The lattice A_n^* and applications to synchronization

Robby McKilliam

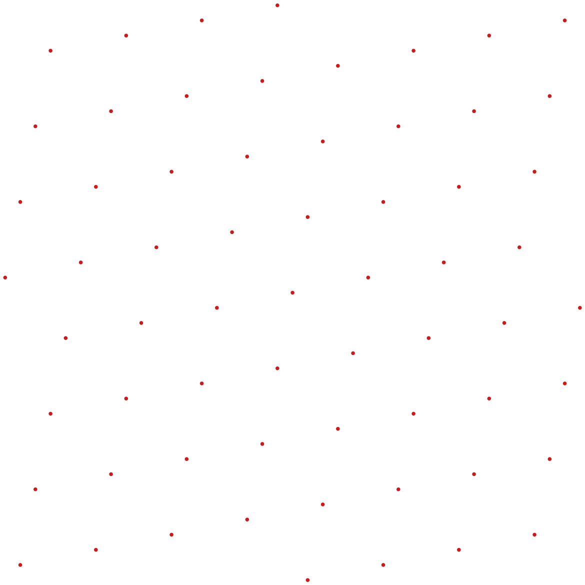


Figure : A lattice in 2 dimensions

Lattices

- A **lattice**, L, is a set of points in $\mathbb{R}^m$ such that

$$L = \{\mathbf{x} \in \mathbb{R}^m \mid \mathbf{x} = \mathbf{B}\mathbf{w}, \mathbf{w} \in \mathbb{Z}^n\}$$

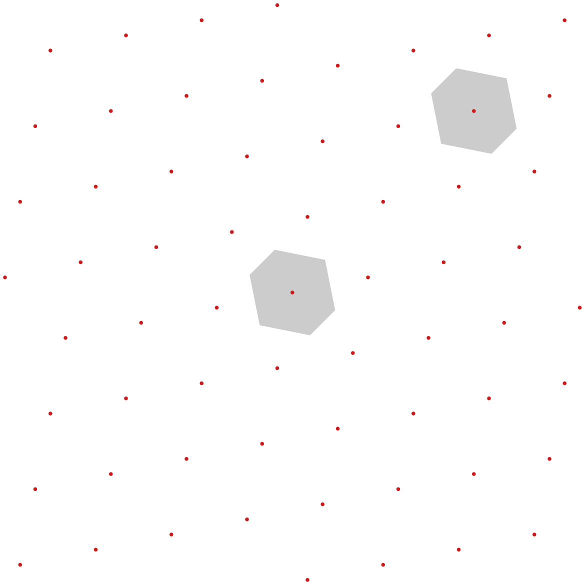
- The matrix $\mathbf{B}$ is an $m \times n$ matrix called the **generator matrix**.
- We write this more succinctly as

$$L = \mathbf{B}\mathbb{Z}^n$$

- $\mathbb{Z}^n$ is called the **integer lattice**.

The Voronoi Cell

- The **Voronoi cell** is the region that is closer to the origin than any other lattice point.
- The Voronoi cell is a convex polytope (an n-dimensional polygon).
- Translating the Voronoi cell by a lattice point $\mathbf{x}$ gives the region closer to $\mathbf{x}$ than any other lattice point.



The nearest lattice point problem

Definition

Given $\mathbf{y} \in \mathbb{R}^n$ and some lattice L whose lattice points lie in $\mathbb{R}^n$, find the lattice point $\mathbf{x} \in L$ that is closest to $\mathbf{y}$.

- The lattice point $\mathbf{x}$ is nearest to $\mathbf{y}$ if and only if $\mathbf{y}$ is inside the Voronoi cell translated about $\mathbf{x}$.
- Many applications including:
  - coding
  - quantisation
  - communications systems using multiple antennas
  - public key cryptography
  - frequency and polynomial phase estimation

The Nearest Point Problem

- NP-complete for general lattices (computationally very hard!).
- Easier for specific lattices.
- We will describe a very fast algorithm for the famous lattice A_n^*.
- Requires only $O(n)$ operations!
- Use this algorithm for **delay estimation**.

The lattice A_n^*

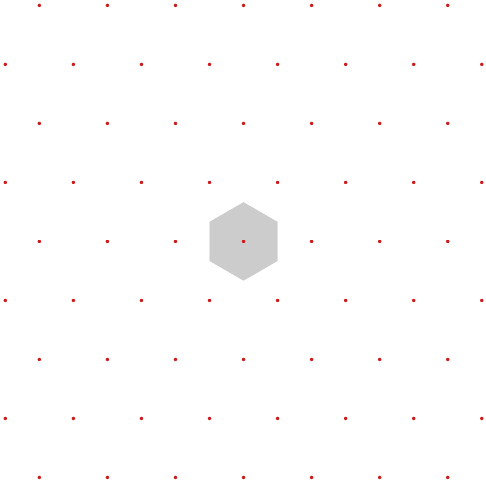


Figure : In 2 dimensions A_2^* is the **hexagonal lattice**

The lattice A_n^*

- More generally A_n^* is the projection of $\mathbb{Z}^{n+1}$ into the space orthogonal to
$$\mathbf{1} = [1, 1, 1, \ldots, 1]'$$
- The matrix to perform this projection is
$$\mathbf{Q} = \mathbf{I} - \frac{\mathbf{1}\mathbf{1}'}{n+1}$$
where $\mathbf{I}$ is the $(n+1) \times (n+1)$ identity matrix.
- The points in A_n^* are
$$A_n^* = \mathbf{Q}\mathbb{Z}^{n+1}$$
- Given $\mathbf{y} \in \mathbb{R}^{n+1}$ our aim is to find the $\mathbf{x} \in A_n^*$ closest to $\mathbf{y}$.

The nearest point in A_n^*

Definition

Define the function $\mathbf{f} : \mathbb{R} \mapsto \mathbb{Z}^{n+1}$

$$\mathbf{f}(\lambda) = \lceil \mathbf{y} + \lambda \mathbf{1} \rfloor$$

where $\lceil \cdot \rfloor$ rounds each element to its nearest integer

Lemma

The nearest point in A_n^* to $\mathbf{y} \in \mathbb{R}^{n+1}$ is one of

$$\mathbf{Q}\mathbf{f}\left(\frac{i-1}{n+1}\right)$$

where $i = 1, 2, \ldots, n+1$.

The nearest point in A_n^*

- So there are only $n + 1$ candidates for the nearest point.
- A naïve approach to finding the nearest point would be to calculate each
$$\mathbf{Qf}\left(\frac{i-1}{n+1}\right)$$
directly and return the one closest to $\mathbf{y}$.
- This would require $O(n^2)$ arithmetic operations.

The nearest point in A_n^*

- The first candidate is

$$\mathbf{f}(0) = \lceil \mathbf{y} \rfloor$$

- Construct the set of indices

$$S_1 = \left\{ j \mid \lceil y_j \rfloor - y_j \in \left(0, \frac{1}{n+1} \right] \right\}$$

- Then

$$\mathbf{f}\left(\frac{1}{n+1} \right) = \lceil \mathbf{y} \rfloor + \sum_{j \in S_1} \mathbf{e}_j$$

where $\mathbf{e}_j$ is a vector of 0's except the jth element which is 1.

The nearest point in A_n^*

- Continuing this approach we can construct $n+1$ sets
$$S_i = \left\{ j \mid \lceil y_j \rceil - y_j \in \left(\frac{i-1}{n+1}, \frac{i}{n+1} \right] \right\}$$
for $i = 1, 2, \ldots, n+1$.
- Then
$$\mathbf{f}\left(\frac{i}{n+1}\right) = \mathbf{f}\left(\frac{i-1}{n+1}\right) + \sum_{j \in S_i} \mathbf{e}_j$$

The nearest point in A_n^*

- All the S_i can be computed in linear-time using a **bucket sort**.
- So we can find all of the candidate nearest points in linear time.
- It remains to show that we can efficiently compute the distance between each candidate and $\mathbf{y}$.
- We require to compute

$$d_{i-1} = \left\| \mathbf{y} - \mathbf{Qf}\left(\frac{i-1}{n+1}\right) \right\|^2$$

for $i = 1, 2, \cdots, n+1$.

The nearest point in A_n^*

- We find that the d_{i-1} can be computed by the following recursion

$$\alpha_i = \alpha_{i-1} - |S_i|$$
$$\beta_i = \beta_{i-1} + |S_i| - 2\sum_{j \in S_i} y_j - \lceil y_j \rfloor$$
$$d_i = \beta_i - \frac{\alpha_i^2}{n+1} + t$$

- Where α_0, β_0 and t can all be computed in linear time.

The nearest point in A_n^*

- So all of the
$$\mathbf{f}\left(\frac{i-1}{n+1}\right)$$
and the d_{i-1} can be computed in linear time.
- The algorithm then returns
$$\mathbf{Q}\mathbf{f}\left(\frac{j}{n+1}\right)$$
where d_j is the minimum of the d_{i-1}.
- Multiplication by $\mathbf{Q}$ can be performed in linear time

Table : Computation time in seconds for 10^5 trials

Algorithm	n=20	n=50	n=100	n=500
CS $O(n^2)$	13.77	80.17	327.01	$> 10^4$
IVLC $O(n \log n)$	2.72	5.73	10.99	58.98
MCQ $O(n \log n)$	2.28	4.60	8.61	32.73
$O(n)$	1.83	3.33	5.86	25.91

Delay estimation

- We will now apply this nearest point algorithm to a delay estimation problem.
- Applications are to synchronisation for communications devices and also radar.
- Consider sampling the time of arrival of N periodic events with known period T and unknown delay μ_0
- Assume that the sampling process is both **noisy** and **sparse**.

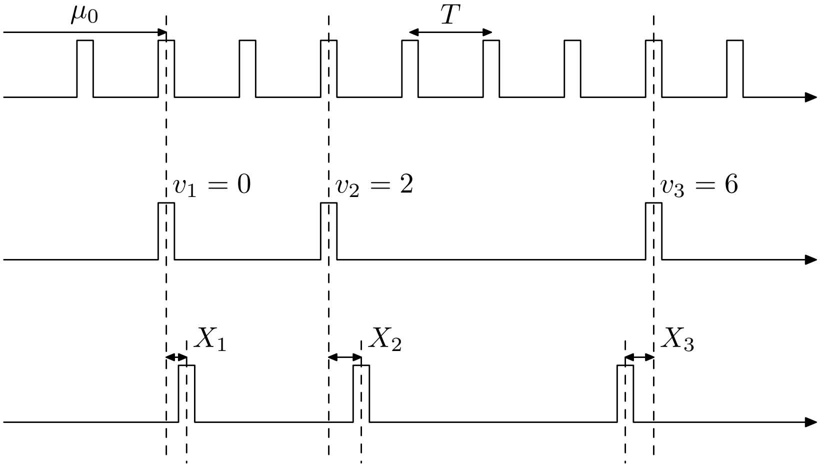


Figure : Delay estimation from incomplete data.

Delay estimation

- The model is
$$Y_n = Tv_n + \mu_0 + X_n$$
- T is a known period
- μ_0 is the delay to estimate
- X_n is noise
- v_n are unknown integers representing the pulses that are received
- Assume N observations so $n = 1, \dots, N$

Delay estimation

- Setting $T = 1$ for simplicity

$$Y_n = v_n + \mu_0 + X_n$$

- If the v_n were known

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^{N} (Y_n - v_n)$$

- If the X_n are zero mean i.i.d. with variance σ_g^2 then the estimator has variance

$$\operatorname{var}(\hat{\mu} - \mu_0) = \frac{\sigma_g^2}{N}.$$

- More interested in the case when v_n are unknown.

Delay estimation

- Take a least squares approach

$$\hat{\mu} = \arg\min_{\mu} \min_{w_n \in \mathbb{Z}} \sum_{n=1}^{N} (Y_n - \mu - w_n)^2$$

- In vector form

$$\hat{\mu} = \arg\min_{\mu} \min_{\mathbf{w} \in \mathbb{Z}^N} \|\mathbf{y} - \mu\mathbf{1} - \mathbf{w}\|^2$$

- Fix w_n and minimising with respect to μ

$$\hat{\mu} = \frac{\mathbf{1}'(\mathbf{y} - \mathbf{w})}{N}$$

- Substituting this gives

$$\hat{\mathbf{v}} = \arg\min_{\mathbf{w} \in \mathbb{Z}^N} \|\mathbf{Q}\mathbf{y} - \mathbf{Q}\mathbf{w}\|^2$$

- Where $\mathbf{Q}$ is the generator matrix for A^*_{N-1}

Asymptotic properties

Theorem

If $X_1, X_2, \ldots, X_N$ are i.i.d. random variables and the fractional parts $X_n - \lceil X_n \rfloor$ have zero **unwrapped mean**, *variance σ^2 and pdf f. Then:*

1. **(Strong consistency)** *$\hat{\mu}$ converges almost surely to μ_0 as $N \to \infty$.*
2. **(Asymptotic normality)** *The distribution of*

$$\sqrt{N}(\hat{\mu} - \mu_0)$$

approaches the normal with zero mean and variance

$$\frac{\sigma^2}{\big(1 - f(-1/2)\big)^2}.$$

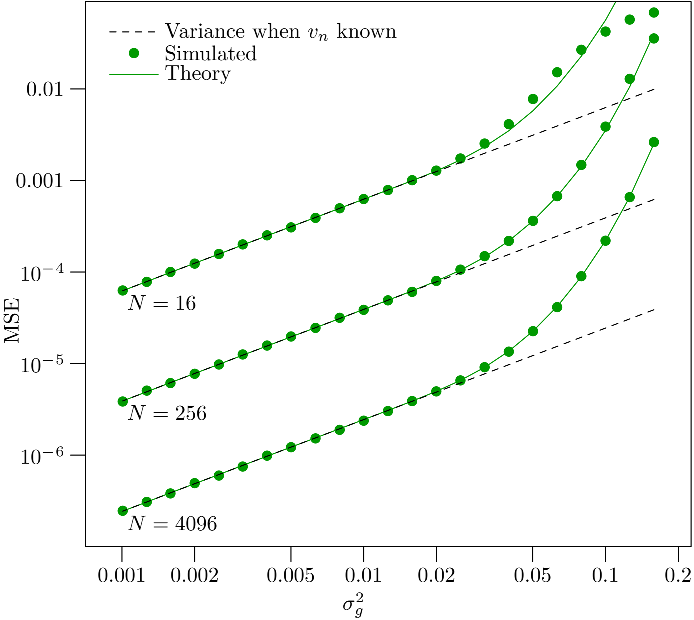


Figure : Performance when X_n are normal with variance σ_g^2

- This delay estimation problem can also be described using **circular statistics**.
- Equivalent to estimating the **mean direction** of a circular random variable.
- There is another simple estimator called the **sample circular mean**.
- Can also be computed in linear time.
- Statistical properties of this estimator are different.
- The problem is substantially harder if the period is also unknown, but we are working on it!