i, ℓ_{i+1}.

The new values $\check{\ell}_i, \check{\ell}_{i+1}, \check{y}_i$ satisfy

$$\check{\ell}_i = \rho\, \ell_i \quad \check{\ell}_{i+1} = \left(\frac{1}{\rho}\right) \ell_{i+1}$$

During the exchange: ℓ_i is decreasing, both ℓ_{i+1} and y_i are increasing.

Description of the LLL Algorithm.



LLL(τ) Algorithm $\quad (\tau \le 1)$

Input. A basis $\boldsymbol{b} = (b_1, \ldots, b_n)$ of a lattice $\mathcal{L}$.

Output. A LLL(τ)-reduced basis $\widehat{\boldsymbol{b}}$ of $\mathcal{L}$

Compute the vector $\boldsymbol{b}^\star$ and the matrix P;

Size reduce $\boldsymbol{b}$;

While the set $\mathcal{J}_\tau(\boldsymbol{b})$ is not empty, **do**

- Choose an index $i \in \mathcal{J}_\tau(\boldsymbol{b})$;
- Exchange b_i and b_{i+1};
- Update $\boldsymbol{b}^\star$ and P;
- Size-reduce $\boldsymbol{b}$

$$\mathcal{J}_\tau(\boldsymbol{b}) := \{i \in [1..d-1];\ \mathcal{L}_\tau(i) \text{ is not fulfilled}\} = \{i \in [1..d-1];\ x_i^2 + y_i^2 < \tau^2\},$$

Various strategies for the choice of the index $i \in \mathcal{J}_\tau(\boldsymbol{b})$

Description of the LLL Algorithm.

Important role of the potential $P(\boldsymbol{b})$

$$P(\boldsymbol{b}) = \prod_{i=1}^{n-1} \det \mathcal{L}(b_1, b_2, \ldots, b_i) = \prod_{i=1}^{n} \ell_i^{n-i}$$

At each step of the while,
$P(\boldsymbol{b})$ is decreased with the factor $\rho < 1$.

Number of iterations.

$$K(\boldsymbol{b}) \le \frac{1}{|\log \rho_0|} \log \frac{P(\boldsymbol{b})}{P(\widehat{\boldsymbol{b}})},$$

where ρ_0 is the maximal value of ρ.

LLL(τ) Algorithm $\quad (\tau \le 1)$

Input. A basis $\boldsymbol{b} = (b_1, \ldots, b_n)$ of a lattice $\mathcal{L}$.
Output. A LLL(τ)-reduced basis $\widehat{\boldsymbol{b}}$ of $\mathcal{L}$

Compute the vector $\boldsymbol{b}^\star$ and the matrix P;
Size reduce $\boldsymbol{b}$;
While the set $\mathcal{J}_\tau(\boldsymbol{b})$ is not empty, **do**

- Choose an index $i \in \mathcal{J}_\tau(\boldsymbol{b})$;
- Exchange b_i and b_{i+1};
- Update $\boldsymbol{b}^\star$ and P;
- Size-reduce $\boldsymbol{b}$

$$\mathcal{J}_\tau(\boldsymbol{b}) := \{i \in [1..d-1];\ \mathcal{L}_\tau(i) \text{ is not fulfilled}\} = \{i \in [1..d-1];\ x_i^2 + y_i^2 < \tau^2\},$$

Description of the LLL Algorithm.

Important role of the potential $P(\boldsymbol{b})$

$$P(\boldsymbol{b}) = \prod_{i=1}^{n-1} \det \mathcal{L}(b_1, b_2, \ldots, b_i) = \prod_{i=1}^{n} \ell_i^{n-i}$$

At each step of the while,
$P(\boldsymbol{b})$ is decreased with the factor $\rho < 1$.

Number of iterations.

$$K(\boldsymbol{b}) \le \frac{1}{|\log \rho_0|} \log \frac{P(\boldsymbol{b})}{P(\widehat{\boldsymbol{b}})},$$

where ρ_0 is the maximal value of ρ.

LLL(τ) Algorithm $\quad (\tau \le 1)$

Input. A basis $\boldsymbol{b} = (b_1, \ldots, b_n)$ of a lattice $\mathcal{L}$.
Output. A LLL(τ)-reduced basis $\widehat{\boldsymbol{b}}$ of $\mathcal{L}$

Compute the vector $\boldsymbol{b}^\star$ and the matrix P;
Size reduce $\boldsymbol{b}$;
While the set $\mathcal{J}_\tau(\boldsymbol{b})$ is not empty, **do**
    Choose an index $i \in \mathcal{J}_\tau(\boldsymbol{b})$;
    Exchange b_i and b_{i+1};
    Update $\boldsymbol{b}^\star$ and P;
    Size-reduce $\boldsymbol{b}$

$$\mathcal{J}_\tau(\boldsymbol{b}) := \{i \in [1..d-1];\ \mathcal{L}_\tau(i) \text{ is not fulfilled}\} = \{i \in [1..d-1];\ x_i^2 + y_i^2 < \tau^2\},$$

Siegel conditions.

We consider two parameters $\tau \leq 1$ and $\sigma \leq \sqrt{3}/2$
with the relation $\tau^2 = \sigma^2 + 1/4$.

We replace the Lovász condition $\mathcal{L}_\tau(i) : |z_i| \geq \tau$
by the weaker Siegel condition $\mathcal{S}_\sigma(i) : |y_i| \geq \sigma$

The Siegel condition is indeed weaker...
(due to the size reduction $|x_i| \leq 1/2$)

$$|z_i|^2 \geq \tau^2 \Longrightarrow y_i^2 = |z_i|^2 - x_i^2 \geq \tau^2 - (1/4) = \sigma^2$$

One has another version of the LLL algorithm,
the Siegel algorithm, denoted as the $\Sigma(\sigma)$ Algorithm.

Siegel conditions.

We consider two parameters $\tau \leq 1$ and $\sigma \leq \sqrt{3}/2$
with the relation $\tau^2 = \sigma^2 + 1/4$.

We replace the Lovász condition $\mathcal{L}_\tau(i)$: $|z_i| \geq \tau$
by the weaker Siegel condition $\mathcal{S}_\sigma(i)$: $|y_i| \geq \sigma$

The Siegel condition is indeed weaker...
(due to the size reduction $|x_i| \leq 1/2$)

$$|z_i|^2 \geq \tau^2 \Longrightarrow y_i^2 = |z_i|^2 - x_i^2 \geq \tau^2 - (1/4) = \sigma^2$$

One has another version of the LLL algorithm,
the Siegel algorithm, denoted as the $\Sigma(\sigma)$ Algorithm.

Siegel conditions.

We consider two parameters $\tau \leq 1$ and $\sigma \leq \sqrt{3}/2$
with the relation $\tau^2 = \sigma^2 + 1/4$.

We replace the Lovász condition $\mathcal{L}_\tau(i)$: $|z_i| \geq \tau$
by the weaker Siegel condition $\mathcal{S}_\sigma(i)$: $|y_i| \geq \sigma$

The Siegel condition is indeed weaker...
(due to the size reduction $|x_i| \leq 1/2$)

$$|z_i|^2 \geq \tau^2 \Longrightarrow y_i^2 = |z_i|^2 - x_i^2 \geq \tau^2 - (1/4) = \sigma^2$$

One has another version of the LLL algorithm,
the Siegel algorithm, denoted as the $\Sigma(\sigma)$ Algorithm.

Siegel conditions.

We consider two parameters $\tau \leq 1$ and $\sigma \leq \sqrt{3}/2$

with the relation $\tau^2 = \sigma^2 + 1/4$.

We replace the Lovász condition $\mathcal{L}_\tau(i)$: $|z_i| \geq \tau$

by the weaker Siegel condition $\mathcal{S}_\sigma(i)$: $|y_i| \geq \sigma$

The Siegel condition is indeed weaker...

(due to the size reduction $|x_i| \leq 1/2$)

$$|z_i|^2 \geq \tau^2 \Longrightarrow y_i^2 = |z_i|^2 - x_i^2 \geq \tau^2 - (1/4) = \sigma^2$$

One has another version of the LLL algorithm,

the Siegel algorithm, denoted as the $\Sigma(\sigma)$ Algorithm.

Siegel version –An additive point of view.

$\Sigma(\sigma)$ Algorithm $\quad (\sigma \le \sqrt{3}/2)$

Input. A basis $\boldsymbol{b} = (b_1, \ldots, b_n)$ of a lattice $\mathcal{L}$.
Output. A Siegel(σ)-reduced basis $\widehat{\boldsymbol{b}}$ of $\mathcal{L}$

Compute the vector $\boldsymbol{b}^\star$ and the matrix P;
Size reduce $\boldsymbol{b}$;
While the set $\mathcal{K}_\sigma(\boldsymbol{b})$ is not empty, **do**

  Choose an index $i \in \mathcal{K}_\sigma(\boldsymbol{b})$;
  Exchange b_i and b_{i+1};
  Update $\boldsymbol{b}^\star$ and P;
  Size-reduce $\boldsymbol{b}$

With an additive point of view

$$t_i = -\log_\sigma y_i, \quad \alpha = -\log_\sigma \rho,$$

and only viewed on the vector $\boldsymbol{t}$,
the $\Sigma(\sigma)$ algorithm is written as:

While $\exists t_i > 1$, **do**

$$\check{t}_i := t_i - 2\alpha;$$
$$\check{t}_{i-1} := t_{i-1} + \alpha;$$
$$\check{t}_{i+1} := t_{i+1} + \alpha;$$

$$\mathcal{K}_\sigma(\boldsymbol{b}) := \{i \in [1..d-1];\ \mathcal{S}_\sigma(i) \text{ is not fulfilled}\} = \{i \in [1..d-1];\ y_i < \sigma\},$$

An additive point of view on the LLL algorithm.

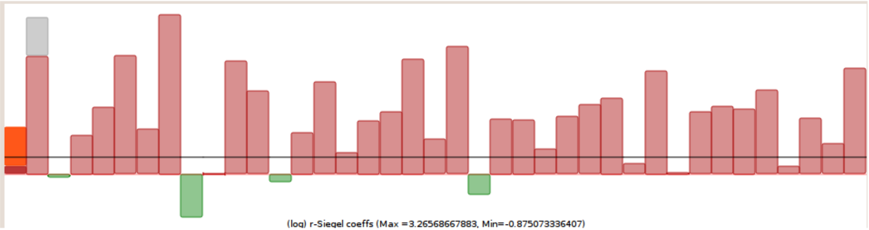
$$q_i := \log_\sigma \ell_i, \qquad c_i := -\log_\sigma y_i = q_i - q_{i+1}, \qquad \alpha := -\log_\sigma \rho,$$

The Siegel condition becomes $q_i \le q_{i+1} + 1$ or $c_i \le 1$,

The exchange in the LLL algorithm becomes

If $q_i > q_{i+1} + 1$, then $[\check{q_i} = q_i - \alpha, \quad \check{q_{i+1}} = q_{i+1} + \alpha]$.

If $c_i > 1$, then $[\check{c_i} = c_i - 2\alpha, \quad \check{c_{i+1}} = c_{i+1} + \alpha, \quad \check{c_{i-1}} = c_{i-1} + \alpha,]$.



An additive point of view on the LLL algorithm.

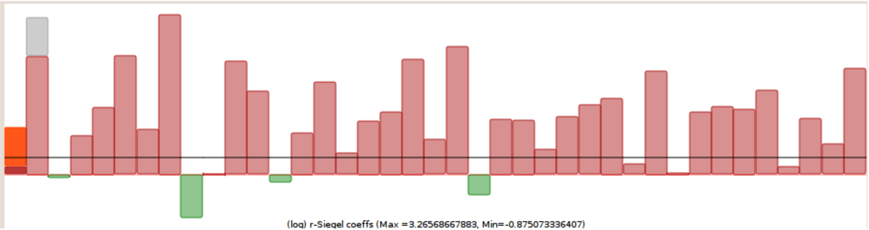
$$q_i := \log_\sigma \ell_i, \qquad c_i := -\log_\sigma y_i = q_i - q_{i+1}, \qquad \alpha := -\log_\sigma \rho,$$

The Siegel condition becomes $\quad q_i \le q_{i+1} + 1 \quad$ or $\quad c_i \le 1,$

The exchange in the LLL algorithm becomes

If $\quad q_i > q_{i+1} + 1, \quad$ then $\quad [\check{q}_i = q_i - \alpha, \quad \check{q}_{i+1} = q_{i+1} + \alpha].$

If $\quad c_i > 1, \quad$ then $\quad [\check{c}_i = c_i - 2\alpha, \quad \check{c}_{i+1} = c_{i+1} + \alpha, \quad \check{c}_{i-1} = c_{i-1} + \alpha,].$



An additive point of view on the LLL algorithm.

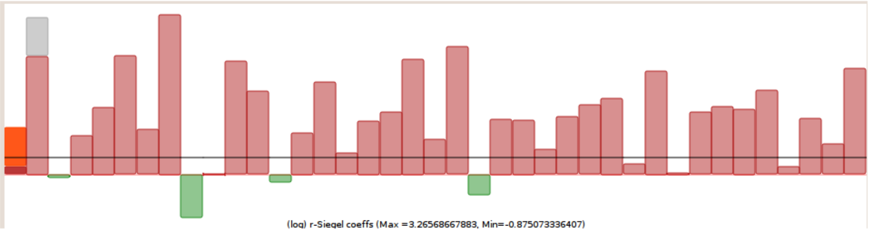
$$q_i := \log_\sigma \ell_i, \qquad c_i := -\log_\sigma y_i = q_i - q_{i+1}, \qquad \alpha := -\log_\sigma \rho,$$

The Siegel condition becomes $q_i \leq q_{i+1} + 1$ or $c_i \leq 1,$

The exchange in the LLL algorithm becomes

If $q_i > q_{i+1} + 1,$ then $[\check{q}_i = q_i - \alpha, \quad \check{q_{i+1}} = q_{i+1} + \alpha].$

If $c_i > 1,$ then $[\check{c}_i = c_i - 2\alpha, \quad \check{c_{i+1}} = c_{i+1} + \alpha, \quad \check{c_{i-1}} = c_{i-1} + \alpha,].$



An additive point of view on the LLL algorithm.

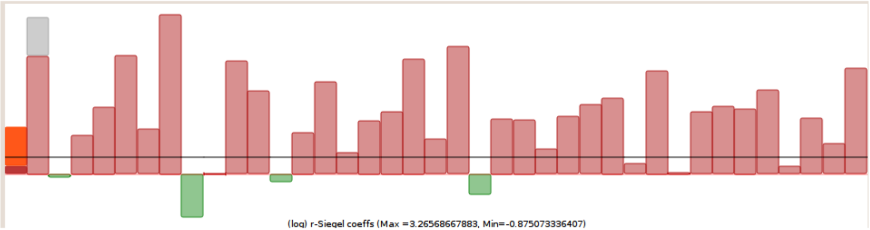
$$q_i := \log_\sigma \ell_i, \qquad c_i := -\log_\sigma y_i = q_i - q_{i+1}, \qquad \alpha := -\log_\sigma \rho,$$

The Siegel condition becomes $q_i \le q_{i+1} + 1$ or $c_i \le 1$,

The exchange in the LLL algorithm becomes

If $q_i > q_{i+1} + 1$, then $[\check{q_i} = q_i - \alpha, \quad \check{q_{i+1}} = q_{i+1} + \alpha]$.

If $c_i > 1$, then $[\check{c_i} = c_i - 2\alpha, \quad \check{c_{i+1}} = c_{i+1} + \alpha, \quad \check{c_{i-1}} = c_{i-1} + \alpha,]$.



An additive point of view on the LLL algorithm.

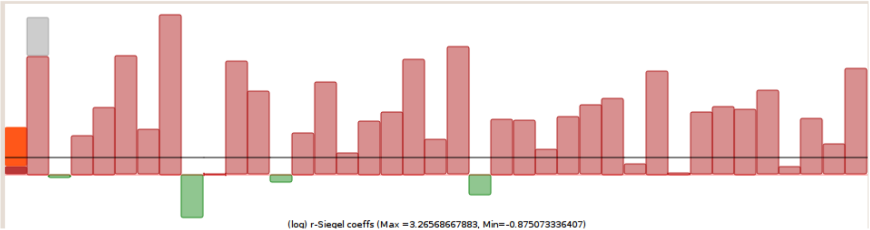
$$q_i := \log_\sigma \ell_i, \qquad c_i := -\log_\sigma y_i = q_i - q_{i+1}, \qquad \alpha := -\log_\sigma \rho,$$

The Siegel condition becomes $q_i \leq q_{i+1} + 1$ or $c_i \leq 1,$

The exchange in the LLL algorithm becomes

If $q_i > q_{i+1} + 1,$ then $[\check{q_i} = q_i - \alpha, \quad \check{q_{i+1}} = q_{i+1} + \alpha].$

If $c_i > 1,$ then $[\check{c_i} = c_i - 2\alpha, \quad \check{c_{i+1}} = c_{i+1} + \alpha, \quad \check{c_{i-1}} = c_{i-1} + \alpha,].$



Examples of problems that are solved with the LLL Algorithm.

Algorithme Construction de base

Donnée : un système générateur $b=(b_1, b_2, \ldots, b_n)$ d'un réseau L de $\mathbf{R}^p$.

Résultat : une base b de L.

Gram;

$q := 0$.

Pour $i \in I$ *répéter*

$q := q+1$.

Pour j allant de $i-1$ à q *faire*

1. *Tant que* $l_j \neq 0$ faire

   Translater v_j parallèlement à u_j et donc b_{j+1} parallèlement à b_j.

   Échanger b_j et b_{j+1}.

   Recalculer par la procédure *Nouvortho* le triplet (b^*, m, l).

2. *Translater* alors b_{j+1} parallèlement aux b_k pour $k<j$ au moyen de *Propre* $(j+1)$.

3.10. La recherche d'une relation linéaire courte entre n vecteurs de $\mathbf{Z}^p$

Soit $y=(y_1, y_2, \ldots, y_n)$ le système formé par ces vecteurs, Y la matrice dont les colonnes sont les y_i. Soit $x=(x_1, x_2, \ldots, x_p)$ le système formé par les lignes de la matrice Y et L le réseau de $\mathbf{Z}^n$ engendré par x qu'on suppose de rang $q\,(q \leqq p)$. On veut construire un vecteur court de ce qu'on appelle *le réseau des relations* c'est-à-dire le réseau R des vecteurs $v=(v_1, v_2, \ldots, v_n)$ de $\mathbf{Z}^n$ vérifiant

$$\sum_{i=1}^{n} v_i y_i = 0 \quad \text{et donc } (v \mid x_i) = 0 \quad \text{pour tout } i, \qquad 1 \leqq i \leqq p.$$

On procède de la manière suivante :

1. On construit une base $b=(b_1, b_2, \ldots, b_n)$ du réseau $\mathbf{Z}^n$ tel que les q premiers vecteurs de b engendrent le même $\mathbf{Q}$-sous-espace vectoriel H que x.

2. Les derniers $n-q$ vecteurs de la base $c=(c_1, c_2, \ldots, c_n)$ duale de la base b sont alors une base du réseau des relations.

3. Il reste alors à chercher un vecteur court de ce réseau.

Il est clair que **LLL** résoud l'étape 3. Il est vrai aussi qu'un algorithme assez semblable à celui du paragraphe précédent permet de résoudre la première étape :

Partant de la base canonique b de $\mathbf{Z}^n$, nous définissons

1. le système b^* formé par les vecteurs b_i^*, projections des vecteurs b_i orthogonalement aux sous-espaces $K_{i-1} = H + H_{i-1}$;

2. le couple (m, l) et la partie I correspondant dont le cardinal est q, dimension de H.

Travaillant alors sur le triplet (b^*, m, l), nous cherchons par une succession d'échanges et de translations à faire décroître les indices $i \in I$ jusqu'à ce que $I = \{1, 2, \ldots, q\}$: nous avons ainsi obtenu la base b cherchée.

Remarquons que la première phase de cet algorithme permet aussi de calculer une base normale d'Hermite d'un réseau entier, c'est-à-dire une base b vérifiant la propriété suivante : pour tout i, b_i est un vecteur de l'hyperplan engendré par les i premiers vecteurs de la base canonique.

Cette même première phase permet aussi de compléter un vecteur primitif en une matrice unimodulaire.

Le second procède de manière récursive en utilisant un argument géométrique assez simple : la longueur $|b_n^*|$ mesure la distance entre deux hyperplans consécutifs du réseau parallèles à H_{n-1}. Or, puisque b est s-réduite au sens de Siegel, ces hyperplans sont assez « espacés » et on a, d'après le paragraphe 2.8, pour les valeurs usuelles de s et t :

$$|b_n^*|^2 \geqq \frac{1}{2^{n-1}} |b_n|^2 \quad \text{et donc} \quad |b_n^*|^2 \geqq \frac{1}{2^{n-1}} |\Lambda_1(L)|.$$

Par conséquent, $\lambda_1(L)$ ne peut se trouver que dans un petit nombre d'hyperplans de direction parallèle à H_{n-1} (ce « petit » nombre est de l'ordre de $2^{(n+1)/2}$) : on projette successivement dans ce nombre fini d'hyperplans affines, et dans chacun d'eux on peut utiliser le même genre d'arguments car l'inégalité précédente est vraie quand on remplace n par $n-1$ et $\lambda_1(L)$ par ses projetés dans ces hyperplans.

On obtient ainsi un algorithme qui considère $2^{n(n+1)/4}$ vecteurs du réseau.

Soit $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ un n-uplet de nombres réels. On cherche n nombres entiers $(p_1, p_2, \ldots, p_n)$ et un nombre entier q tels que les n nombres rationnels $(p_1/q, p_2/q, \ldots, p_n/q)$ soient de bonnes approximations des nombres donnés.

On connaît une réponse à cette question, due à Dirichlet, fondée sur le théorème de Minkowski, et donc non constructive :

Pour tout n, pour tout n-uplet $(\alpha_1, \alpha_2, \ldots, \alpha_n)$, pour tout couple (ε, Q) vérifiant $\varepsilon > 0$ et $Q \geqq \varepsilon^{-n}$, il existe des entiers $(p_1, p_2, \ldots, p_n)$ et un entier q vérifiant

$$0 < q \leqq Q \qquad \text{et} \qquad |q\alpha_i - p_i| < \varepsilon \quad \text{pour tout } i, \qquad 1 \leqq i \leqq n.$$

Lagarias [11] a pu donner une version approchée mais constructive à ce théorème en appliquant l'algorithme **LLL** au réseau L engendré par les lignes v_i de la matrice

$$A = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 0 \\ \alpha_1 & \alpha_2 & \alpha_3 & \ldots & \alpha_n & \varepsilon/Q \end{pmatrix}$$

Étant donné un polytope P de $\mathbf{R}^n$ de volume non nul, déterminer les points de coordonnées entières situés à l'intérieur de ce polytope.

On sait que ce problème est NP-dur en général. Mais, là encore, on peut chercher un algorithme qui soit polynomial quand la dimension n est fixée. C'est la démarche de Lenstra [16], bien décrite dans [17], qui utilise à la fois des arguments géométriques liés à la réduction des réseaux — l'espacement des hyperplans du réseau parallèles à H_{n-1} — et d'autres arguments liés à la méthode de l'ellipsoïde en programmation linéaire — la possibilité de coincer un polytope entre deux ellipsoïdes concentriques et homothétiques.

On commence par considérer que P est un ellipsoïde, puis on se ramènera à ce cas, en coinçant un polytope entre deux ellipsoïdes.

Soit f la transformation linéaire qui transforme P en une sphère unité S. Soit L le transformé du réseau $\mathbf{Z}^n$ par f. Le problème est alors transformé en le suivant :

Déterminer les points de L situés à l'intérieur de S.

On réduit le réseau L en lui appliquant l'algorithme **LLL** : on obtient ainsi une base $(b_1, b_2, \ldots, b_n)$. Puis, on procède de manière récursive, en utilisant

Simplified models for the LLL Algorithm.

An additive point of view on the LLL algorithm.

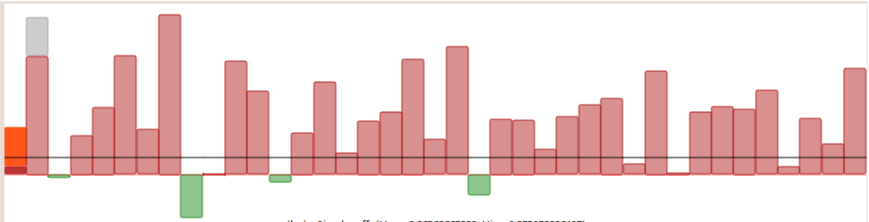
$$q_i := \log_\sigma \ell_i, \qquad c_i := -\log_\sigma y_i = q_i - q_{i+1}, \qquad \alpha := -\log_\sigma \rho,$$

The Siegel condition becomes $q_i \le q_{i+1} + 1$ or $c_i \le 1$,

The exchange in the LLL algorithm becomes

If $q_i > q_{i+1} + 1$, then $[\check{q_i} = q_i - \alpha, \quad \check{q_{i+1}} = q_{i+1} + \alpha]$.

If $c_i > 1$, then $[\check{c_i} = c_i - 2\alpha, \quad \check{c_{i+1}} = c_{i+1} + \alpha, \quad \check{c_{i-1}} = c_{i-1} + \alpha,]$.



An additive point of view on the LLL algorithm.

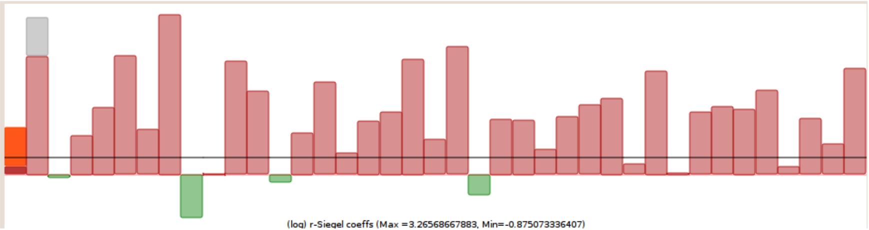
$$q_i := \log_\sigma \ell_i, \qquad c_i := -\log_\sigma y_i = q_i - q_{i+1}, \qquad \alpha := -\log_\sigma \rho,$$

The Siegel condition becomes $q_i \le q_{i+1} + 1$ or $c_i \le 1$,

The exchange in the LLL algorithm becomes

If $q_i > q_{i+1} + 1$, then $[\check{q}_i = q_i - \alpha, \quad \check{q}_{i+1} = q_{i+1} + \alpha]$.

If $c_i > 1$, then $[\check{c}_i = c_i - 2\alpha, \quad \check{c}_{i+1} = c_{i+1} + \alpha, \quad \check{c}_{i-1} = c_{i-1} + \alpha,]$.



An additive point of view on the LLL algorithm.

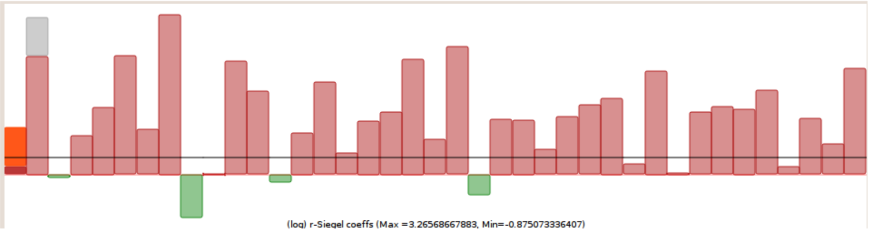
$$q_i := \log_\sigma \ell_i, \qquad c_i := -\log_\sigma y_i = q_i - q_{i+1}, \qquad \alpha := -\log_\sigma \rho,$$

The Siegel condition becomes $q_i \le q_{i+1} + 1$ or $c_i \le 1,$

The exchange in the LLL algorithm becomes

If $q_i > q_{i+1} + 1,$ then $[\check{q_i} = q_i - \alpha, \quad \check{q_{i+1}} = q_{i+1} + \alpha].$

If $c_i > 1,$ then $[\check{c_i} = c_i - 2\alpha, \quad \check{c_{i+1}} = c_{i+1} + \alpha, \quad \check{c_{i-1}} = c_{i-1} + \alpha,].$



An additive point of view on the LLL algorithm.

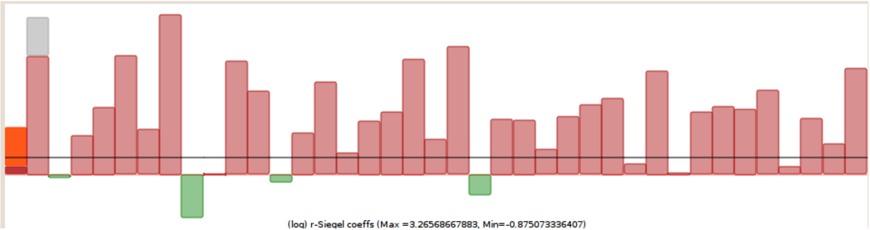
$$q_i := \log_\sigma \ell_i, \qquad c_i := -\log_\sigma y_i = q_i - q_{i+1}, \qquad \alpha := -\log_\sigma \rho,$$

The Siegel condition becomes $q_i \le q_{i+1} + 1$ or $c_i \le 1,$

The exchange in the LLL algorithm becomes

If $q_i > q_{i+1} + 1,$ then $[\check{q_i} = q_i - \alpha, \quad \check{q_{i+1}} = q_{i+1} + \alpha]$.

If $c_i > 1,$ then $[\check{c_i} = c_i - 2\alpha, \quad \check{c_{i+1}} = c_{i+1} + \alpha, \quad \check{c_{i-1}} = c_{i-1} + \alpha,]$.



An additive point of view on the LLL algorithm.

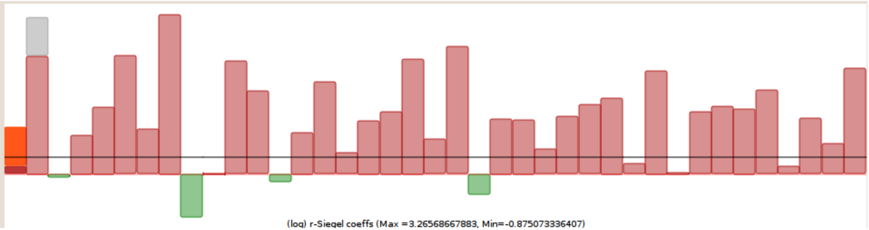
$$q_i := \log_\sigma \ell_i, \qquad c_i := -\log_\sigma y_i = q_i - q_{i+1}, \qquad \alpha := -\log_\sigma \rho,$$

The Siegel condition becomes $q_i \leq q_{i+1} + 1$ or $c_i \leq 1,$

The exchange in the LLL algorithm becomes

If $q_i > q_{i+1} + 1,$ then $[\check{q_i} = q_i - \alpha, \quad \check{q_{i+1}} = q_{i+1} + \alpha]$.

If $c_i > 1,$ then $[\check{c_i} = c_i - 2\alpha, \quad \check{c_{i+1}} = c_{i+1} + \alpha, \quad \check{c_{i-1}} = c_{i-1} + \alpha,]$.



Simplified models for the LLL algorithm (I).

In the dynamical system underlying the LLL algorithm,
the crucial parameter in the analysis of the LLL algorithm is the factor ρ

$$\rho = \frac{\ell_{i+1}^2}{\ell_i^2} + \{\{m_{i+1,i}\}\}^2, \qquad \{\{x\}\} := \text{centered fractional part of } x$$

Its analysis seems very difficult. We then introduce simplified models:

First model : the model `M1`.

The decreasing factor ρ (and its logarithm $\alpha := -\log_s \rho$) are constant.

Then, the equation

$$\text{If} \quad c_i > 1, \quad \text{then} \quad [\check{c_i} = c_i - 2\alpha, \quad \check{c_{i+1}} = c_{i+1} + \alpha, \quad \check{c_{i-1}} = c_{i-1} + \alpha,]$$

defines a chip firing game.

A very well studied dynamical system....

Simplified models for the LLL algorithm (I).

In the dynamical system underlying the LLL algorithm,
the crucial parameter in the analysis of the LLL algorithm is the factor ρ

$$\rho = \frac{\ell_{i+1}^2}{\ell_i^2} + \{\{m_{i+1,i}\}\}^2, \qquad \{\{x\}\} := \text{centered fractional part of } x$$

Its analysis seems very difficult. We then introduce simplified models:

First model : the model `M1`.

The decreasing factor ρ (and its logarithm $\alpha := -\log_s \rho$) are constant.

Then, the equation

$$\text{If} \quad c_i > 1, \quad \text{then} \quad [\check{c_i} = c_i - 2\alpha, \quad \check{c_{i+1}} = c_{i+1} + \alpha, \quad \check{c_{i-1}} = c_{i-1} + \alpha,]$$

defines a chip firing game.

A very well studied dynamical system....

Simplified models for the LLL algorithm (I).

In the dynamical system underlying the LLL algorithm,
the crucial parameter in the analysis of the LLL algorithm is the factor ρ

$$\rho = \frac{\ell_{i+1}^2}{\ell_i^2} + \{\{m_{i+1,i}\}\}^2, \qquad \{\{x\}\} := \text{centered fractional part of } x$$

Its analysis seems very difficult. We then introduce simplified models:

First model : the model `M1`.

The decreasing factor ρ (and its logarithm $\alpha := -\log_s \rho$) are constant.

Then, the equation

$$\text{If} \quad c_i > 1, \quad \text{then} \quad [\check{c_i} = c_i - 2\alpha, \quad \check{c_{i+1}} = c_{i+1} + \alpha, \quad \check{c_{i-1}} = c_{i-1} + \alpha,]$$

defines a chip firing game.

A very well studied dynamical system....

Simplified models for the LLL algorithm (II): the model M2

There are two terms in the decreasing factor $\rho = \frac{\ell_{i+1}^2}{\ell_i^2} + \{\{m_{i+1,i}\}\}^2$,

the ratio $y_i := \ell_{i+1}/\ell_i$ and the subdiagonal coefficient $x_i := \{\{m_{i+1,i}\}\}$.

In the model $\mathtt{M2}(\sigma)$, with $\sigma \le 3/4$

- the main variables are $y_i := \ell_{i+1}/\ell_i$,
- the coefficients $x_i := \{\{m_{i+1,i}\}\}$ play an auxiliary role
- they are chosen unif. at random in $[0, 1/2]$ and indep. of y_i's.
- the algorithm stops as soon as all the variables y_i satisfy $y_i \ge \sigma$.

  when it runs, there is an index i for which $x_i + y_i < 1$.

While there exists an index i for which $y_i < \sigma$,

choose such an index i and $x_i \in [0, 1/2]$,

$$y_{i-1} := y_{i-1}(y_i^2 + x_i^2)^{1/2}, \quad y_{i+1} := y_{i+1}(y_i^2 + x_i^2)^{1/2}, \quad y_i := \frac{y_i}{(y_i^2 + x_i^2)};$$

The model $\mathtt{M2}(\sigma)$ is a simplified model for the $\mathrm{LLL}(\sqrt{\sigma})$ algorithm.

Simplified models for the LLL algorithm (II): the model M2

There are two terms in the decreasing factor $\rho = \frac{\ell_{i+1}^2}{\ell_i^2} + \{\{m_{i+1,i}\}\}^2$,

the ratio $y_i := \ell_{i+1}/\ell_i$ and the subdiagonal coefficient $x_i := \{\{m_{i+1,i}\}\}$.

In the model $\mathtt{M2}(\sigma)$, with $\sigma \le 3/4$

- the main variables are $y_i := \ell_{i+1}/\ell_i$,
- the coefficients $x_i := \{\{m_{i+1,i}\}\}$ play an auxiliary role
- they are chosen unif. at random in $[0, 1/2]$ and indep. of y_i's.
- the algorithm stops as soon as all the variables y_i satisfy $y_i \ge \sigma$.

  when it runs, there is an index i for which $x_i + y_i < 1$.

While there exists an index i for which $y_i < \sigma$,

choose such an index i and $x_i \in [0, 1/2]$,

$$y_{i-1} := y_{i-1}(y_i^2 + x_i^2)^{1/2}, \quad y_{i+1} := y_{i+1}(y_i^2 + x_i^2)^{1/2}, \quad y_i := \frac{y_i}{(y_i^2 + x_i^2)};$$

The model $\mathtt{M2}(\sigma)$ is a simplified model for the LLL$(\sqrt{\sigma})$ algorithm.

Simplified models for the LLL algorithm (II): the model M2

There are two terms in the decreasing factor $\rho = \frac{\ell_{i+1}^2}{\ell_i^2} + \{\{m_{i+1,i}\}\}^2$,

the ratio $y_i := \ell_{i+1}/\ell_i$ and the subdiagonal coefficient $x_i := \{\{m_{i+1,i}\}\}$.

In the model $\mathtt{M2}(\sigma)$, with $\sigma \leq 3/4$

- the main variables are $y_i := \ell_{i+1}/\ell_i$,
- the coefficients $x_i := \{\{m_{i+1,i}\}\}$ play an auxiliary role
- they are chosen unif. at random in $[0, 1/2]$ and indep. of y_i's.
- the algorithm stops as soon as all the variables y_i satisfy $y_i \geq \sigma$.
  when it runs, there is an index i for which $x_i + y_i < 1$.

While there exists an index i for which $y_i < \sigma$,

choose such an index i and $x_i \in [0, 1/2]$,

$$y_{i-1} := y_{i-1}(y_i^2 + x_i^2)^{1/2}, \quad y_{i+1} := y_{i+1}(y_i^2 + x_i^2)^{1/2}, \quad y_i := \frac{y_i}{(y_i^2 + x_i^2)};$$

The model $\mathtt{M2}(\sigma)$ is a simplified model for the $\mathrm{LLL}(\sqrt{\sigma})$ algorithm.

Simplified models for the LLL algorithm (II): the model M2

There are two terms in the decreasing factor $\rho = \frac{\ell_{i+1}^2}{\ell_i^2} + \{\{m_{i+1,i}\}\}^2$,

the ratio $y_i := \ell_{i+1}/\ell_i$ and the subdiagonal coefficient $x_i := \{\{m_{i+1,i}\}\}$.

In the model $\mathtt{M2}(\sigma)$, with $\sigma \le 3/4$

- the main variables are $y_i := \ell_{i+1}/\ell_i$,
- the coefficients $x_i := \{\{m_{i+1,i}\}\}$ play an auxiliary role
- they are chosen unif. at random in $[0, 1/2]$ and indep. of y_i's.
- the algorithm stops as soon as all the variables y_i satisfy $y_i \ge \sigma$.
  when it runs, there is an index i for which $x_i + y_i < 1$.

While there exists an index i for which $y_i < \sigma$,

choose such an index i and $x_i \in [0, 1/2]$,

$$y_{i-1} := y_{i-1}(y_i^2 + x_i^2)^{1/2}, \quad y_{i+1} := y_{i+1}(y_i^2 + x_i^2)^{1/2}, \quad y_i := \frac{y_i}{(y_i^2 + x_i^2)};$$

The model $\mathtt{M2}(\sigma)$ is a simplified model for the $\mathrm{LLL}(\sqrt{\sigma})$ algorithm.

Simplified models for the LLL algorithm (II): the model M2

There are two terms in the decreasing factor $\rho = \frac{\ell_{i+1}^2}{\ell_i^2} + \{\{m_{i+1,i}\}\}^2$,

the ratio $y_i := \ell_{i+1}/\ell_i$ and the subdiagonal coefficient $x_i := \{\{m_{i+1,i}\}\}$.

In the model $\mathtt{M2}(\sigma)$, with $\sigma \le 3/4$

- the main variables are $y_i := \ell_{i+1}/\ell_i$,
- the coefficients $x_i := \{\{m_{i+1,i}\}\}$ play an auxiliary role
- they are chosen unif. at random in $[0, 1/2]$ and indep. of y_i's.
- the algorithm stops as soon as all the variables y_i satisfy $y_i \ge \sigma$.

  when it runs, there is an index i for which $x_i + y_i < 1$.

While there exists an index i for which $y_i < \sigma$,

choose such an index i and $x_i \in [0, 1/2]$,

$$y_{i-1} := y_{i-1}(y_i^2 + x_i^2)^{1/2}, \quad y_{i+1} := y_{i+1}(y_i^2 + x_i^2)^{1/2}, \quad y_i := \frac{y_i}{(y_i^2 + x_i^2)};$$

The model $\mathtt{M2}(\sigma)$ is a simplified model for the $\mathrm{LLL}(\sqrt{\sigma})$ algorithm.

Simplified models for the LLL algorithm (II): the model $\mathtt{M2}$

There are two terms in the decreasing factor $\rho = \dfrac{\ell_{i+1}^2}{\ell_i^2} + \{\{m_{i+1,i}\}\}^2$,

the ratio $y_i := \ell_{i+1}/\ell_i$ and the subdiagonal coefficient $x_i := \{\{m_{i+1,i}\}\}$.

In the model $\mathtt{M2}(\sigma)$, with $\sigma \le 3/4$

- the main variables are $y_i := \ell_{i+1}/\ell_i$,
- the coefficients $x_i := \{\{m_{i+1,i}\}\}$ play an auxiliary role
- they are chosen unif. at random in $[0, 1/2]$ and indep. of y_i's.
- the algorithm stops as soon as all the variables y_i satisfy $y_i \ge \sigma$.

  when it runs, there is an index i for which $x_i + y_i < 1$.

While there exists an index i for which $y_i < \sigma$,

choose such an index i and $x_i \in [0, 1/2]$,

$$y_{i-1} := y_{i-1}(y_i^2 + x_i^2)^{1/2}, \quad y_{i+1} := y_{i+1}(y_i^2 + x_i^2)^{1/2}, \quad y_i := \frac{y_i}{(y_i^2 + x_i^2)};$$

The model $\mathtt{M2}(\sigma)$ is a simplified model for the LLL$(\sqrt{\sigma})$ algorithm.