

CSE206A: Lattice Algorithms and Applications

Lecture 1

Point Lattices: Introduction

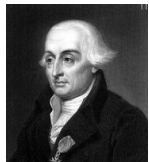
UCSD – Fall 2025

Daniele Micciancio



Point Lattices:

- simple mathematical objects ... with many applications in Math and CS
- Studied by mathematicians for centuries



...



Lattices in Computer Science:

- Algorithms: Integer Programming, Cryptanalysis, Error Correcting Codes, ...
- Complexity: NP-hardness, Average-case complexity, interactive proof systems, ...
- Cryptography: “Post-quantum” Cryptography, Fully Homomorphic Encryption, ...

- A lattice is the set of all integer linear combinations of n linearly independent vectors:

$\mathbf{b}_1, \dots, \mathbf{b}_n \in \mathbb{R}^d$ linearly independent

$$\mathcal{L}(\mathbf{b}_1, \dots, \mathbf{b}_n) = \{\sum_i \mathbf{b}_i x_i : x_i \in \mathbb{Z}\} \subset \mathbb{R}^d$$

- Basis matrix $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_n]$

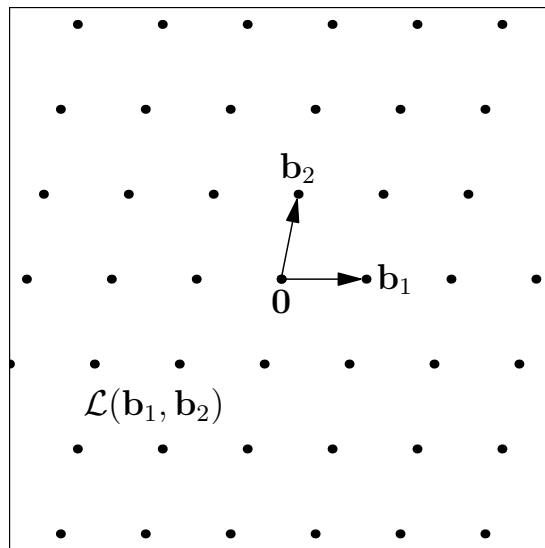
1. $\mathbf{b}_i$: column vectors

2. linearly independent $\implies n \leq d$

- Full dimensional lattices ($n = d$)

1. $\mathbf{B}$ is a basis for $\mathbb{R}^d$ as a vector space

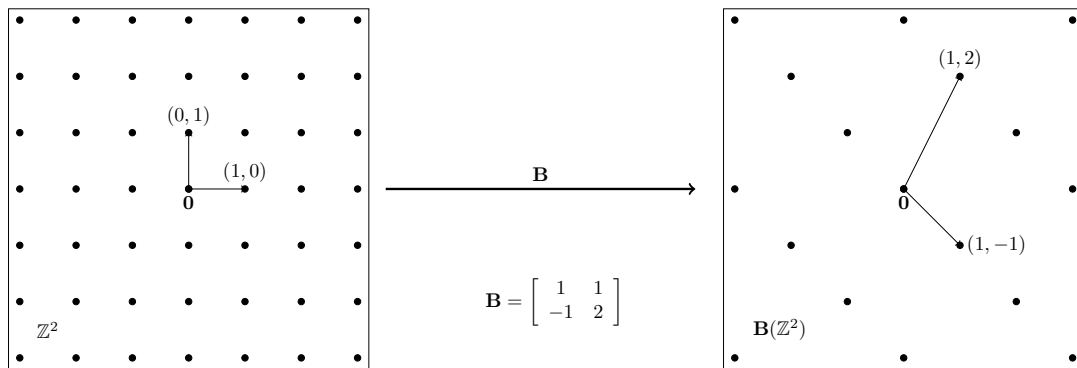
2. Not every basis of $\mathbb{R}^d$ is a basis for $\mathcal{L}$



- Lattices are linear: the sum (and difference) of lattice points is still in the lattice

$$\sum_i b_i x_i + \sum_i b_i y_i = \sum_i b_i (x_i + y_i)$$

- Lattice basis $B = [b_1, \dots, b_n]$ as a linear transformation mapping $x \in \mathbb{Z}^n$ to Bx



Example: $\mathbb{Z}^2 \subset \mathbb{R}^2 = \text{span}(\mathbb{Z}^2)$

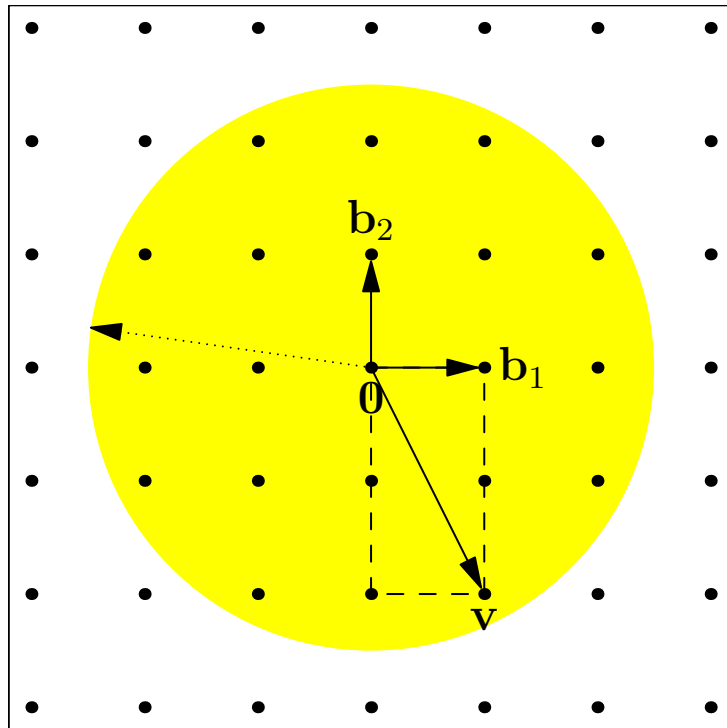
- The lattice $\mathbb{Z}^2$ is a “discrete” sets of points
- It lives inside the vector space: $\mathbb{R}^2 = \text{span}(\mathbb{Z}^2)$

Interplay between geometry and algebra:

- $\mathbb{Z}^2$ inherits the geometry of $\mathbb{R}^2$
- $\mathbb{Z}^2$ has a (discrete) group structure

Typical properties:

- Algebra: $v = b_1 - 2b_2$
- Geometry: $\|v\| < 2.5$



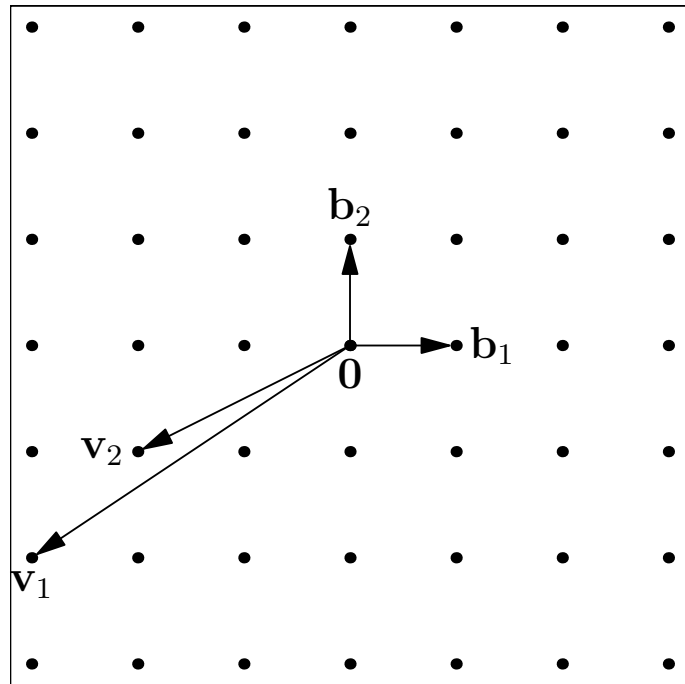
- The same lattice has many bases

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad V = \begin{bmatrix} -3 & -2 \\ -2 & -1 \end{bmatrix}$$

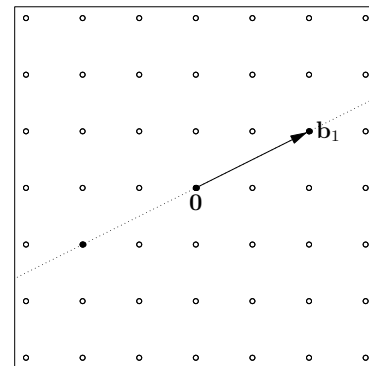
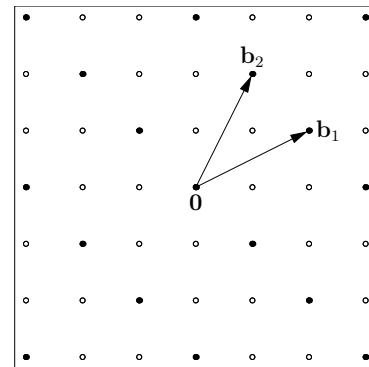
- Some bases are “better” than others

Question 1. What does it mean for a basis to be good/bad? Is there a best/worst basis?

Question 2. How can you tell if two bases generate the same lattice?



- Λ_1 is a sublattice of Λ_2 if it is
 1. contained in Λ_2 : $\Lambda_1 \subseteq \Lambda_2$, and
 2. closed under linear operations: $\mathbf{v} + \mathbf{u}, \mathbf{v} - \mathbf{u} \in \Lambda_1$ for all $\mathbf{v}, \mathbf{u} \in \Lambda_1$
- A sublattice is a lattice, i.e., it admits a basis
 - Intuitively obvious, but how do you prove it?
- Λ_1 is a
 - *full rank* sublattice if $\dim(\Lambda_1) = \dim(\Lambda_2)$
 - *saturated* sublattice if $\Lambda_1 = \Lambda_2 \cap \text{span}(\Lambda_1)$



Question 3. Can a sublattice be both full rank and saturated?

Theorem 1. $\mathcal{L}(C)$ is a sublattice of $\mathcal{L}(B)$ iff $C = B X$ for some integer matrix $X \in \mathbb{Z}^{n \times n}$

Proof. Write each $c_i = B x_i$ as a combination of the basis vectors, and set $X = [x_1, \dots, x_n]$ $\square$

Theorem 2. $\mathcal{L}(B) = \mathcal{L}(C)$ iff $C = B U$ for some unimodular matrix $U \in \mathbb{Z}^{n \times n}$, $\det(U) = \pm 1$

Proof. We know $C = B X$ and $B = C Y$ for some $X, Y \in \mathbb{Z}^{n \times n}$. Then, we have

$$\begin{aligned} C &= B X = (C Y) X \\ I &= X Y \end{aligned}$$

So X is invertible, with integer inverse Y , and $\det(X) = \det(Y) = \pm 1$ $\square$

- Elementary operations that do not change the lattice generated by a basis
 1. Swap: $\mathbf{b}_i \leftrightarrow \mathbf{b}_j$
 2. Sign change: $\mathbf{b}_i \mapsto -\mathbf{b}_i$ (multiplication by $c \in \mathbb{Z}^* = \{\pm 1\}$)
 3. Addition: $\mathbf{b}_i \mapsto \mathbf{b}_i + c \mathbf{b}_j$ for some $j \neq i$ and $c \in \mathbb{Z}$
- Each operation can be described as multiplication by a unimodular $\mathbf{U}$
- Similar to elementary operations to modify a basis of a vector space, but cannot multiply $\mathbf{b}_i$ by arbitrary $c \in \mathbb{R}^*$.

Theorem 3. *Any unimodular matrix can be described as a sequence of elementary integer operations.*

- Normal form for (full rank) lattice basis

- A matrix $\mathbf{H} \in \mathbb{Z}^{n \times n}$ is in HNF if

1. $\mathbf{H}$ is upper triangular
2. diagonal elements are positive
3. off diagonal elements are reduced:

$$0 \leq h_{i,j} < h_{i,i}$$

$$\mathbf{H} = \begin{bmatrix} h_{1,1} & \cdots & \cdots & \cdots & \cdots & \cdots \\ & \ddots & & & & \\ & & h_{i,i} & \cdots & h_{i,j} & \cdots \\ & & & \ddots & & \\ & & & & \ddots & \\ & & & & & h_{n,n} \end{bmatrix}$$

Theorem 4. Any full rank integer lattice has a basis in HNF

Proof. Transform any $\mathbf{B} \in \mathbb{Z}^{n \times n}$ into HNF using elementary integer operations



- Recall Gaussian elimination process to transform a square matrix into diagonal form:
 1. Forward elimination: make matrix triangular using elementary real operations
 2. “Back substitution”: make matrix diagonal
- Usually applied to the rows of a matrix, but can equivalently apply to columns
- Problem:
 - Both forward and backward steps make use of arbitrary (nonzero) multipliers
 - Cannot be applied to lattice bases because they are not integer operations
- Solution:
 - Forward elimination steps $\implies$ Euclid's GCD algorithm
 - Backward elimination steps $\implies$ Modular reduction

- Start from an arbitrary (square, integer) matrix $A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$
- Run Euclid's algorithm $g_{n2} = \gcd(a_{n1}, a_{n2})$, applying the operations to the whole columns (1,2), and repeat for columns (2,3), (3,4), ..., (n-1, n), to get

$$\Rightarrow \begin{bmatrix} a'_{11} & a'_{12} & \dots & a_{1n} \\ \vdots & \vdots & & \vdots \\ 0 & g_{n2} & \dots & a_{nn} \end{bmatrix} \Rightarrow \begin{bmatrix} a'_{11} & a''_{12} & a'_{13} & \dots & a_{1n} \\ \vdots & \vdots & & & \vdots \\ 0 & 0 & g_{n3} & \dots & a_{nn} \end{bmatrix} \Rightarrow \begin{bmatrix} A' & a' \\ 0 & g_{nn} \end{bmatrix}$$

- Repeat recursively on A' to get a triangular matrix $\begin{bmatrix} g_{11} & \dots & a'_{1n} \\ 0 & \ddots & \vdots \\ 0 & 0 & g_{nn} \end{bmatrix}$
- Use diagonal elements g_{ii} to reduce a'_{ij} ($j > i$) modulo g_{ii} , in order $i = n, \dots, 1$

- Any basis of a lattice can be obtained using elementary integer operations
 - Let $\mathbf{B}$ and $\mathbf{C} = \mathbf{B}\mathbf{U}$ be lattice bases, with $\det(\mathbf{U}) = \pm 1$
 - Compute $\mathbf{H} = \text{HNF}(\mathbf{U}) = \mathbf{U}\mathbf{P}$ using integer ops $\mathbf{P}$
 - $\det(\mathbf{H}) = \det(\mathbf{U}) = \pm 1$ and $\mathbf{H}$ in HNF $\implies \mathbf{H} = \mathbf{I}$
 - $\mathbf{C} \xRightarrow{\mathbf{P}} \mathbf{B}$ because $\mathbf{C}\mathbf{P} = (\mathbf{B}\mathbf{U})\mathbf{P} = \mathbf{B}\mathbf{H} = \mathbf{B}$
- App 1: Lattice “reduction” algorithms can focus on elementary integer operations!
- App 2: Compute a lattice basis from a (linearly dependent) generating set
 $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{C})$ if and only if $\text{HNF}(\mathbf{B}) = \text{HNF}(\mathbf{C})$
- App 3: compare two lattice bases by checking if $\text{HNF}(\mathbf{B}) = \text{HNF}(\mathbf{C})$

- For any two bases $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{C})$ there is a sequence of operations $\mathbf{B} \xRightarrow{P} \mathbf{C}$
 1. Is the length of the sequence of operations polynomial in the input size?
 2. Are all integer coefficients c and intermediate values polynomial size?
- HNF algorithm performs $O(n^3)$ gcd computations
 1. ... however, in a straightforward implementation integer sizes can grow exponentially!
 2. Still, with some care, HNF can be computed in polynomial time (and linear space!)

See [M., Warinschi, 2001] for a polynomial time / linear space algorithm

Question 4. Is HNF a “good” basis?

- Consider “geometric” lattice problems: given basis B , compute some property of $\mathcal{L}(B)$
- HNF can be computed in polynomial time from any other basis $B \Rightarrow H$
- Any lattice algorithm $\mathcal{A}$ that works on input H can be efficiently converted into

$$\mathcal{A}'(B) = \mathcal{A}(\text{HNF}(B))$$

that works on input any lattice basis

- If a lattice problem is hard at all, then it is hard when input basis is in HNF
- App 4: “Improving Lattice Based Cryptosystems using the HNF” (M., CaLC 2001)
 - Pick a “good” random lattice basis B as a secret key, and set public key to $H = \text{HNF}(B)$
 - Implicitly done in many practical cryptosystems, like NTRU, and optimized LWE