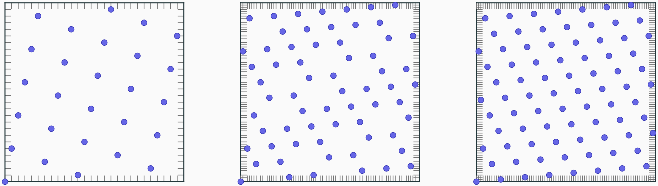
Lattice point sets and applications (part I)

Dirk Nuyens — NUMA, KU Leuven, Belgium



Workshop and Summer School on Applied Analysis 2023
TU Chemnitz
Chemnitz, Germany
September 2023

The plan for today

The plan for today

- A light introduction to “lattice points” & “lattice rules”.
- Usage for numerical integration of “periodic” functions.
- Analysis of the error.
- Some words on function spaces and the worst-case error.
- Some Julia code to demonstrate things. . .

Lattice “points” or lattice “rules”?

Lattice rule = equal weight cubature using lattice points

For $f \in \mathcal{H}_\alpha$ approximate the d-dimensional integral

$$I(f) := \int_{[0,1]^d} f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$$

by an n-point lattice rule with generating vector $\boldsymbol{z} \in \mathbb{Z}_n^d$

$$Q_{n,\boldsymbol{z}}(f) := \frac{1}{n} \sum_{k \in \mathbb{Z}_n} f\left(\frac{\boldsymbol{z}k \bmod n}{n}\right).$$

Worst-case error for $f \in \mathcal{H}_\alpha$ for a given algorithm Q_n (e.g. $Q_{n,\boldsymbol{z}}$):

$$e^{\mathrm{det}}(Q_n, \mathcal{H}_\alpha) := \sup_{\substack{f \in \mathcal{H}_\alpha \\ \|f\|_\alpha \leq 1}} |I(f) - Q_n(f)|.$$

Lattice rule = equal weight quadrature using lattice points

For $f \in \mathcal{H}_\alpha$ approximate the d-dimensional integral

$$I(f) := \int_{[0,1]^d} f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$$

by an n-point lattice rule with generating vector $\boldsymbol{z} \in \mathbb{Z}_n^d$

$$Q_{n,\boldsymbol{z}}(f) := \frac{1}{n} \sum_{k \in \mathbb{Z}_n} f\left(\frac{\boldsymbol{z} k \bmod n}{n} \right).$$

Worst-case error for $f \in \mathcal{H}_\alpha$ for a given algorithm Q_n (e.g. $Q_{n,\boldsymbol{z}}$):

$$e^{\mathrm{det}}(Q_n, \mathcal{H}_\alpha) := \sup_{\substack{f \in \mathcal{H}_\alpha \\ \|f\|_\alpha \le 1}} |I(f) - Q_n(f)|.$$

Lattice rule = equal weight quadrature using lattice points

For $f \in \mathcal{H}_\alpha$ approximate the d-dimensional integral

$$I(f) := \int_{[0,1]^d} f(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}$$

by an n-point lattice rule with generating vector $\boldsymbol{z} \in \mathbb{Z}_n^d$

$$Q_{n,\boldsymbol{z}}(f) := \frac{1}{n} \sum_{k \in \mathbb{Z}_n} f\left(\frac{\boldsymbol{z}k \bmod n}{n}\right).$$

Worst-case error for $f \in \mathcal{H}_\alpha$ for a given algorithm Q_n (e.g. $Q_{n,\boldsymbol{z}}$):

$$e^{\det}(Q_n, \mathcal{H}_\alpha) := \sup_{\substack{f \in \mathcal{H}_\alpha \\ \|f\|_\alpha \le 1}} |I(f) - Q_n(f)|.$$

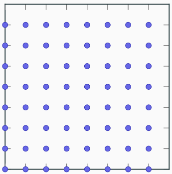
$\rightsquigarrow$ For good lattice rule $Q_{n,\boldsymbol{z}}$ converges like $n^{-\alpha} \|f\|_\alpha$.
Optimal. Bakhvalov. Matching upper and lower bounds (mod logs).

“Monte Carlo type” methods: $\frac{1}{n}\sum_{k=1}^{n} f(\boldsymbol{x}_k)$

What kind of cubature/quadrature method to use for d large?

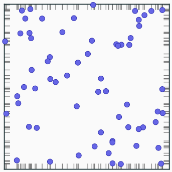
- A product of classical quadrature rules? (Product of weights!)
  $\rightarrow n = m^d \Rightarrow$ The curse “by construction”!
- The plain Monte Carlo method: $\boldsymbol{x}_k \sim U[0,1)^d$.
  $\rightarrow$ Free to choose n.
- Quasi-Monte Carlo methods: using some algebraic structure.
  $\rightarrow$ Free to choose n.

grid



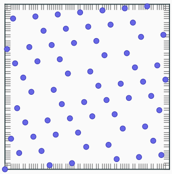
$n = m^d$
error $= O(n^{-r/d})$

MC



n free
std $= O(n^{-1/2})$

QMC



n free
error $= O(n^{-1})$, ...

Function space

Korobov space of dominating mixed smoothness $\alpha > 0$:

$$\mathcal{H}_\alpha := \left\{ f \in L_2([0,1]^d) : \|f\|_\alpha^2 := \sum_{\boldsymbol{h} \in \mathbb{Z}^d} r_\alpha^2(\boldsymbol{h}) \, |\hat{f}(\boldsymbol{h})|^2 < \infty \right\},$$

with

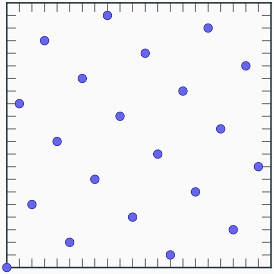
$$r_\alpha(\boldsymbol{h}) := \gamma_{\mathrm{supp}(\boldsymbol{h})}^{-1} \prod_{j \in \mathrm{supp}(\boldsymbol{h})} |h_j|^\alpha.$$

Weighted spaces: Sloan & Woźniakowski (2001), Novak & Woźniakowski (2008, 2010, 2012), . . .

More on norms tomorrow. . .

Example of a good lattice rule

Eg: $n = 21$ and $\boldsymbol{z} = (1, 13)$: Fibonacci rule: $n = F_k$, $\boldsymbol{z} = (1, F_{k-1})$.



Only $d = 2$, $d \geq 2$: Constructive methods for deterministic error:

Fast component-by-component (Nuyens & Cools 2006, ...)

$\rightarrow$ Fixed vector $\boldsymbol{z}$ for a given n.

(Or sequence of $n = p^m$, Cools, Kuo & Nuyens 2006).

What can we do with lattice points???

- INT: The integration problem: approximate

$$I(f) := \int_{[0,1]^d} f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}.$$

- APP: The function approximation problem: find an approximation for an $f \in \mathcal{H}$ minimizing some norm.
- Collocation methods.
- Least-squares methods.
- ...

Note: If you are familiar with information based complexity (IBC): Since we use the lattice points as sample points this is the setting of standard information, sometimes called Λ^{std}.

Lots of work: Korobov, Sloan, Temlyakov, Niederreiter, a lot of people in this audience...

First demo

Julia break...

- A (rank 1) lattice point generator (as in `Generator`).
- The "order" of the points.
- Rotated grids or grids?
- Use for numerical integration.
- Good and bad rules?

Error for an integrand; Worst-case error for function space

Error for an integrand using lattice rule approximation

For $f \in \mathcal{H}_\alpha$, with $\alpha > 1/2$, or actually,
for f with abs. conv. Fourier series, "Wiener algebra",

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{h} \in \mathbb{Z}^d} \hat{f}(\boldsymbol{h}) \, \mathrm{e}^{2\pi \mathrm{i} \, \boldsymbol{h} \cdot \boldsymbol{x}}, \qquad \hat{f}(\boldsymbol{h}) := \int_{[0,1]^d} f(\boldsymbol{x}) \, \mathrm{e}^{-2\pi \mathrm{i} \, \boldsymbol{h} \cdot \boldsymbol{x}} \, \mathrm{d}\boldsymbol{x},$$

Error for an integrand using lattice rule approximation

For $f \in \mathcal{H}_\alpha$, with $\alpha > 1/2$, or actually,
for f with abs. conv. Fourier series, "Wiener algebra",

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{h} \in \mathbb{Z}^d} \hat{f}(\boldsymbol{h})\, \mathrm{e}^{2\pi \mathrm{i}\, \boldsymbol{h}\cdot\boldsymbol{x}}, \qquad \hat{f}(\boldsymbol{h}) := \int_{[0,1]^d} f(\boldsymbol{x})\, \mathrm{e}^{-2\pi \mathrm{i}\, \boldsymbol{h}\cdot\boldsymbol{x}}\, \mathrm{d}\boldsymbol{x},$$

we have

$$E(f) := \frac{1}{n} \sum_{k \in \mathbb{Z}_n} f\left(\frac{\boldsymbol{z} k \bmod n}{n}\right) - \int_{[0,1]^d} f(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} = \sum_{\substack{0 \neq \boldsymbol{h} \in \mathbb{Z}^d \\ \boldsymbol{h}\cdot\boldsymbol{z} \equiv 0 \pmod{n}}} \hat{f}(\boldsymbol{h}),$$

by the character sum for $\mathbb{Z}_n$, we have for $a = \boldsymbol{z} \cdot \boldsymbol{h} \in \mathbb{Z}$,

$$\frac{1}{n} \sum_{k \in \mathbb{Z}_n} \exp(2\pi \mathrm{i}\, k\, a/n) = \mathbb{1}\{a \equiv 0 \pmod{n}\}.$$

(Show other slides with duals. . .)

Worst-case error

Remember the definition:

Worst-case error for $f \in \mathcal{H}_\alpha$ for a given algorithm Q_n (e.g. $Q_{n,\boldsymbol{z}}$):

$$e^{\text{det}}(Q_n, \mathcal{H}_\alpha) := \sup_{\substack{f \in \mathcal{H}_\alpha \\ \|f\|_\alpha \leq 1}} |I(f) - Q_n(f)|.$$

Spaces based on series representations & Koksma–Hlawka

Assume L_2-ONB $\{\phi_h\}_h$, $\phi_0 = 1$, $Q_n(1) = 1$, and abs. summ.

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{h}} \hat{f}(\boldsymbol{h})\, \phi_{\boldsymbol{h}}(\boldsymbol{x}), \qquad \text{with} \qquad \hat{f}(\boldsymbol{h}) := \int_{[0,1]^d} f(\boldsymbol{x})\, \overline{\phi_{\boldsymbol{h}}(\boldsymbol{x})}\, \mathrm{d}\boldsymbol{x},$$

Spaces based on series representations & Koksma–Hlawka

Assume L_2-ONB $\{\phi_h\}_h$, $\phi_0 = 1$, $Q_n(1) = 1$, and abs. summ.

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{h}} \hat{f}(\boldsymbol{h})\, \phi_{\boldsymbol{h}}(\boldsymbol{x}), \qquad \text{with} \qquad \hat{f}(\boldsymbol{h}) := \int_{[0,1]^d} f(\boldsymbol{x})\, \overline{\phi_{\boldsymbol{h}}(\boldsymbol{x})}\, \mathrm{d}\boldsymbol{x},$$

then, for $r_{\alpha,\boldsymbol{\gamma}}(\boldsymbol{h}) > 0$ an "increasing" function,

$$|I(f) - Q_n(f)| = \left| \sum_{\boldsymbol{h} \neq 0} \hat{f}(\boldsymbol{h})\, Q_n(\phi_{\boldsymbol{h}})\, r_{\alpha,\boldsymbol{\gamma}}(\boldsymbol{h})\, r_{\alpha,\boldsymbol{\gamma}}^{-1}(\boldsymbol{h}) \right|$$

$$\leq \left(\sum_{\boldsymbol{h}} \left| \hat{f}(\boldsymbol{h}) \right|^p r_{\alpha,\boldsymbol{\gamma}}^p(\boldsymbol{h}) \right)^{1/p} \left(\sum_{\boldsymbol{h} \neq 0} |Q_n(\phi_{\boldsymbol{h}})|^q\, r_{\alpha,\boldsymbol{\gamma}}^{-q}(\boldsymbol{h}) \right)^{1/q}$$

norm $\times$ worst-case error* .

(See next slide.)

Worst-case error (continued...)

$$|I(f) - Q_n(f)| = \left|\sum_{\boldsymbol{h}\neq 0} \hat{f}(\boldsymbol{h})\, Q_n(\phi_{\boldsymbol{h}})\, r_{\alpha,\gamma}(\boldsymbol{h})\, r_{\alpha,\gamma}^{-1}(\boldsymbol{h})\right|$$

$$\leq \underbrace{\left(\sum_{\boldsymbol{h}} \left|\hat{f}(\boldsymbol{h})\right|^p r_{\alpha,\gamma}^p(\boldsymbol{h})\right)^{1/p}}_{\text{norm}} \times \underbrace{\left(\sum_{\boldsymbol{h}\neq 0} |Q_n(\phi_{\boldsymbol{h}})|^q\, r_{\alpha,\gamma}^{-q}(\boldsymbol{h})\right)^{1/q}}_{\text{worst-case error}^*}.$$

For $1 < p \leq \infty$ and compatible choices of $\phi_{\boldsymbol{h}}$, Q_n and $r_{\alpha,\gamma}$ we can find a "worst-case" representer $\xi(\boldsymbol{x})$ for which

$$|Q_n(\xi) - I(\xi)|^{1/q} = e(Q_n, \mathcal{F}_d), \qquad (*)$$

independent of the particular Q_n, e.g., Fourier series and lattice rules, Walsh series and digital nets, see Nuyens (2014) and Hickernell (1998a,b).

Reproducing kernel Hilbert spaces, $p = q = 2$

Given a one-dimensional reproducing kernel $K(x, y) = \overline{K(y, x)}$.
Suppose $\mathcal{H}(K)$ is separable: $\mathcal{H}(K) = \text{span}\{\phi_h\}_h$ and $\phi_0 = 1$.
Determine the eigenvalues and eigenfunctions, and assume $\lambda_0 = 1$,

$$\int_{[0,1]} \phi(x)\, \overline{K(x, y)}\, \mathrm{d}x = \lambda\, \phi(y).$$

Then

$$K(x, y) = \sum_h \frac{\phi_h(x)}{\sqrt{\lambda_h}}\, \overline{\frac{\phi_h(y)}{\sqrt{\lambda_h}}} = \sum_h \frac{\phi_h(x)}{\|\phi_h\|_{L_2}}\, \frac{\overline{\phi_h(y)}}{\|\phi_h\|_{L_2}},$$

the ϕ_h are L_2-orthogonal, with $\|\phi_h\|_{L_2} = \sqrt{\lambda_h}$ and $\|\phi_h\|_{\mathcal{H}} = 1$, with

$$\langle f, g\rangle_{\mathcal{H}} = \sum_h \lambda_h\, \hat{f}(h)\, \overline{\hat{g}(h)}, \qquad \|f\|^2_{\mathcal{H}} = \sum_h \lambda_h \left|\hat{f}(h)\right|^2.$$

Multivariate weighted reproducing kernel Hilbert space

Use the one-dimensional space as building block for d dimensions by taking weighted tensor products (tensor product basis):

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\mathfrak{u} \subseteq \{1,\ldots,d\}} \gamma_{\mathfrak{u}} \prod_{j \in \mathfrak{u}} K(x_j, y_j) = \sum_{\boldsymbol{h}} \gamma_{\mathfrak{u}(\boldsymbol{h})} \prod_{j=1}^{d} \frac{\phi_{h_j}(x_j)}{\sqrt{\lambda_{h_j}}} \frac{\overline{\phi_{h_j}(y_j)}}{\sqrt{\lambda_{h_j}}}$$

$$= \sum_{\boldsymbol{h}} r_{\alpha,\gamma}^{-2}(\boldsymbol{h})\, \phi_{\boldsymbol{h}}(\boldsymbol{x})\, \overline{\phi_{\boldsymbol{h}}(\boldsymbol{y})},$$

with

(You could now interpret α as the decay of the eigenvalues.)

$$r_{\alpha,\gamma}^{-2}(\boldsymbol{h}) = \gamma_{\mathfrak{u}(\boldsymbol{h})} \prod_{j \in \mathfrak{u}} \lambda_{h_j}^{-1} = \gamma_{\mathfrak{u}(\boldsymbol{h})} \prod_{j=1}^{d} \lambda_{h_j}^{-1},$$

and $\mathfrak{u}(\boldsymbol{h}) = \{h_j : h_j \neq 0\} = \operatorname{supp}(\boldsymbol{h})$. Now, with $\gamma_{\emptyset} = 1$, $Q_n(1) = 1$,

$$e^2(Q_n; \mathcal{H}) = -1 + \sum_{k,\ell=1}^{n} w_k\, w_\ell\, K(\boldsymbol{x}_k, \boldsymbol{y}_\ell).$$

For a shift-invariant space and lattice rule

For a shift-invariant space we have

$$K(\boldsymbol{x}, \boldsymbol{y}) = K(\boldsymbol{x} - \boldsymbol{y}, 0)$$

and for a lattice rule we have

$$\boldsymbol{x}_k - \boldsymbol{x}_{k'} = \boldsymbol{x}_{k-k' \bmod n},$$

all on the torus $[0,1)^d$.

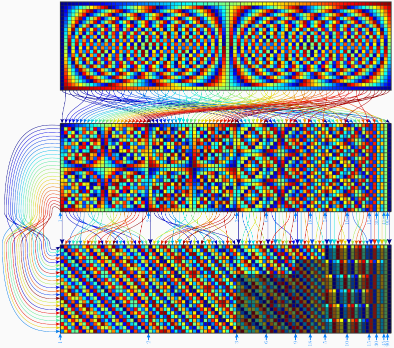
Hence:

$$\begin{aligned} e^2(Q_{n,\boldsymbol{z}}; \mathcal{H}) &= -1 + \sum_{k,\ell=1}^{n} w_k\, w_\ell\, K(\boldsymbol{x}_k, \boldsymbol{y}_\ell) \\ &= -1 + \sum_{\ell=1}^{n} \frac{1}{n} \sum_{k=1}^{n} \frac{1}{n} K(\boldsymbol{x}_{k-\ell \bmod n}, 0) \\ &= -1 + \frac{1}{n} \sum_{k=1}^{n} K(\boldsymbol{x}_k, 0). \end{aligned}$$

Fast component-by-component constructions

Construction of lattice rules and polynomial lattice rules

Point sets constructed for weighted spaces using fast component-by-component constructions using number theoretic transforms.



See https://www.cs.kuleuven.be/~dirkn/qmc4pde/ and https://www.cs.kuleuven.be/~dirkn/fast-cbc/.

See, e.g., N. & Cools (2006a,2006b), Cools, Kuo, & N. (2006), Dick, Kuo, Le Gia, N. & Schwab (2014), N. (2014), Kuo & N. (2016), . . .
Variations and speedups by: Gantner, Kritzer, Laimer, Leobacher, Pillichshammer, Schwab, . . . New methods: Ebert, Kritzer, N., Osisiogu (2021), Kuo, N., Wilkes (2023), N., Wilkes (2023), . . .

Point generators

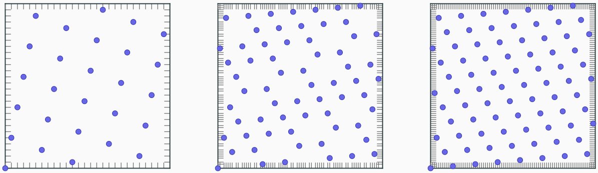
- Matlab/Octave: procedural generators like Matlab's `rand`:
  - `latticeseq_b2.m`: radical inverse lattice sequence generator,
  - `digitalseq_b2g.m`: gray coded radical inverse digital sequence generator (incl. higher-order, max 53 bit).
- Python: iterator classes, which can be used as standalone point generators from the command line (`__main__`):
  - `latticeseq_b2.py`: iterator based (`__iter__`), `set_state` for parallel computing,
  - `digitalseq_b2g.py`: ditto, arbitrary precision using `mpmath` if needed.
- C++: header file based implementation with driver program for the command line:
  - `latticeseq_b2.(h|cpp)`: complies to `ForwardIterator` concept, `set_state` for parallel computing,
  - `digitalseq_b2g.(h|cpp)`: ditto, max 64 bit.

Welcome to "The Magic Point Shop!"

Different flavours of quasi-Monte Carlo points to choose:

- Lattice rules.
- Lattice sequences.
- Polynomial lattice rules.
- Interlaced Sobol' sequences (higher-order).
- Interlaced polynomial lattice rules (higher-order).

And code (C++ and Matlab) to use them...



Subsidiaries: QMC4PDE: construct points for parametrised PDEs.

Second demo

Julia break...

- The van der Corput sequence for $d = 1$.
- The Korobov trick.
- Estimating the error by use of standard error...

The end for today

The end for today from me

- Thanks for listening...

The end for today from me

- Thanks for listening. . .
- Please ask questions. . .

The end for today from me

- Thanks for listening...
- Please ask questions...
- Now or later...

The end for today from me

- Thanks for listening. . .
- Please ask questions. . .
- Now or later. . .

Tomorrow more advanced things: weighted function spaces, function approximation, . . .