LLL lattice basis reduction algorithm

Helfer Etienne

21.03.2010

Contents

1 Lattice

1.1 Introduction

Since the LLL lattice reduction basis algorithm operates on a lattice it is important to understand what is it. Many concepts in Lattice theory are related with linear algebra : a lattice can be represented with the matrix of its basis, it has a determinant, and so on. Later we will need linear algebra methods and matrix properties for the LLL algorithm. I won't give a complete and precise view of the lattice theory but favor the geometrical point of view and focus on the elements that are needed to understand LLL basis reduction.

1.2 Definition

A lattice is a discrete subgroup of an Euclidean vector space. In general the vector space is $\mathbb{R}^n$ or a subspace of $\mathbb{R}^n$. It is conveniant to describe a lattice using its basis. The basis of a lattice is a set of linearly independent vectors in $\mathbb{R}^n$ which can generate the lattice by combining them. Notice that different bases can generate the same lattice (cf. figure 1).

Definition 1. *A set of vectors $\{\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_m}\}$ in $\mathbb{R}^n$ is linearly independent if the equation*

$$c_1\mathbf{b_1} + c_2\mathbf{b_2} + \cdots + c_m\mathbf{b_m} = \mathbf{0} \text{ where } c_i \in \mathbb{R} \tag{1}$$

accepts only the trivial solution $c_1 = c_2 = \cdots = c_m = 0$

Theorem 1. *If a set of vectors in $\mathbb{R}^n$ contains more vectors than n (if $m > n$), then this set of vectors is not linearly independent.*

Definition 2. *A subspace of $\mathbb{R}^n$ is a an arbitrary set H that has the following properties :*

1. *the nul vector $\mathbf{0}$ is an element of H*
2. *H is close under addition : for every $\mathbf{u}$ and $\mathbf{v}$ in H, their sum $\mathbf{u} + \mathbf{v}$ is an element of H*
3. *H is close under scalar multiplication : for every $\mathbf{u}$ in H and scalar c, the vector $c\mathbf{u}$ is an element of H*

Notice that $\mathbb{R}^n$ is a subspace of $\mathbb{R}^n$

Definition 3. *A basis B of a subspace H of $\mathbb{R}^n$ is a set of linearly independent vectors in $\mathbb{R}^n$ that generates H.*

$$B = \{\mathbf{b_1}, \mathbf{b_2}, ..., \mathbf{b_m}\} \text{ where } \mathbf{b_i} \in \mathbb{R}^n \tag{2}$$

$$H = \sum_{i=1}^{m} \mathbb{R}\mathbf{b_i} = \left\{ \sum_{i=1}^{m} c_i\mathbf{b_i} \text{ where } c_i \in \mathbb{R},\ \mathbf{b_i} \in \mathbb{R}^n \right\} \tag{3}$$

Definition 4. *A lattice Λ is a discrete subgroup of H generated by all the integer combinations of the vectors of some basis B :*

$$\Lambda = \sum_{i=1}^{m} \mathbb{Z}\mathbf{b_i} = \left\{ \sum_{i=1}^{m} z_i\mathbf{b_i} \text{ where } z_i \in \mathbb{Z},\ \mathbf{b_i} \in \mathbb{R}^n \right\} \tag{4}$$

Definition 5. *The rank m of a lattice Λ generated by a basis B of a subspace H of $\mathbb{R}^n$ is the number of vectors in B.*

Theorem 1 implies that $m \leq n$. If $m = n$ then H is $\mathbb{R}^n$ and Λ is a full rank lattice.

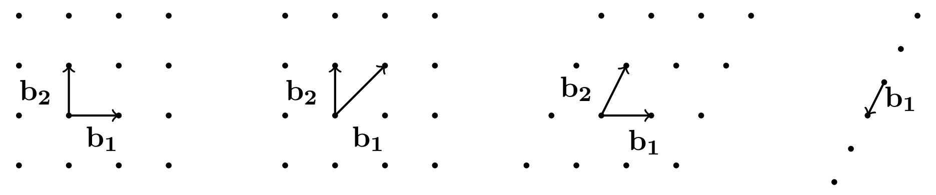


Figure 1: Examples of lattices generated with different bases in $\mathbb{R}^2$. The first and the second lattice are the same. The first three lattices are rank 2 and the fourth is rank 1.

1.3 Determinant

The determinant of a lattice Λ ($\det \Lambda$) is an important numerical invariant of Λ. Geometrically speaking, $\det \Lambda$ is the volume of the parallelepiped spanned by the basis. The determinant does not depend on the choice of the basis.

Another more general perspective is to consider $\det \Lambda$ as the inverse of the volume density of elements in Λ.

Definition 6. *The volume of a n-dimensional ball B of radius r is given by proposition*

$$\text{vol}B(r, n) = r^n \text{vol}B(1, n) = \frac{r^n \pi^{n/2}}{\frac{n}{2}!} \tag{5}$$

where $\frac{n}{2}!$ is inductively defined by $0! = 1$, $\frac{1}{2}! = \frac{\sqrt{\pi}}{2}$, and $\frac{n}{2}! = \frac{n}{2}\frac{n-2}{2}!$

Definition 7 (Determinant definition)**.** $\det \Lambda$ *is the volume of the m-dimensional ball B, where m is the rank of Λ, divided by the number of elements belonging to Λ in B when radius of B tends to infinity.*

$$\det \Lambda = \lim_{r \to \infty} \frac{\text{vol}B(r, m)}{\#\{y \in \Lambda \, where \, \|y\| \leq r\}} \tag{6}$$

Theorem 2. *Given a lattice Λ with a basis $\{\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_i}\}$ then the determinant is equal to the volume of the parallelepiped spanned by $\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_m}$.*

Proof. Argument : It is coherent with our definition because when the radius of the ball r is big then $volB(r, n) \approx \#\{y \in \Lambda \, where \, \|y\| \leq r\}$ times the volume of the parallelepiped. Figure 2 is an illustration of this approximation. $\square$

Notice that for a full rank lattice, from linear algebra we know that $|\det[\mathbf{b_1}\ \mathbf{b_2}\ \ldots\ \mathbf{b_n}]|$ is the volume of the parallelepiped spanned by the basis vectors $\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_n}$, so

$$\det\Lambda = |\det[\mathbf{b_1}\ \mathbf{b_2}\ \ldots\ \mathbf{b_n}]|$$

Theorem 3 (Hadamard's inequality)**.** *The determinant is less than or equal to the product of the norm of the basis vectors.*

$$\det\Lambda \leq \prod_{i=1}^{m} \|\mathbf{b_i}\| \tag{7}$$

Equality holds if and only if the vectors $\mathbf{b_i}$ are pairwise orthogonal (if $\mathbf{b_i} \cdot \mathbf{b_j} = \mathbf{0}$ when $i \neq j$ and $\cdot$ is the scalar product).

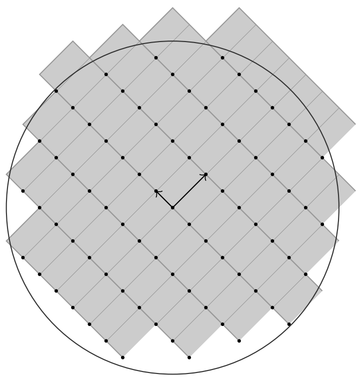


Figure 2: Illustration with a lattice of rank 2 that the volume of the ball approximates $\det\Lambda$ times the number of elements in Λ lying inside the ball.

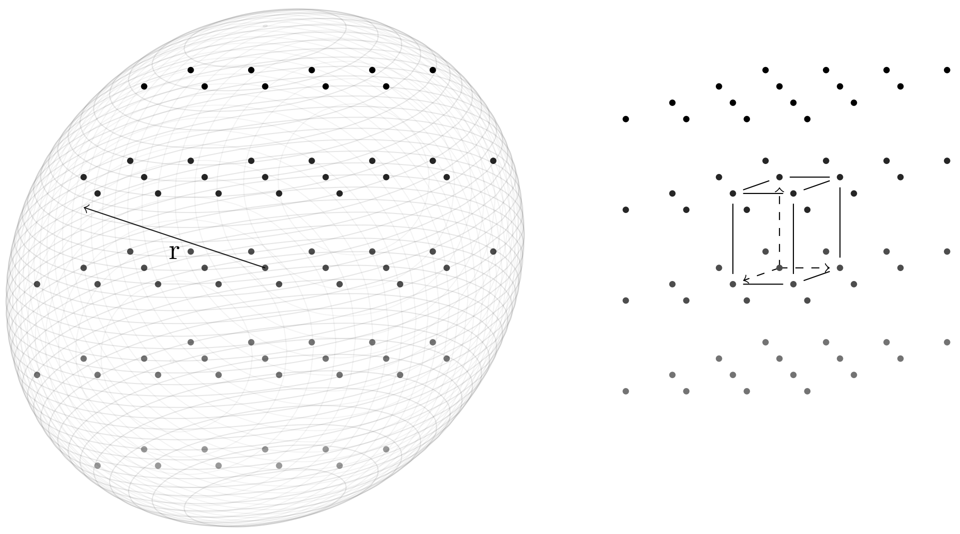


Figure 3: A lattice Λ of rank 3. On the left $\det \Lambda$ is represented as the ball and the elements of Λ inside. On the right $\det \Lambda$ is represented as the volume of the parallelepiped spanned by a basis of Λ.

1.4 Shortest vector problem

The shortest vector is the following : given a lattice Λ find a shortest vector $\mathbf{v}$ among the set of vectors going from the zero element to any non-zero element x in Λ. Notice that $-\mathbf{v}$ is also a shortest vector. In an algorithmic context, one may take 'shortest possible' to mean : shortest possible given the time one is willing to spend. The main theoretical result about this is the Minkowski's theorem which gives us an upper bound for the shortest vector.

Theorem 4 (Minkowski's theorem). *Given a lattice Λ of rank m, if λ is the norm of the shortest vector then :*

$$\lambda \leq \frac{2}{\sqrt{\pi}} \frac{m}{2}!^{\frac{1}{m}} \det \Lambda^{\frac{1}{m}} \tag{8}$$

Proof. Argument : If we place a m-dimensional ball of radius $\frac{\lambda}{2}$ on each element of Λ one can see that the balls are pairwise disjoint (cf Figure 4). From that one deduces that the volume of the ball is less than or equal than the determinant and the theorem follows :

$$volB(\frac{\lambda}{2}, m) \leq \det \Lambda$$

$$\frac{\frac{\lambda}{2}^m \pi^{m/2}}{\frac{m}{2}!} \leq \det \Lambda$$

$$\frac{\lambda}{2}^m \leq \frac{\frac{m}{2}!\det\Lambda}{\pi^{\frac{m}{2}}}$$

$$\lambda \leq \frac{2}{\sqrt{\pi}}\frac{m}{2}!^{\frac{1}{m}}\det\Lambda^{\frac{1}{m}}$$

Equality holds only with lattices of rank 1. □

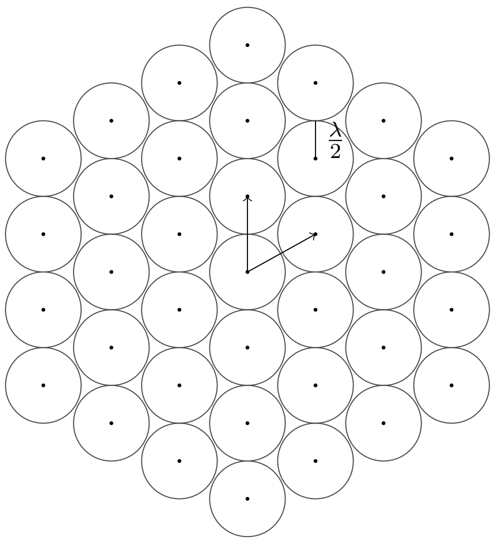


Figure 4: Illustration of a rank 2 lattice. One can see that the balls of radius $\frac{\lambda}{2}$ are pairwise disjoint. From that and the fact that the determinant is the inverse of the volume density of elements one concludes that $\det\Lambda < \frac{\pi\lambda^2}{4}$.

2 Basis reduction

2.1 Introduction

The idea of the basis reduction is to change a basis B of a lattice Λ into a shorter basis B' such that Λ remains the same. To do this we can use these following operations :

1. Swapping 2 vectors of the basis. As the swapping changes only the order of vectors in the basis it is trivial that Λ is not affected.
2. Replacing $\mathbf{b_j}$ by $-\mathbf{b_j}$. It is trivial that Λ is not affected.
3. Adding (or substracting) to a vector $\mathbf{b_j}$ a linear and discrete combination of the other vectors of the basis. The lattice is not affected because

if we take an arbitrary vector $\mathbf{v}$ which belongs to Λ we can express it as a discrete combination of the vectors of the basis : $\mathbf{v} = \sum_{i=1}^{m} z_i \mathbf{b_i}$ and if then we replace $\mathbf{b_j}$ by a discrete combination of the other vectors of the basis : $\mathbf{b_j} \leftarrow \mathbf{b_j} + \sum_{i \neq j} y_i \mathbf{b_i}$ we can still express $\mathbf{v}$ as a discrete combination of the vectors of the new basis : $\mathbf{v} = \sum_{i \neq j} z_i \mathbf{b_i} + z_j(\mathbf{b_j} - \sum_{i \neq j} y_i \mathbf{b_i})$. In a similar way we can show that if $\mathbf{v}$ belongs not to Λ, then we cannot express it with a discrete combination of the new basis. It follows that the 2 bases generate the exact same lattice.

Basis reduction can be used to solve the shortest vector problem in the sense that the shortest vector of the basis ($\mathbf{b_1}$ in the basis reduction algorithms we will see) is very short. In rank 2 for a reduced basis we have that $\mathbf{b_1}$ is the shortest vector of Λ and we can get it in polynomial time. But for higher ranks there is no known algorithms that finds the shortest vector in polynomial time. The LLL basis reduction algorithm finds a fairly short vector in polynomial time and it is often sufficient for applications.

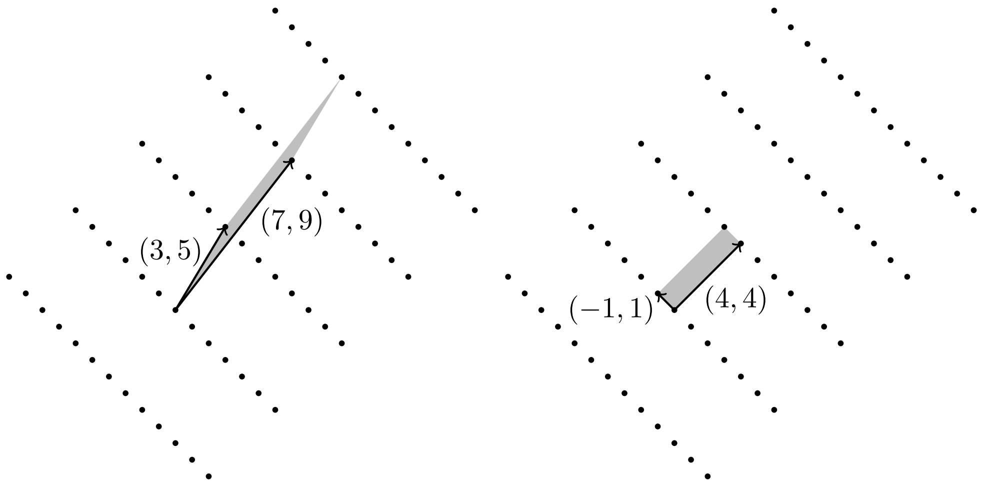


Figure 5: A lattice of rank 2 with two different bases. The determinant is depicted as the area of the parallelogram defined by the basis. The second basis is reduced and orthogonal.

2.2 Rank 2 basis reduction

Basis reduction of rank 2 lattices is easy to understand and plays a pivotal role in LLL basis reduction algorithm. We start with a basis $\{\mathbf{b_1}, \mathbf{b_2}\}$ and we try to reduce it. If $\mathbf{b_1}$ is shorter than $\mathbf{b_2}$ the intuitive approach is to substract from $\mathbf{b_2}$ an integer multiple z of $\mathbf{b_1}$. We want to choose z such that the new vector $\mathbf{b_2} - z\mathbf{b_1}$ is as short as possible. To solve this problem we take for z the coefficient u of the orthogonal projection of $\mathbf{b_2}$ on $\mathbf{b_1}$ (cf figure 6) rounded to the nearest integer. We repeat this process until we can no longer reduce the basis.

Definition 8 (Reduced basis in rank 2)**.** *A basis* $\{\mathbf{b_1}, \mathbf{b_2}\}$ *is said to be reduced if and only if the norm of* $\mathbf{b_1}$ *is less than or equal to the norm of* $\mathbf{b_2}$ *and the absolute value of the orthogonal projection coeffecient* $u = \frac{\mathbf{b_1} \cdot \mathbf{b_2}}{\mathbf{b_1} \cdot \mathbf{b_1}}$ *is less than or equal to* $\frac{1}{2}$*.*

$$\{\mathbf{b_1}, \mathbf{b_2}\}\, \textit{is reduced} \iff \|\mathbf{b_1}\| \leq \|\mathbf{b_2}\|\; \textit{and}\; \frac{|\mathbf{b_1} \cdot \mathbf{b_2}|}{\mathbf{b_1} \cdot \mathbf{b_1}} \leq \frac{1}{2} \tag{9}$$

To picture this observe that in figure 7 given an arbitrary $\mathbf{b_1}$ the basis is reduced if and only if $\mathbf{b_2}$ lies on the shaded area.

Theorem 5. *Given a lattice* Λ *of rank 2, if* λ *is the norm of the shortest vector then :*

$$\lambda \leq \sqrt{\frac{2}{\sqrt{3}} \det \Lambda} \tag{10}$$

Proof. Suppose that we have a reduced basis $\{\mathbf{b_1}, \mathbf{b_2}\}$ of Λ. Using orthogonal projection and the properties of reduced bases we get :

$$\mathbf{b_2} = \mathbf{b_2^*} + u\mathbf{b_1}$$

$$\|\mathbf{b_2}\|^2 = \|\mathbf{b_2^*}\|^2 + u^2\|\mathbf{b_1}\|^2$$

$$\|\mathbf{b_2^*}\|^2 = \|\mathbf{b_2}\|^2 - u^2\|\mathbf{b_1}\|^2 \geq \|\mathbf{b_1}\|^2 - \frac{1}{4}\|\mathbf{b_1}\|^2 = \frac{3}{4}\|\mathbf{b_1}\|^2$$

$$\|\mathbf{b_2^*}\| \geq \frac{\sqrt{3}}{2}\|\mathbf{b_1}\|$$

$$\|\mathbf{b_2^*}\|\|\mathbf{b_1}\| = \det \Lambda \geq \frac{\sqrt{3}}{2}\|\mathbf{b_1}\|^2$$

$$\sqrt{\frac{2}{\sqrt{3}} \det \Lambda} \geq \|\mathbf{b_1}\|$$

It gives us for rank 2 lattice a new bound for λ which is better than the bound given by the Minkowski's theorem (cf theorem 4). □

Theorem 6. *If a basis* $\{\mathbf{b_1}, \mathbf{b_2}\}$ *of* Λ *is reduced then* $\mathbf{b_1}$ *is a shortest vector of* Λ*.*

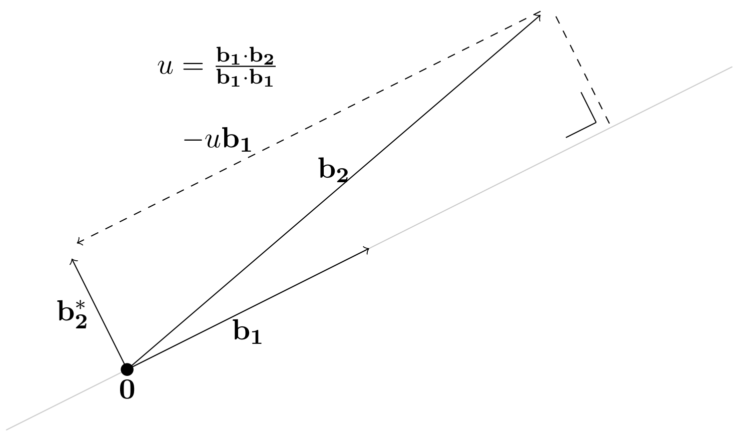
$$\{\mathbf{b_1}, \mathbf{b_2}\}\, \textit{is reduced} \;\Rightarrow\; \mathbf{b_1}\; \textit{is a shortest vector.} \tag{11}$$

Proof. Let $\mathbf{x}$ be a shortest vector of $\Lambda - \{\mathbf{0}\}$. We can express it with the reduced basis : $\mathbf{x} = z_1\mathbf{b_1} + z_2\mathbf{b_2}$. We have $\|\mathbf{x}\|^2 = \|z_1\mathbf{b_1} + z_2(\mathbf{b_2^*} + u\mathbf{b_1})\|^2 = (z_1 - z_2u)^2\|\mathbf{b_1}\|^2 + z_2^2\|\mathbf{b_2^*}\| \geq (z_1 - z_2u)^2\|\mathbf{b_1}\|^2 + \frac{3}{4}z_2^2\|\mathbf{b_1}\|^2$.

$$\|\mathbf{x}\|^2 \geq (z_1 - z_2u)^2\|\mathbf{b_1}\|^2 + \frac{3}{4}z_2^2\|\mathbf{b_1}\|^2$$

1. for $z_2 = 0$ and $z_1 \neq 0$: $\|\mathbf{x}\|^2 \geq z_1^2\|\mathbf{b_1}\|^2 \geq \|\mathbf{b_1}\|^2$
2. for $|z_2| = 1$: $\|\mathbf{x}\|^2 \geq (z_1 \pm u)^2\|\mathbf{b_1}\|^2 + \frac{3}{4}\|\mathbf{b_1}\|^2 \geq u^2\|\mathbf{b_1}\|^2 + \frac{3}{4}\|\mathbf{b_1}\|^2 \geq \frac{1}{4}\|\mathbf{b_1}\|^2 + \frac{3}{4}\|\mathbf{b_1}\|^2 = \|\mathbf{b_1}\|^2$
3. for $|z_2| \geq 2$: $\|\mathbf{x}\|^2 \geq (z_1 - z_2u)^2\|\mathbf{b_1}\|^2 + \frac{3}{4}4\|\mathbf{b_1}\|^2 > \|\mathbf{b_1}\|^2$

So we have $\|\mathbf{x}\|^2 \geq \|\mathbf{b_1}\|^2$ and as $\mathbf{x}$ is a shortest vector only equality can hold : $\|\mathbf{x}\|^2 = \|\mathbf{b_1}\|^2$. We conclude that $\mathbf{b_1}$ is also a shortest vector. $\square$



$u = \frac{\mathbf{b_1}\cdot\mathbf{b_2}}{\mathbf{b_1}\cdot\mathbf{b_1}}$
$-u\mathbf{b_1}$
$\mathbf{b_2}$
$\mathbf{b_2^*}$
$\mathbf{b_1}$
$\mathbf{0}$

Figure 6: Orthogonal projection. The vector $u\mathbf{b_1}$ is called the orthogonal projection of $\mathbf{b_2}$ on $\mathbf{b_1}$. The set $\{\mathbf{b_1}, \mathbf{b_2^*}\}$ is an orthogonal basis for the subspace generated by $\{\mathbf{b_1}, \mathbf{b_2}\}$. Notice that the lattice Λ generated by $\{\mathbf{b_1}, \mathbf{b_2}\}$ has $\det\Lambda = \|\mathbf{b_1}\|\|\mathbf{b_2^*}\|$.

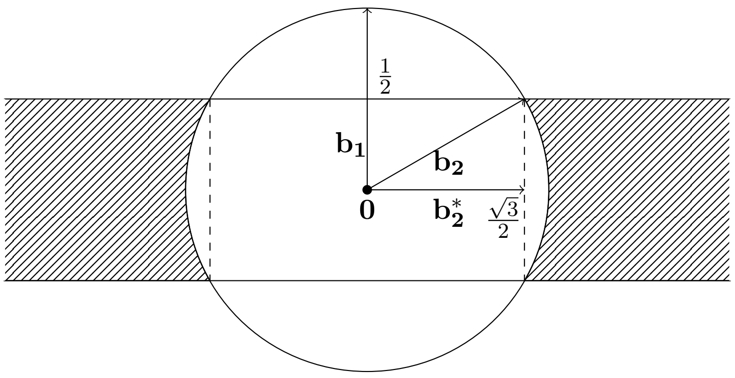


Figure 7: If the basis is reduced then $\mathbf{b_2}$ lies on the shaded area. We can see that for a reduced basis we have $\|\mathbf{b_2^*}\| \geq \frac{\sqrt{3}}{2}\|\mathbf{b_1}\|$.

Algorithm 1: Rank 2 basis reduction.

```
input  : basis { $\mathbf{b_1}$, $\mathbf{b_2}$ }
output: reduced basis { $\mathbf{b_1}$, $\mathbf{b_2}$ }
if $\|\mathbf{b_1}\| > \|\mathbf{b_2}\|$ then
    $swap(\mathbf{b_1}, \mathbf{b_2})$
while $\|\mathbf{b_1}\| < \|\mathbf{b_2}\|$ do
    $u = (\mathbf{b_1} \cdot \mathbf{b_2})/(\mathbf{b_1} \cdot \mathbf{b_1})$
    $\mathbf{b_2} = \mathbf{b_2} - round(u)\mathbf{b_1}$
    $swap(\mathbf{b_1}, \mathbf{b_2})$
$swap(\mathbf{b_1}, \mathbf{b_2})$ // to have $\|\mathbf{b_1}\| \leq \|\mathbf{b_2}\|$
```

2.3 LLL basis reduction

The LLL-reduction algorithm (Lenstra Lenstra Lovász lattice basis reduction) is a polynomial time lattice reduction algorithm invented by Arjen Lenstra, Hendrik Lenstra and László Lovász in 1982. Since no efficient (polynomial time) algorithm is known to solve the shortest vector problem exactly in arbitrary high dimension, LLL is used to get an approximation of the shortest vector. This approximation is sufficient for many applications.

Roughly speaking, LLL performs successives orthogonal projections, if necessary swapping 2 consecutives vectors of the basis, in order to get a reduced or near orthogonal basis.

Theorem 7. *Gram-Schmidt orthogonalization method. Given a basis* $\{\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_m}\}$ *of a subspace* H_m *of* $\mathbb{R}^n$*, we define :*

$$\mathbf{b_1^*} = \mathbf{b_1}$$

$$\mathbf{b_2^*} = \mathbf{b_2} - \frac{\mathbf{b_2} \cdot \mathbf{b_1^*}}{\mathbf{b_1^*} \cdot \mathbf{b_1^*}} \mathbf{b_1^*}$$

$$\mathbf{b_3^*} = \mathbf{b_3} - \frac{\mathbf{b_3} \cdot \mathbf{b_1^*}}{\mathbf{b_1^*} \cdot \mathbf{b_1^*}} \mathbf{b_1^*} - \frac{\mathbf{b_3} \cdot \mathbf{b_2^*}}{\mathbf{b_2^*} \cdot \mathbf{b_2^*}} \mathbf{b_2^*}$$

$$\vdots$$

$$\mathbf{b_m^*} = \mathbf{b_m} - \frac{\mathbf{b_m} \cdot \mathbf{b_1^*}}{\mathbf{b_1^*} \cdot \mathbf{b_1^*}} \mathbf{b_1^*} - \frac{\mathbf{b_m} \cdot \mathbf{b_2^*}}{\mathbf{b_2^*} \cdot \mathbf{b_2^*}} \mathbf{b_2^*} - \cdots - \frac{\mathbf{b_m} \cdot \mathbf{b_{m-1}^*}}{\mathbf{b_{m-1}^*} \cdot \mathbf{b_{m-1}^*}} \mathbf{b_{m-1}^*}$$

Then for $1 \leq k \leq m$ *and* H_k *the subspace generated by the basis* $\{\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_k}\}$

$$\{\mathbf{b_1^*}, \mathbf{b_2^*}, \ldots, \mathbf{b_k^*}\} \textit{ is an orthogonal basis of } H_k \tag{12}$$

Proof. 1. For $k = 1$: It is trivial because $\mathbf{b_1^*} = \mathbf{b_1}$.

2. For $k = 2$:

   - $\{\mathbf{b_1^*}, \mathbf{b_2^*}\}$ is orthogonal because $\mathbf{b_2^*}$ is constructed using the orthogonal projection of $\mathbf{b_2}$ on $\mathbf{b_1^*}$.
   - As $\mathbf{b_2^*}$ is obtained substracting from $\mathbf{b_2}$ a multiple of $\mathbf{b_1^*}$ or equally a multiple of $\mathbf{b_1}$ since $\mathbf{b_1^*} = \mathbf{b_1}$ and the fact that substracting from a vector of a basis a linear combination of the other vectors of the basis do not modify the subspace then it follows that $\{\mathbf{b_1^*}, \mathbf{b_2^*}\}$ is a basis of H_2.

3. For $2 < k \leq m$:

   - $\{\mathbf{b_1^*}, \mathbf{b_2^*}, \ldots, \mathbf{b_k^*}\}$ is orthogonal because $\{\mathbf{b_1^*}, \mathbf{b_2^*}, \ldots, \mathbf{b_{k-1}^*}\}$ is an orthogonal basis by induction hypothesis and $\mathbf{b_k^*}$ is constructed using successive orthogonal projections of $\mathbf{b_k^*}$ on the vectors $\mathbf{b_1^*}, \mathbf{b_2^*}, \ldots, \mathbf{b_{k-1}^*}$ such that $\mathbf{b_k^*}$ is pairwise orthogonal with them.
   - As $\mathbf{b_k^*}$ is obtained substracting from $\mathbf{b_k}$ a linear combination of the vectors $\mathbf{b_1^*}, \mathbf{b_2^*}, \ldots, \mathbf{b_{k-1}^*}$ or equally a linear combination of the vectors $\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_{k-1}}$ since by induction hypothesis we have that $\{\mathbf{b_1^*}, \mathbf{b_2^*}, \ldots, \mathbf{b_{k-1}^*}\}$ is a basis of H_{k-1} and the fact that substracting from a vector of a basis a linear combination of the other vectors of the basis do not modify the subspace then it follows that $\{\mathbf{b_1^*}, \mathbf{b_2^*}, \ldots, \mathbf{b_k^*}\}$ is a basis of H_k.

□

QR matrix factorisation : If we define the matrices $B = [\mathbf{b_1}\ \mathbf{b_2}\ \ldots\ \mathbf{b_m}]$, $Q = [\mathbf{b_1^*}\ \mathbf{b_2^*}\ \ldots\ \mathbf{b_m^*}]$, and $R = [\mathbf{u_1}\ \mathbf{u_2}\ \ldots\ \mathbf{u_j}]$ such that $\mathbf{u_i} \in R^m$ and the j^{th} element of $\mathbf{u_i}$ is defined as follows :

- $\mathbf{u_i}[j] = (\mathbf{b_i} \cdot \mathbf{b_j^*})/(\mathbf{b_j^*} \cdot \mathbf{b_j^*})$ if $j < i$

- $\mathbf{u_i}[j] = 1$ if $j = i$
- $\mathbf{u_i}[j] = 0$ if $j > i$

One has $B = QR$. Doing the matrix multiplication one notices that this is an equivalent way of expressing the Gram-Schmidt orthogonalization method. Since R is an upper triangular matrix with only 1's on the diagonal then $\det R = 1$.

Theorem 8. *Given a lattice Λ generated by the basis $\{\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_m}\}$ and the orthogonal basis $\{\mathbf{b_1^*}, \mathbf{b_2^*}, \ldots, \mathbf{b_m^*}\}$ obtained using the Gram-Schmidt method, then :*

$$\det \Lambda = \prod_{i=1}^{m} \|\mathbf{b_i^*}\| \tag{13}$$

Proof. First notice that Λ and the lattice generated by $\{\mathbf{b_1^*}, \mathbf{b_2^*}, \ldots, \mathbf{b_m^*}\}$ are not the same. The proof come from the fact that one can transform the basis $\{\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_m}\}$ into the orthogonal basis $\{\mathbf{b_1^*}, \mathbf{b_2^*}, \ldots, \mathbf{b_m^*}\}$ using only the operation that consists of substracting from a vector of the basis a linear combination of the other vectors of the basis, which do not modify the volume of the parallelepiped spanned by the basis. Such transformation is done by rewriting the Gram-Schmidt method as follows :

$$\mathbf{b_1} \leftarrow \mathbf{b_1}$$

$$\mathbf{b_2} \leftarrow \mathbf{b_2} - \frac{\mathbf{b_2} \cdot \mathbf{b_1}}{\mathbf{b_1} \cdot \mathbf{b_1}} \mathbf{b_1}$$

$$\mathbf{b_3} \leftarrow \mathbf{b_3} - \frac{\mathbf{b_3} \cdot \mathbf{b_1}}{\mathbf{b_1} \cdot \mathbf{b_1}} \mathbf{b_1} - \frac{\mathbf{b_3} \cdot \mathbf{b_2}}{\mathbf{b_2} \cdot \mathbf{b_2}} \mathbf{b_2}$$

$$\vdots$$

$$\mathbf{b_m} \leftarrow \mathbf{b_m} - \frac{\mathbf{b_m} \cdot \mathbf{b_1}}{\mathbf{b_1} \cdot \mathbf{b_1}} \mathbf{b_1} - \frac{\mathbf{b_m} \cdot \mathbf{b_2}}{\mathbf{b_2} \cdot \mathbf{b_2}} \mathbf{b_2} - \cdots - \frac{\mathbf{b_m} \cdot \mathbf{b_{m-1}}}{\mathbf{b_{m-1}} \cdot \mathbf{b_{m-1}}} \mathbf{b_{m-1}}$$

As $\det \Lambda$ is equal to the volume of the parallelepiped spanned by $\{\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_m}\}$, which is equal to the volume of the other parallelepiped spanned by $\{\mathbf{b_1^*}, \mathbf{b_2^*}, \ldots, \mathbf{b_m^*}\}$, which is a parallelepiped rectangle whose volume is equal to the product of its edges, one concludes that $\det \Lambda = \prod_{i=1}^{m} \|\mathbf{b_i^*}\|$.
Notice that if Λ is of a full rank lattice then using the QR factorisation we have that $B = QR$ which implies that $\det \Lambda = |\det B| = |\det Q||\det R| = |\det Q| = \prod_{i=1}^{m} \|\mathbf{b_i^*}\|$. □

Definition 9. *c-Reduced basis. A basis* $\{\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_m}\}$ *is said to be c-reduced if and only if its orthogonal basis obtained with the Gram-Schmidt method* $\{\mathbf{b_1^*}, \mathbf{b_2^*}, \ldots, \mathbf{b_m^*}\}$ *verifies the following inequality for* $i = 1$ *to* $m - 1$ *:*

$$\|\mathbf{b_{i+1}^*}\|^2 \geq \frac{\|\mathbf{b_i^*}\|^2}{c} \tag{14}$$

A small value for c means a good reduction. Not every basis is 1-reducable, but each basis is $\frac{4}{3}$-reducable. The $\frac{4}{3}$ comes from the $\frac{\sqrt{3}}{2}$ one can see in figure 7. Figure 8 shows c-reduced basis in rank 2. Notice from Gram-Schmidt method that $\mathbf{b_1^*} = \mathbf{b_1}$.

Theorem 9. *Near-orthogonality of c-reduced basis. Given a lattice* Λ *and its c-reduced basis* $\{\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_m}\}$ *with* $c \geq \frac{4}{3}$ *then it is near-orthogonal in the sense that :*

$$\prod_{i=1}^{m} \|\mathbf{b_i}\| \leq c^{(m(m-1))/4} \det \Lambda \tag{15}$$

Proof. Multiplying for $i = 1$ to m the following inequality

$$\|\mathbf{b_i}\|^2 \leq c^{i-1} \|\mathbf{b_i^*}\|^2$$

and then taking the square root and using $\det \Lambda = \prod_{i=1}^{m} \mathbf{b_i^*}$ we conclude

$$\prod_{i=1}^{m} \|\mathbf{b_i}\| \leq c^{(m(m-1))/4} \det \Lambda$$

□

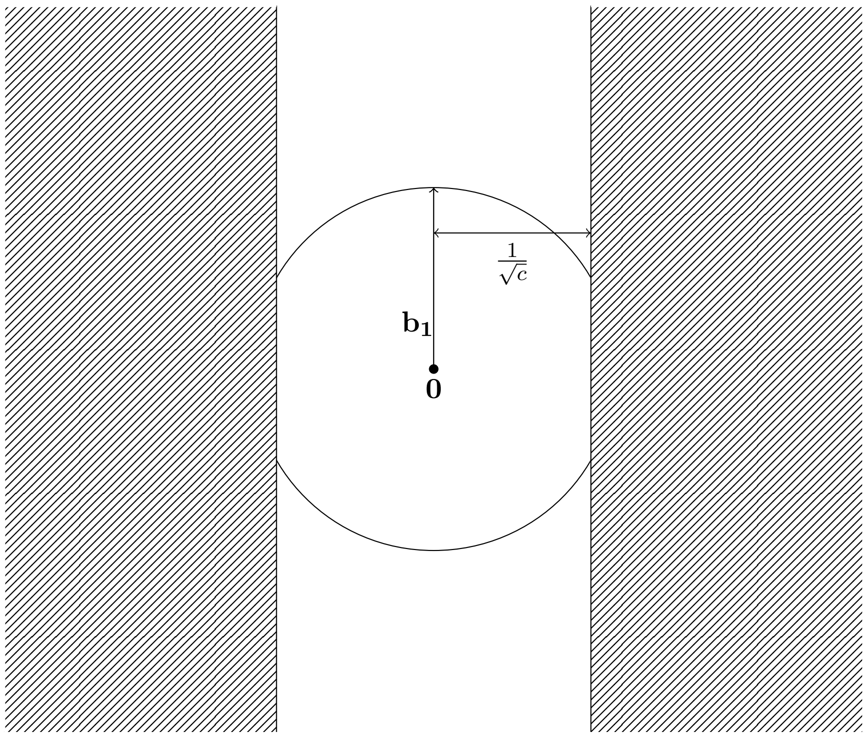


Figure 8: c-reduced basis in rank 2. Given an arbitrary $\mathbf{b_1}$, the basis is c-reduced if and only if $\mathbf{b_2}$ lies on the shaded area. Here $c = \frac{4}{3}$ which is the minimal value for c.

Theorem 10. *Shortest vector approximation with a c-reduced basis. If the basis* $\{\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_m}\}$ *is c-reduced and* λ *is the shortest vector norm then :*

$$\|\mathbf{b_1}\| \leq c^{(m-1)/4} \det \Lambda^{\frac{1}{m}} \tag{16}$$

$$\|\mathbf{b_1}\| \leq c^{(m-1)/2} \lambda \tag{17}$$

Proof. 1. From the definition of a c-reduced basis we have that for $i = 1$ to m

$$\|\mathbf{b_1}\|^2 = \|\mathbf{b_1^*}\|^2 \leq c^{i-1} \|\mathbf{b_i^*}\|^2.$$

Multiplying together all this inequalities we get

$$\|\mathbf{b_1}\|^{2m} \leq c^{0+1+2+\cdots+(m-1)} \prod_{i=1}^{m} \|\mathbf{b_i^*}\|^2.$$

As from theorem 8 we have $\det \Lambda = \prod_{i=1}^{m} \|\mathbf{b_i^*}\|$ and that $\sum_{i=0}^{m-1} i = \frac{m(m-1)}{2}$ we get

$$\|\mathbf{b_1}\|^{2m} \leq c^{m(m-1)/2} \det \Lambda^2.$$

Passing the inequality to the power $\frac{1}{2m}$ we get

$$\|\mathbf{b_1}\| \leq c^{(m-1)/4} \det \Lambda^{\frac{1}{m}}.$$

2. Let $x \in \Lambda - \mathbf{0}$, and let i be minimal such that $x \in \Lambda_i$ which is the sublattice of Λ generated by $\{\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_i}\}$. Notice that $\|x\|$ is at least $\|\mathbf{b_i^*}\|$ then it follows that

$$\lambda^2 \geq \|x\|^2 \geq \|\mathbf{b_i^*}\|^2 \geq \frac{\|\mathbf{b_1^*}\|^2}{c^{(i-1)}} \geq \frac{\|\mathbf{b_1}\|^2}{c^{(m-1)}}.$$

Multiplying by c^{m-1} and passing to the power $\frac{1}{2}$ we prove that

$$c^{(m-1)/2}\lambda \geq \|\mathbf{b_1}\|.$$

□

Theorem 11. *For $c > \frac{4}{3}$ LLL finds a c-reduced basis in polynomial time*

Proof. We define the size of the orthogonal basis $\{\mathbf{b_1^*}, \mathbf{b_2^*}, \ldots, \mathbf{b_m^*}\}$ as

$$s = \|\mathbf{b_1^*}\|^m \, \|\mathbf{b_2^*}\|^{m-1} \, \ldots \, \|\mathbf{b_m^*}\|$$

and $s_{initial}$ as the size of the basis orthogonal at the initialization of LLL and s_{final} the size of the orthogonal basis at termination of LLL. It will be usefull to analyse the effect of swapping $\mathbf{b_i^*}$ and $\mathbf{b_{i+1}^*}$ when $\|\mathbf{b_i^*}\|^2 > c\|\mathbf{b_{i+1}^*}\|^2$ on the new orthogonal basis $\{\mathbf{a_1^*}, \mathbf{a_2^*}, \ldots, \mathbf{a_m^*}\}$ and especially on s. Let s_b be the size before and s_a the size after the swapping and compare them :

1. $$s_b = \|\mathbf{b_1^*}\|^m \, \|\mathbf{b_2^*}\|^{m-1} \, \ldots \, \|\mathbf{b_i^*}\|^{m-i+1} \, \|\mathbf{b_{i-1}^*}\|^{m-i} \, \ldots \, \|\mathbf{b_m^*}\|$$
$$s_b = \|\mathbf{b_1^*}\|^m \, \|\mathbf{b_2^*}\|^{m-1} \, \ldots \, \|\mathbf{b_i^*}\| \, (\|\mathbf{b_i^*}\| \, \|\mathbf{b_{i+1}^*}\|)^{m-i} \, \ldots \, \|\mathbf{b_m^*}\|$$

2. $$s_a = \|\mathbf{a_1^*}\|^m \, \|\mathbf{a_2^*}\|^{m-1} \, \ldots \, \|\mathbf{a_i^*}\|^{m-i+1} \, \|\mathbf{a_{i+1}^*}\|^{m-i} \, \ldots \, \|\mathbf{a_m^*}\|$$

   As we can notice it in the algorithm the swapping only affects the $\mathbf{b_i^*}$ and $\mathbf{b_{i+1}^*}$ (for the vectors before the i^{th} it is trivial and for the ones after the $(i+1)^{th}$ it comes from the fact that $\mathbf{b_{i+2}^*}$ is constructed using the orthogonal projection of $\mathbf{b_{i+2}}$ on the subspace $\{\mathbf{b_1^*}, \mathbf{b_2^*}, \ldots, \mathbf{b_{i+1}^*}\}$ which is not affected by the swapping and so on for $\mathbf{b_{i+3}^*}, \ldots, \mathbf{b_m^*}$) and we get :

$$s_a = \|\mathbf{b_1^*}\|^m \, \|\mathbf{b_2^*}\|^{m-1} \, \ldots \, \|\mathbf{a_i^*}\| \, (\|\mathbf{a_i^*}\| \, \|\mathbf{a_{i+1}^*}\|)^{m-i} \, \ldots \, \|\mathbf{b_m^*}\|$$

Algorithm 2: LLL basis reduction.

```
Input: basis { $\mathbf{b_1}$, $\mathbf{b_2}$, ..., $\mathbf{b_m}$ } and $c$ such that $\mathbf{b_i} \in \mathbb{R}^n$ and $c \geq \frac{4}{3}$
Data: orthogonal basis { $\mathbf{b_1^*}$, $\mathbf{b_2^*}$, ..., $\mathbf{b_m^*}$ } and orthogonal
      projection coefficent vectors { $\mathbf{u_1}$, $\mathbf{u_2}$, ..., $\mathbf{u_m}$ } such that
      $\mathbf{b_i^*} \in \mathbb{R}^n$ and $\mathbf{u_i} \in \mathbb{R}^m$
Output: c-reduced basis { $\mathbf{b_1}$, $\mathbf{b_2}$, ..., $\mathbf{b_m}$}

for $i \leftarrow 1$ to $m$ do // initialization
    $\mathbf{u_i} = \mathbf{0}$
    $\mathbf{u_i}[i] = 1$
    $\mathbf{b_i^*} = \mathbf{b_i}$
    for $j \leftarrow 1$ to $i-1$ do
        $\mathbf{u_i}[j] = (\mathbf{b_i} \cdot \mathbf{b_j^*})/(\mathbf{b_j^*} \cdot \mathbf{b_j^*})$
        $\mathbf{b_i^*} = \mathbf{b_i^*} - \mathbf{u_i}[j]\,\mathbf{b_j^*}$
    reduce($i$)

while $i \leftarrow 1 < m$ do
    if $\|\mathbf{b_i^*}\|^2 \leq c\|\mathbf{b_{i+1}^*}\|^2$ then // $\{\mathbf{b_1}, \dots, \mathbf{b_{i+1}}\}$ is c-reduced
        $\mathrm{i} = \mathrm{i} + 1$
    else
        /* modify Q and R in order to keep the relation B =
        QR after the swapping                               */
        $\mathbf{b_{i+1}^*} = \mathbf{b_{i+1}^*} + \mathbf{u_{i+1}}[i]\,\mathbf{b_i^*}$
        $\mathbf{u_i}[i] = (\mathbf{b_i} \cdot \mathbf{b_{i+1}^*})/(\mathbf{b_{i+1}^*} \cdot \mathbf{b_{i+1}^*})$
        $\mathbf{u_i}[i+1] = 1$
        $\mathbf{u_{i+1}}[i] = 1$
        $\mathbf{u_{i+1}}[i+1] = 0$
        $\mathbf{b_i^*} = \mathbf{b_i^*} - \mathbf{u_i}[i]\,\mathbf{b_{i+1}^*}$
        swap($\mathbf{u_i}, \mathbf{u_{i+1}}$)
        swap($\mathbf{b_i^*}, \mathbf{b_{i+1}^*}$)
        swap($\mathbf{b_i}, \mathbf{b_{i+1}}$)
        for $k \leftarrow i+2$ to $m$ do
            $\mathbf{u_k}[i] = (\mathbf{b_k} \cdot \mathbf{b_i^*})/(\mathbf{b_i^*} \cdot \mathbf{b_i^*})$
            $\mathbf{u_k}[i+1] = (\mathbf{b_k} \cdot \mathbf{b_{i+1}^*})/(\mathbf{b_{i+1}^*} \cdot \mathbf{b_{i+1}^*})$
        if $|\mathbf{u_{i+1}}[i]| > \frac{1}{2}$ then reduce($i+1$)
        $i = \max(i-1, 1)$

Subroutine: reduce
  Input: $i$ such that $(i \leq m)$

  while $j \leftarrow (i-1) > 0$ do
      $\mathbf{b_i} = \mathbf{b_i} - \mathrm{round}(\mathbf{u_i}[j])\mathbf{b_j}$
      $\mathbf{u_i} = \mathbf{u_i} - \mathrm{round}(\mathbf{u_i}[j])\mathbf{u_j}$
      $j = j - 1$
```

From theorem 8 we have $\det \Lambda = \prod_{k=1}^{m} \|\mathbf{b_k^*}\| = \prod_{i=k}^{k} \|\mathbf{a_k^*}\|$ and it implies that $\|\mathbf{b_i^*}\| \, \|\mathbf{b_{i+1}^*}\| = \|\mathbf{a_i^*}\| \, \|\mathbf{a_{i+1}^*}\|$ and we get :

$$s_a = \|\mathbf{b_1^*}\|^m \, \|\mathbf{b_2^*}\|^{m-1} \dots \|\mathbf{a_i^*}\| \, (\|\mathbf{b_i^*}\| \, \|\mathbf{b_{i+1}^*}\|)^{m-i} \dots \|\mathbf{b_m^*}\|$$

As we can notice it from the algorithm $\mathbf{a_i^*} = \mathbf{b_{i+1}^*} + \mathbf{u_{i+1}}[i] \, \mathbf{b_i^*}$ which is simply $\mathbf{b_{i+1}^*}$ without the projection component on $\mathbf{b_i^*}$. So $\|\mathbf{a_i^*}\|^2 = \|\mathbf{b_{i+1}^*}\|^2 + \mathbf{u_{i+1}}[i]^2 \, \|\mathbf{b_i^*}\|^2$ and we get :

$$s_a = \|\mathbf{b_1^*}\|^m \|\mathbf{b_2^*}\|^{m-1} \dots (\|\mathbf{b_{i+1}^*}\|^2 + \mathbf{u_{i+1}}[i]^2 \, \|\mathbf{b_i^*}\|^2)^{\frac{1}{2}} (\|\mathbf{b_i^*}\| \|\mathbf{b_{i+1}^*}\|)^{m-i} \dots \|\mathbf{b_m^*}\|$$

Combining together s_b and s_a we get :

$$s_b = \frac{s_a \, (\|\mathbf{b_{i+1}^*}\|^2 + \mathbf{u_{i+1}}[i]^2 \, \|\mathbf{b_i^*}\|^2)^{\frac{1}{2}}}{\|\mathbf{b_i^*}\|}$$

Finally using $\mathbf{u_{i+1}}[i]^2 \le \frac{1}{4}$ and multiplying by $\|\mathbf{b_i^*}\|$ and as $\|\mathbf{b_i^*}\|^2 > c\|\mathbf{b_{i+1}^*}\|^2$ we conclude :

$$s_b \le \left(\frac{\|\mathbf{b_{i+1}^*}\|^2}{\|\mathbf{b_i^*}\|^2} + \frac{1}{4} \right)^{\frac{1}{2}} s_a < \left(\frac{1}{c} + \frac{1}{4} \right)^{\frac{1}{2}} s_a$$

From this result we observe that the number of times swapping happens in LLL is at most

$$\frac{2 \log (s_{initial}/s_{final})}{|\log(\frac{1}{c} + \frac{1}{4})|}$$

Notice that this expression is not defined for $c = \frac{4}{3}$ because a division by 0 occurs. So it suffices to take $c > \frac{4}{3}$ and a good lower bound for s_{final} to prove that LLL runs in polynomial time. □

3 LLL basis reduction for solving RSA problem

In this section, we will show how we can use Coppersmith's algorithm for finding small roots of univariate modular polynomials, which uses LLL, in order to attack the RSA cryptosystem under certain conditions.

3.1 RSA

Rivest Shamir Adleman or RSA is a very well known asymetric public key cryptographic algorithm used to exchange confidential information over the Internet. Here is how it works :

- Keys generation :

   1. Choose 2 prime numbers p and q.

2. Denote n as the product of p and $q : n = pq$.
3. Calculate Euler's totient function of $n : \phi(n) = (p-1)(q-1)$.
4. Choose an integer e coprime with $\phi(n) : \gcd(e, \phi(n)) = 1$.
5. Compute d as the inverse modulo $\phi(n)$ of $e : ed \equiv 1 \mod \phi(n)$.

- Encryption : $C = M^e \mod n$ where M is the message and C the message encrypted.
- Decryption : $M = C^d \mod n$ where M is the message and C the message encrypted.

Definition 10 (RSA problem)**.** *Given $M^e \mod N$ find $M \in \mathbb{Z}_N$.*

Definition 11 (Relaxed RSA problem : Small e, High Bits Known)**.** *Given $M^e, \tilde{M}$ with $|M - \tilde{M}| \leq N^{\frac{1}{e}}$ find $M \in \mathbb{Z}_N$.*

3.2 Coppersmith

In this section we present the Coppersmith method to find small roots of a monic univariate polynomial : We want to efficiently find all the solutions x_0 satisfying

$$f(x_0) = 0 \mod N \text{ with } |x_0| \leq X \tag{18}$$

Theorem 12 (Howgrave-Graham)**.** *Let $g(x)$ be an univariate polynomial of degree δ. Further, let m be a positive integer. Suppose that*

$$g(x_0) = 0 \mod N^m \text{ where } |x_0| \leq X \tag{19}$$

$$\|g(xX)\| < \frac{N^m}{\sqrt{\delta + 1}} \tag{20}$$

Then $g(x_0) = 0$ holds over the integers.

Proof. $|g(x_0)| = |\sum_{i=0}^{\delta} c_i x_0^i| \leq \sum_{i=0}^{\delta} |c_i x_0^i| \leq \sum_{i=0}^{n} |c_i| X^i \leq \sqrt{\delta+1} \|g(xX)\| < N^m$. But $g(x_0)$ is a multiple of N^m and therefore it must be zero. □

Given the Howgrave-Graham theorem, the idea is to construct a collection $f_1(x), ..., f_n(x)$ of polynomials that all have the desired roots x_0 modulo N^m. Notice that for every integer linear combination g we have

$$g(x_0) = \sum_{i=1}^{n} a_i f_i(x_0) = 0 \mod N^m \text{ where } a_i \in \mathbb{Z}. \tag{21}$$

Then, using LLL basis reduction on the coefficient vectors of $f_i(xX)$, we might find a small coefficient vector $\mathbf{v}$ such that $\mathbf{v}$ respects the second condition of the Howgrave-Graham theorem : $\|\mathbf{v}\| < \frac{N^m}{\sqrt{n}}$ where n is the dimension of the coefficient vector $\mathbf{v}$.

Theorem 13 (Coppersmith)**.** *Let f(x) be a univariate monic polynomial of degree δ. Let N be an integer of unknown factorization. And let $\epsilon > 0$. Then we can find all solutions x_0 for the equation*

$$f(x) = 0 \mod N \textit{ with } |x_0| \leq \frac{1}{2} N^{\frac{1}{\delta}-\epsilon}. \tag{22}$$

Proof. To prove this we apply the coppersmith method. First we set $m = \lceil 1/(\delta\epsilon) \rceil$ and $X = \frac{1}{2}N^{\frac{1}{\delta}-\epsilon}$. Then we construct a collection of polynomials, where each polynomial has a root x_0 modulo N^m whenever $f(x)$ has the root x_0 modulo N. Here is the collection of polynomials we choose :

$$\begin{array}{lllll}
N^m & xN^m & x^2N^m & \dots & x^{\delta-1}N^m \\
N^{m-1}f & xN^{m-1}f & x^2N^{m-1}f & \dots & x^{\delta-1}N^{m-1}f \\
N^{m-2}f^2 & xN^{m-2}f^2 & x^2N^{m-2}f^2 & \dots & x^{\delta-1}N^{m-2}f^2 \\
\vdots & \vdots & \vdots & & \vdots \\
Nf^{m-1} & xNf^{m-1} & x^2Nf^{m-1} & \dots & x^{\delta-1}Nf^{m-1}
\end{array}$$

Or more compactly :

$$g_{i,j}(x) = x^j N^i f^{m-i}(x) \; for \; i = 1, \dots, m \; and \; j = 0, \dots, \delta - 1$$

We can see that $g_{i,j}(x_0) = 0 \mod N^m$ if $f(x_0) = 0 \mod N$ noticing that N^i is divisible i times by N and $f(x_0)^{m-i}$ is divisible $m - i$ times by N. Then we construct the lattice Λ that is spanned by the coefficent vectors of $g_{i,j}(xX)$:

$$\begin{bmatrix}
0 & 0 & & 0 & & & 0 & 0 & & NX^{\delta m-1} \\
\vdots & \vdots & & \vdots & & & \vdots & \vdots & \iddots & - \\
\vdots & \vdots & & \vdots & & & 0 & NX^{\delta m-\delta+1} & & \vdots \\
\vdots & \vdots & & \vdots & & & NX^{\delta m-\delta} & - & & \vdots \\
\vdots & \vdots & & \vdots & & \iddots & - & \vdots & & \vdots \\
\vdots & \vdots & & 0 & \iddots & \iddots & \vdots & \vdots & & \vdots \\
\vdots & \vdots & & N^mX^{\delta-1} & \iddots & \iddots & \vdots & \vdots & & \vdots \\
\vdots & 0 & \iddots & 0 & \iddots & \iddots & \vdots & \vdots & & \vdots \\
0 & N^mX & & \vdots & \iddots & \iddots & \vdots & \vdots & & \vdots \\
N^m & 0 & & 0 & \iddots & \iddots & - & - & & -
\end{bmatrix}$$

Notice that the rank of the lattice is δm. A nice thing about the matrix is that it is triangular. So we can easily compute the determinant of the lattice multiplying the terms on the diagonal :

$$\det \Lambda = N^{\frac{1}{2}\delta m(m+1)} X^{\frac{1}{2}\delta m(\delta m-1)}.$$

Then we reduce the lattice with LLL choosing $c = 2$ and we obtain a 2-reduced basis. We want the first vector of the basis $\mathbf{b_1}$ to satisfy the condition

$$\|\mathbf{b_1}\| < \frac{N^m}{\sqrt{\delta m}}$$

in order to apply the Howgrave-Graham theorem. Let's see if it does : After the LLL basis reduction we are guaranteed to have

$$\|\mathbf{b_1}\| \leq 2^{\frac{\delta m-1}{4}} \det \Lambda^{\frac{1}{\delta m}}.$$

So we need to prove that

$$2^{\frac{n-1}{4}} \det \Lambda^{\frac{1}{\delta m}} \leq \frac{N^m}{\sqrt{\delta m}}.$$

Using the fact that $\det \Lambda = N^{\frac{1}{2}\delta m(m+1)} X^{\frac{1}{2}\delta m(\delta m-1)}$, we obtain the new condition :

$$N^{\frac{\delta m(m+1)}{2\delta m}} X^{\frac{\delta m-1}{2}} \leq 2^{\frac{-(\delta m-1)}{4}} (\delta m)^{\frac{-1}{2}} N^m.$$

This gives us a condition on the size of X :

$$X \leq 2^{\frac{-1}{2}} (\delta m)^{\frac{-1}{\delta m-1}} N^{\frac{2m}{\delta m-1} - \frac{\delta m(m+1)}{\delta m(\delta m-1)}}.$$

Notice that $(\delta m)^{\frac{-1}{\delta m-1}} = 2^{\frac{-\log(\delta m)}{\delta m-1}} \geq 2^{\frac{-1}{2}}$ for $n > 6$. Therefore, our condition simplifies to

$$X \leq \frac{1}{2} N^{\frac{2m}{\delta m-1} - \frac{m+1}{\delta m-1}}.$$

Remember that we made the choice $X = \frac{1}{2} N^{\frac{1}{\delta} - \epsilon}$. Hence in order to finish the proof of the theorem, it suffices to show that

$$\frac{2m}{\delta m - 1} - \frac{m(1 + \frac{1}{m})}{\delta m - 1} \geq \frac{1}{\delta} - \epsilon.$$

Then multiplying by $\frac{\delta m-1}{\delta m}$ we get :

$$\frac{2}{\delta} - \frac{1}{\delta}(1 + \frac{1}{m}) \geq \frac{1}{\delta} - \epsilon.$$

This simplifies to

$$\frac{-1}{\delta} \frac{1}{m} \geq -\epsilon.$$

and finally it gives us the condition

$$m \geq \frac{1}{\delta \epsilon},$$

which holds because we made the choice $m = \lceil 1/(\delta\epsilon) \rceil$. $\square$

Algorithm 3: Coppersmith method.

```
Input: Polynomial $f(x)$ of degree $\delta$, modulus $N$ of unknown
       factorization and $0 < e < \frac{1}{7}$
Output: Set $R$, where $x_0 \in R$ whenever $f(x_0) = 0 \mod N$ for an
        $|x_0| \leq X$
$m = \lceil 1/(\delta\epsilon) \rceil$
$X = \frac{1}{2} N^{\frac{1}{\delta} - \epsilon}$
for $i \leftarrow 1$ to $m$ do
    for $j \leftarrow 0$ to $\delta - 1$ do
        $g_{i,j}(x) = x^j N^i f^{m-i}(x)$
Construct the lattice basis $B$, where the basis vectors of $B$ are the
coefficient vectors of $g_{i,j}(xX)$.
$\mathbf{v}$ = LLL($B$ , $c = 2$).get_column(0)
Construct $g(x)$ from $\mathbf{v}$.
Find the set $R$ of all roots of $g(x)$ over the integers using standart
methods. For every root $x_0 \in R$ check wether $\gcd(N, f(x_0)) \geq N$. If
it is not the case then remove $x_0$ from $R$.
```

3.3 Application

We can use the coppersmith method in order to attack RSA under certain conditions. We can use it to solve the Relaxed RSA problem where e is small and we have an approximation $\tilde{M}$ of M such that $M = \tilde{M} + x_0$ for some unknown part $|x_0| \leq N^{\frac{1}{e}}$. To solve this problem using Coppersmith method we define

$$f(x) = (\tilde{M} + x)^e - M^e \mod N.$$

And we can recover x_0 applying the coppersmith method to $f(x)$ as long as $x_0 \leq N^{\frac{1}{e}}$.

4 Implementation and examples

To implement and test the algorithms (LLL & Coppersmith), I used Sage. Sage is a free open-source mathematics software system licensed under the GPL. It combines the power of many existing open-source packages into a common Python-based interface. Its goal is to create a viable free open source alternative to Magma, Maple, Mathematica and Matlab. Website : `http://www.sagemath.org/`

4.1 LLL

As python is an interpreted language (it is not compiled), we don't expect to have good performance with our implementation of LLL. When a lot of computation are required, we prefer the native and optimized sage implementation that is much faster.

Code 1: Computes the Gram-Schmidt orthogonalization and then test if the matrix B is c-reduced or not.

```
def is_LLL_reduced(B, c = 2):
   n = B.nrows()
   m = B.ncols()
   U = matrix(RR, m, m)
   O = matrix(RR, n, m)
   for i in range(0, m) :
      U[i,i] = 1
      O.set_column(i, B.column(i))
      for j in range(0, i) :
         U[j,i] = (B.column(i)*O.column(j))/ \
                  (O.column(j)*O.column(j))
         O.set_column(i, O.column(i) - U[j,i]*O.column(j))

   for i in range(0, m-1) :
      if O.column(i)*O.column(i) > \
         c*O.column(i+1)*O.column(i+1) :
         return False
   return True
```

Code 2: LLL.

```
def reduce(i, B, U):
   j = i-1
   while j >= 0 :
      B.set_column(i, B.column(i) - \
         round(U[j,i])*B.column(j))
      U.set_column(i, U.column(i) - \
         round(U[j,i])*U.column(j))
      j = j - 1

def LLL(B, c = 2):
   n = B.nrows()
   m = B.ncols()
   U = matrix(RR, m, m)
   O = matrix(RR, n, m)
   for i in range(0, m) :
      U[i,i] = 1
      O.set_column(i, B.column(i))
      for j in range(0, i) :
         U[j,i] = (B.column(i)*O.column(j))/ \
                  (O.column(j)*O.column(j))
         O.set_column(i, O.column(i) - U[j,i]*O.column(j))
         reduce(i, B, U)

   i = 0
   while i < m-1 :
   if O.column(i)*O.column(i) <= \
      c*O.column(i+1)*O.column(i+1) :
      i = i + 1
   else :
      O.set_column(i+1, O.column(i+1) + \
         U[i,i+1]*O.column(i))
      U[i,i] = (B.column(i)*O.column(i+1))/ \
               (O.column(i+1)*O.column(i+1))
      U[i+1,i] = 1
      U[i, i+1] = 1
      U[i+1,i+1] = 0
      O.set_column(i, O.column(i)-U[i,i]*O.column(i+1))
      U.swap_columns(i,i+1)
      O.swap_columns(i,i+1)
      B.swap_columns(i,i+1)
      for k in range(i+2, m) :
         U[i,k] = (B.column(k)*O.column(i))/ \
                  (O.column(i)*O.column(i))
         U[i+1, k] = (B.column(k)*O.column(i+1))/ \
                     (O.column(i+1)*O.column(i+1))
      if abs(U[i,i+1]) > 0.5 : reduce(i+1, B, U)
         i = max(i-1,0)
   return B
```

In order to test if the implementation of LLL works properly, and to have an idea of the running time, we run it several times on random matrices of different sizes with the following code :

Code 3: LLL running time test

```
runtimes = []
for i in range(0, 39) :
   runtimes.append(0.0)
   for k in range(0, 10) :
      r = 0
      while r != i+2 :
         A = random_matrix(ZZ, i+2)
         r = A.rank()
      t = cputime()
      res = LLL(A)
      runtimes[i] = runtimes[i] + cputime(t)*0.1
      if is_LLL_reduced(res) == False :
         print("LLL FAILURE")
print(runtimes)
```

With the runtimes list we can plot the running time. Here is the result I got on my computer :

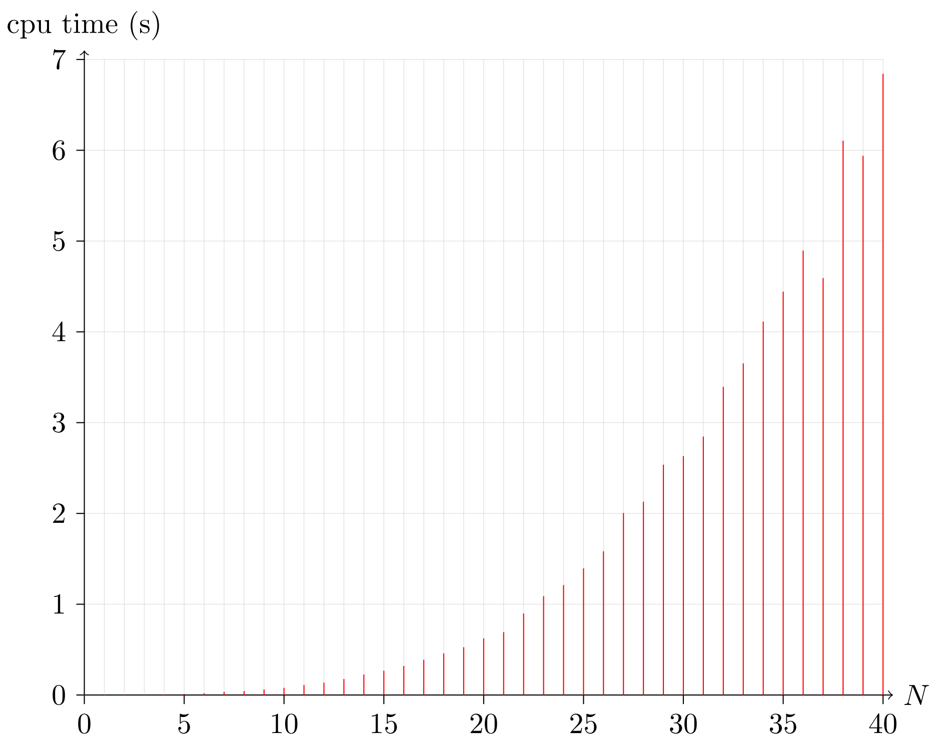


Figure 9: LLL execution time with random matrices of size NxN

We can clearly notice that the runtime is polynomial with respect to the matrix size. The runtime depends also on the basis vectors. For example it is possible that we generate a random matrix that is already reduced, then LLL has nothing to do. Here we compute the running time for each size N doing the average on 10 different matrices of size N. Because 10 is small, it explains why we can get that the running time for a 39x39 matrix is shorter than the one for a 38x38 matrix.

4.2 Coppersmith

Code 4: Coppersmith method

```
def coppersmith(f, N, epsilon = 0.1, fastLLL = False \
                                        , debug = False) :
   if epsilon > 1/7.0 or epsilon <= 0 :
       print("invalid epsilon")
       return None
   f.change_ring(Integers(N))
   delta = f.degree()
   m = ceil(1/delta/epsilon)
   R.<x> = ZZ[]
   #construction of the g[i,j](x)
   g = []
    for j in range(0, delta) :
       g.append([])
       for i in range(1, m+1) :
          g[j].append(x^j*N^(i)*f^(m-i))
   X = ceil(0.5*N^(1/delta - epsilon))
   if debug : print("X = " + str(X))
   size = m*delta
   #construct B from g[i,j](X*x)
   B = matrix(ZZ, size, size)
   compteur = 0
   for i in range(-m+1, 1) :
      for j in range(0, delta) :
         polylist = g[j][-i](X*x).list()
         vector = [0]*size
         vector[0:len(polylist)] = polylist
         vector.reverse()
         B.set_column(compteur, vector)
         compteur = compteur + 1
   if debug : show(B)
   if debug : print "LLL starts"
   coeffs = []
   #computes a small combination of g[i,x](X*x) with LLL
   if fastLLL : #use native sage implementation
      coeffs = B.transpose().LLL().transpose().\
                                column(0).list()
   else  :  #use our python implementation
      coeffs = LLL(B).column(0).list()
   coeffs.reverse()
   #construct g(x)
   g = 0*x
   for i in range(0, size) :
      g = g +  Integer(coeffs[i]/X^i) * x^i
   #get the roots of g(x) over the integers
   roots = g.roots(multiplicities=False)
   result = []
   #test if the roots x_i respect f(x_i) = 0 mod N
   for i in range(0, len(roots)) :
      if gcd(N, f(roots[i])) >= N :
         result.append(roots[i])
   return result
```

Code 5: Coppersmith method example

```
R.<x> = ZZ[]
f = (x-1)*(x-2)*(x-3)*(x-4)*(x-40)
print coppersmith(f, 10000)
```

In this example $X = 2$ and the Coppersmith method outputs the list [2,1].

4.3 RSA attack

To test an attack on RSA using the Coppersmith method, we first need to create RSA keys.

Code 6: RSA key creation : Computes d from p, q, and e.

```
def generate_d(p, q, e) :
   if not is_prime(p) :
      print "p is not prime"
      return None
      if not is_prime(q) :
         print "q is not prime"
         return None
   euler = (p-1)*(q-1)
   if gcd(e, euler) != 1 :
      print "e is not coprime with (p-1)(q-1)"
      return None
   return inverse_mod(e, euler)
```

Code 7: Simple example of an attack

```
p = 17
q = 37
n = p*q
e = 7
d = generate_d(p,q,e)
M = 21
C = power_mod(M, e, n)
MA = 20 #Approximation of M
R.<x> = ZZ[]
f = (MA + x)^e - C
print coppersmith(f, p*q)
```

In this example $X = 1$ and the coppersmith method outputs [1]. It follows that we can recover the message M : $M = MA + 1$

Code 8: Example of an attack

```
p = 955769
q = 650413
```

```
n = p*q
e = 5
d = generate_d(p,q, e)
M = 423909
C = power_mod(M, e, n)
MA = 423919 #Approximation of M
R.<x> = ZZ[]
f = (MA + x)^e - C
print coppersmith(f, p*q, 0.1,True)
```

In this example $X = 8$ and the method coppersmith method outputs [-10]. Notice that since $8 < 10$, it could have failed. In this case we use the LLL implementation of sage, because the coefficients of the $g[i, j](X * x)$ vectors seem to be too large for our naive LLL implementation.

5 Acknowledgements

I thank Bertrand Meyer for his good advice and helpful assistance.

6 Bibliography

References

[1] David C. LAY. *Linear Algebra and its applications.* Pearson Education, Inc, publishing as Addison Wesley Higher Education, 2003.

[2] Hendrik W. Lenstra, Jr. *Lattices.* In Algorithmic number theory: lattices, number fields, curves and cryptography Math. Sci. Res. Inst. Publ. 127-181 Cambridge Univ. Press Cambridge, 2008.

[3] Alexander May. *Using LLL-Reduction for Solving RSA and Factorization Problems.* Department of Computer Science TU Darmstadt.