The LLL Algorithm

1 Motivation

The rows of the following matrices form bases for lattices in $\mathbb{R}^3$:

$$X = \begin{bmatrix} -168 & 602 & 58 \\ 157 & -564 & -57 \\ 594 & -2134 & -219 \end{bmatrix}, \quad Y = \begin{bmatrix} -6 & 6 & -4 \\ 9 & 4 & 1 \\ -1 & 8 & 6 \end{bmatrix}.$$

The rows of X and the rows of Y actually span the *same* lattice. Intuitively, the rows of X seem to be a "worse" basis for L than those of Y. Here we make precise the notion of a "nice" basis and introduce a polynomial-time algorithm that transforms a "bad" basis into a "good" one.

2 Basis Reduction and the LLL Algorithm

A basis is "nice" if its vectors are short and orthogonal to one another. The Gram-Schmidt process transforms a given basis into an orthogonal basis, but when working in a lattice L, this Gram-Schmidt basis need not live in L.

Definition 2.1. Let $x_1, \ldots, x_n$ be an ordered basis for a lattice L in $\mathbb{R}^n$, and let $x_1^*, \ldots, x_n^*$ be its Gram-Schmidt orthogonalization (GSO). Write $X = MX^*$ where X (respectively X^*) is the matrix with x_i (respectively x_i^*) as row i and $M = (\mu_{ij})$ is the matrix of GSO coefficients. Let α be a real number with $\frac{1}{4} < \alpha < 1$. We say that the basis $x_1, \ldots, x_n$ is **α-reduced** if it satisfies

1. $|\mu_{ij}| \leq \frac{1}{2}$ for all $1 \leq j < i \leq n$,
2. $|x_i^* + \mu_{i,i-1}x_{i-1}^*|^2 \geq \alpha |x_{i-1}^*|^2$ for $2 \leq i \leq n$.

Condition (1) says that the i-th basis vector is "almost orthogonal" to the span of the previous $i-1$ vectors. The vector $x_i^* + \mu_{i,i-1}x_{i-1}^*$ is the vector one obtains after swapping vectors x_i and x_{i-1} and then computing the $(i-1)$-st vector of the GSO. Condition (2) then says that this new GSO vector, while potentially shorter than x_{i-1}^*, isn't "too much" shorter.

3 An Application: Small Roots of Polynomials mod M

Say we want to find a root x_0 of $f(x) \equiv 0 \pmod{M}$ (e.g. where $f(x) = x^e$ and M is an RSA modulus). Our plan is to use the LLL algorithm to construct an *integer* polynomial with small coefficients that also has x_0 as a root. Since computing roots of polynomials over $\mathbb{Q}$ is easy, this gives us a solution to $f(x) \equiv 0 \pmod{M}$. Importantly, we do not need to know the factorization of M.

Algorithm 1 The Original LLL Algorithm

Input: A basis $x_1, \dots, x_n$ of the lattice $L \subset \mathbb{R}^n$ and a reduction parameter $\alpha \in (\frac{1}{4}, 1)$.

Output: An α-reduced basis $y_1, \dots, y_n$ of the lattice L.

```
1: procedure REDUCE($k, \ell$)          ▷ makes $y_k$ "almost" orthogonal to $y_\ell$ then updates GSO
2:     if $|\mu_{k\ell}| > \frac{1}{2}$ then
3:         Set $y_k \leftarrow y_k - \lceil \mu_{k\ell} \rfloor y_\ell$.          ▷ $\lceil \mu_{k\ell} \rfloor$ is the closest integer to $\mu_{k\ell}$
4:         for $j = 1, 2, \dots, \ell - 1$ do
5:             Set $\mu_{kj} \leftarrow \mu_{kj} - \lceil \mu_{k\ell} \rfloor \mu_{\ell j}$.
6:         Set $\mu_{k\ell} \leftarrow \mu_{k\ell} - \lceil \mu_{k\ell} \rfloor$.
7: procedure EXCHANGE($k$)          ▷ Exchange $y_{k-1}$ and $y_k$ then update the GSO
8:     Set $z \leftarrow y_{k-1}$, $y_{k-1} \leftarrow y_k$, $y_k \leftarrow z$.
9:     Set $\nu \leftarrow \mu_{k,k-1}$. Set $\delta \leftarrow \gamma_k^* + \nu^2 \gamma_{k-1}^*$.
10:    Set $\mu_{k,k-1} \leftarrow \nu \gamma_{k-1}^* / \delta$. Set $\gamma_k^* \leftarrow \gamma_k^* \gamma_{k-1}^* / \delta$. Set $\gamma_{k-1}^* \leftarrow \delta$.
11:    for $j = 1, 2, \dots, k - 2$ do
12:        Set $t \leftarrow \mu_{k-1,j}$, $\mu_{k-1,j} \leftarrow \mu_{kj}$, $\mu_{kj} \leftarrow t$.
13:    for $i = k + 1, \dots, n$ do
14:        Set $\xi \leftarrow \mu_{ik}$. Set $\mu_{ik} \leftarrow \mu_{i,k-1} - \nu \mu_{ik}$.
15:        Set $\mu_{i,k-1} \leftarrow \mu_{k,k-1} \mu_{ik} + \xi$.
16: procedure MAIN
17:    for $i = 1, 2, \dots, n$ do          ▷ Initialize the vectors $y_1, \dots, y_n$
18:        Set $y_i \leftarrow x_i$.
19:    for $i = 1, 2, \dots, n$ do          ▷ Compute the GSO of the vectors $y_1, \dots, y_n$
20:        Set $y_i^* \leftarrow y_i$.
21:        for $j = 1, 2, \dots, i - 1$ do
22:            $\mu_{ij} \leftarrow (y_i \cdot y_j^*) / \gamma_j^*$ and $y_i^* \leftarrow y_i^* - \mu_{ij} y_j^*$.
23:        Set $\gamma_i^* \leftarrow y_i^* \cdot y_i^*$.
24:    Set $k \leftarrow 2$.
25:    while $k \leq n$ do
26:        Call REDUCE($k, k - 1$).
27:        if $\gamma_k^* \geq (\alpha - \mu_{k,k-1}^2) \gamma_{k-1}^*$ then
28:            for $\ell = k - 2, \dots, 2, 1$ do
29:                Call REDUCE($k, \ell$).
30:            Set $k \leftarrow k + 1$.
31:        else
32:            Call EXCHANGE($k$).
33:            if $k > 2$ then
34:                Set $k \leftarrow k - 1$.
```