Richardson extrapolation of polynomial lattice rules for smooth functions

Josef Dick

The University of New South Wales Sydney

Joint with Takashi Goda and Takehito Yoshiki

MCQMC 2018

Numerical integration

Problem

Compute

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

where $s \in \mathbb{N}$ and $f : [0,1]^s \to \mathbb{R}$ is integrable.

- Linear algorithm denotes an approximation of $I(f)$ by the form

$$Q_{P,W}(f) := \sum_{n=0}^{N-1} w_n f(\mathbf{x}_n).$$

where $P = \{\mathbf{x}_0, \ldots, \mathbf{x}_{N-1}\} \subset [0,1]^s$, $W = \{w_0, \ldots, w_{N-1}\} \subset \mathbb{R}$.

Quasi-Monte Carlo (QMC)

- QMC integration denotes a special case of linear algorithm with

$$w_0 = \cdots = w_{N-1} = \frac{1}{N},$$

i.e., an approximation of $I(f)$ by the form

$$Q_P(f) := \frac{1}{N} \sum_{n=0}^{N-1} f(\mathbf{x}_n).$$

Richardson extrapolation

Classical technique invented by Lewis Fry Richardson (1881–1953; English mathematician, physicist, meteorologist, psychologist and pacifist).

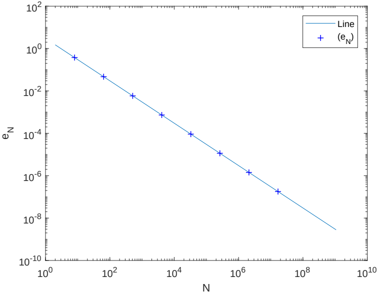
Richardson extrapolation

Classical technique invented by Lewis Fry Richardson (1881–1953; English mathematician, physicist, meteorologist, psychologist and pacifist).

Consider sequence of real numbers $e_1 > e_2 > e_3 > \cdots$ such that $\lim_{n\to\infty} e_n = 0$. Assume $e_1, e_2, \ldots$ 'behave nicely' – for instance, assume that the points

$$(\log N_1, \log e_{N_1}), (\log N_2, \log e_{N_2}), (\log N_3, \log e_{N_3}), \ldots$$

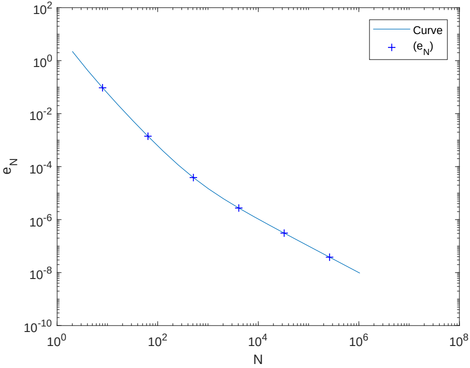
lie on a line.



More generally, assume that the points

$$(\log N_1, \log e_{N_1}), (\log N_2, \log e_{N_2}), (\log N_3, \log e_{N_3}), \ldots$$

lie on some curve.



Richardson's idea: If we can find the curve, we can guess e_N for large values of N and use this information to eliminate/reduce the error.

Richardson extrapolation

Assume that the points $(\log N_i, \log e_{N_i})$, $i = 1, 2, \ldots$ lie on a straight line

$$\log e_N = \log c - \alpha \log N \quad \Leftrightarrow \quad e_N = \frac{c}{N^\alpha}.$$

Richardson extrapolation

Assume that the points $(\log N_i, \log e_{N_i})$, $i = 1, 2, \ldots$ lie on a straight line

$$\log e_N = \log c - \alpha \log N \quad \Leftrightarrow \quad e_N = \frac{c}{N^\alpha}.$$

The errors e_N are given by

$$e_N = I - Q_N.$$

Richardson extrapolation

Assume that the points $(\log N_i, \log e_{N_i})$, $i = 1, 2, \ldots$ lie on a straight line

$$\log e_N = \log c - \alpha \log N \quad \Leftrightarrow \quad e_N = \frac{c}{N^\alpha}.$$

The errors e_N are given by

$$e_N = I - Q_N.$$

Then we have

$$0 = (2N)^\alpha e_{2N} - N^\alpha e_N = (2N)^\alpha I - N^\alpha I - ((2N)^\alpha Q_{2N} - N^\alpha Q_N),$$

which implies

$$I = \frac{(2N)^\alpha Q_{2N} - N^\alpha Q_N}{(2N)^\alpha - N^\alpha} = \frac{2^\alpha}{2^\alpha - 1} Q_{2N} - \frac{1}{2^\alpha - 1} Q_N.$$

Richardson extrapolation

Assume that the points $(\log N_i, \log e_{N_i})$, $i = 1, 2, \ldots$ lie on a straight line

$$\log e_N = \log c - \alpha \log N \quad \Leftrightarrow \quad e_N = \frac{c}{N^\alpha}.$$

The errors e_N are given by

$$e_N = I - Q_N.$$

Then we have

$$0 = (2N)^\alpha e_{2N} - N^\alpha e_N = (2N)^\alpha I - N^\alpha I - ((2N)^\alpha Q_{2N} - N^\alpha Q_N),$$

which implies

$$I = \frac{(2N)^\alpha Q_{2N} - N^\alpha Q_N}{(2N)^\alpha - N^\alpha} = \frac{2^\alpha}{2^\alpha - 1} Q_{2N} - \frac{1}{2^\alpha - 1} Q_N.$$

Note: Sum of the weights is 1 and independent of N.

Assume that

$$e_N = \frac{c_1}{N} + \frac{c_2}{N^2}.$$

Then

$$e_N^{(1)} = e_N - 2e_{2N} = \frac{c_1}{N} + \frac{c_2}{4N^2} - 2\frac{c_1}{2N} - 2\frac{c_2}{4N^2} = \frac{c_2}{2N^2}.$$

Assume that

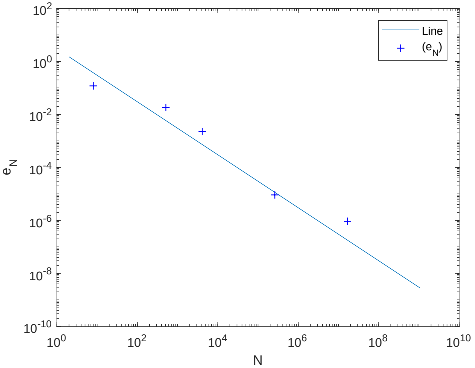
$$e_N = \frac{c_1}{N} + \frac{c_2}{N^2}.$$

Then

$$e_N^{(1)} = e_N - 2e_{2N} = \frac{c_1}{N} + \frac{c_2}{4N^2} - 2\frac{c_1}{2N} - 2\frac{c_2}{4N^2} = \frac{c_2}{2N^2}.$$

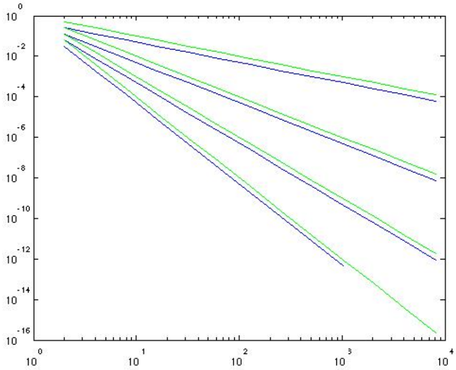
Now repeat the procedure using $e_N^{(1)}$ to eliminate c_2...

If the points are not on a line (curve) this could go horribly wrong...



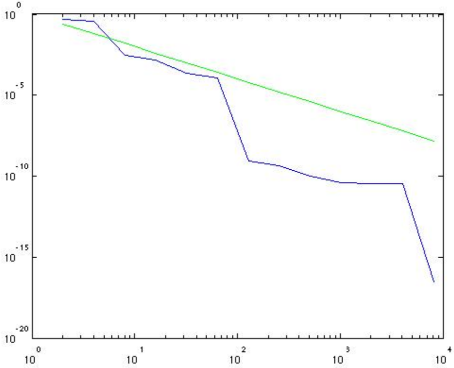
Could Richardson extrapolation work for higher order QMC?

Some HOQMC numerical examples: $f(x) = x$



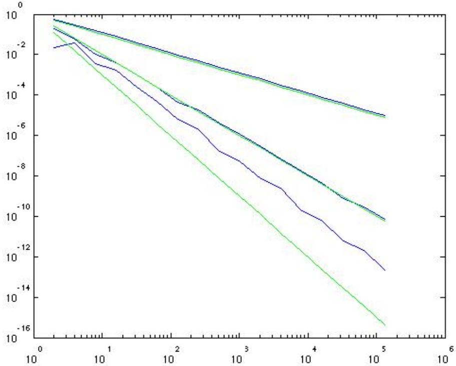
We use higher order Sobol points of order $1, 2, 3, 4$.

Some HOQMC numerical examples: $f(x) = sin(2\pi x)$



We use higher order Sobol points of order 2.

Some HOQMC numerical examples: $f(x) = 5/2x^{3/2}$



We use higher order Sobol points of order $1, 2, 3$.

Look at theory...

Motivation: Fast QMC matrix vector multiplication

In some applications integrals are of the form

$$\int_{[0,1]^s} f(\mathbf{y}A)\,\mathrm{d}\mathbf{y} \approx \frac{1}{N}\sum_{n=0}^{N-1} f(\mathbf{x}_n A).$$

This appears in particular in PDEs with random coefficients, where the main computational cost is the computation of $\mathbf{x}_n A$ for $n = 1, 2, \ldots, N-1$.

(D., Kuo, Le Gia, Schwab, 2015)

For lattice rules and polynomial lattice rules, we define

$$X = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{N-1} \end{pmatrix}.$$

Then

$$X = CP,$$

where C is a circulant matrix and P is a matrix which has one value of 1 in each column and the remaining values are 0's.

Using the fast Fourier transform we can compute CPA in $\mathcal{O}(sN \log N)$ operations assuming that $N \asymp s^z$.

For lattice rules and polynomial lattice rules, we define

$$X = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{N-1} \end{pmatrix}.$$

Then

$$X = CP,$$

where C is a circulant matrix and P is a matrix which has one value of 1 in each column and the remaining values are 0's.

Using the fast Fourier transform we can compute CPA in $\mathcal{O}(sN \log N)$ operations assuming that $N \asymp s^z$.

This does not work for interlaced polynomial lattice rules. But it works with extrapolated polynomial lattice rules.

Fast QMC matrix vector product numerical result for PDE with random coefficients with N QMC points, truncation dimension s and dimension of finite element space M.

N	509	1021	2053	4001	8009	16001
std.	190	1346	10610	74550	$\approx$ 144 hrs	$\approx$ 1000 hrs
fast	0.462	1.562	5.591	19.678	87.246	342.615

Table: Times (in seconds) where $M = s = 2N$

N	509	1021	2053 ·	4001	8009	16001
std.	1.272	3.570	10.813	30.127	89.42	273.873
fast	0.059	0.126	0.265	0.516	1.113	2.443

Table: Times (in seconds) where $M = s = \lceil \sqrt{N} \rceil$

N	67	127 ·	257	509
std.	6	82	1699	27935
fast	0.243	1.385	11.268	107.042

Table: Times (in seconds) where $s = N$ and $M = N^2$

Integration error

- For a digital net P with $C_1, \ldots, C_s \in \mathbb{F}_b^{n \times m}$, we have

$$Q_P(f) - I(f) = \frac{1}{b^m} \sum_{h=0}^{b^m-1} f(\mathbf{x}_h) - I(f)$$

$$= \sum_{\substack{\mathbf{k} \in P^\perp \setminus \{\mathbf{0}\} \\ b^n \nmid \mathbf{k}}} \hat{f}(\mathbf{k}) + \frac{c_1(f)}{b^n} + \cdots + \frac{c_{\alpha-1}(f)}{b^{(\alpha-1)n}} + O(b^{-\alpha n}).$$

In case of *square* generating matrices, i.e., the case $n = m$, a digital net cannot achieve a convergence rate better than $O(N^{-1})$ no matter how small the first term is.

Possible remedy 1

- Consider *non-square* matrices, say, $C_1, \ldots, C_s \in \mathbb{F}_b^{\alpha m \times m}$. Then

$$Q_P(f) - I(f) = \sum_{\substack{\mathbf{k} \in P^\perp \setminus \{\mathbf{0}\} \\ b^{\alpha m} \nmid \mathbf{k}}} \hat{f}(\mathbf{k}) + \frac{c_1(f)}{b^{\alpha m}} + \cdots + \frac{c_{\alpha-1}(f)}{b^{(\alpha-1)\alpha m}} + O(b^{-\alpha^2 m}).$$

- The remaining task is to find P such that the first term is small.

1. Digit interlacing algorithm (D., 2007, 2008)
   $\Rightarrow$ Interlaced polynomial lattice rule (Goda & D., 2015; Goda, 2015; ...)
2. Higher order polynomial lattice rule (D. & Pillichshammer, 2007; Baldeaux, D., Leobacher, Nuyens & Pillichshammer, 2012;...)

- Both approaches achieve $O(b^{-(\alpha-\varepsilon)m})$ error convergence.

Possible remedy 2 (new, this study)

- Consider a set $P_1, \ldots, P_\alpha$ with *square* matrices and $|P_i| = b^{m-\alpha+i}$.

$$Q_{P_1}(f) - I(f) = \sum_{\substack{\mathbf{k} \in P_1^\perp \setminus \{\mathbf{0}\} \\ b^{m-\alpha+1} \nmid \mathbf{k}}} \hat{f}(\mathbf{k}) + \frac{c_1(f)}{b^{m-\alpha+1}} + \cdots + \frac{c_{\alpha-1}(f)}{b^{(\alpha-1)(m-\alpha+1)}} + O(b^{-\alpha(m-\alpha+1)})$$

$$\vdots$$

$$Q_{P_\alpha}(f) - I(f) = \sum_{\substack{\mathbf{k} \in P_\alpha^\perp \setminus \{\mathbf{0}\} \\ b^{m} \nmid \mathbf{k}}} \hat{f}(\mathbf{k}) + \frac{c_1(f)}{b^{m}} + \cdots + \frac{c_{\alpha-1}(f)}{b^{(\alpha-1)m}} + O(b^{-\alpha m})$$

- Then take a weighted sum of $Q_{P_i}(f)$ s.t. the middle terms vanish.

Possible remedy 2 (cont'd)

- This can be done by applying Richardson extrapolation recursively, i.e., we can explicitly compute the coefficients r_i's which satisfy $\sum_{i=1}^{\alpha} r_i = 1$ and

$$\sum_{i=1}^{\alpha} \frac{r_i}{b^{m-\alpha+i}} \sum_{n=0}^{b^{m-\alpha+i}-1} f(\mathbf{x}_n^{(i)}) - I(f) = \sum_{i=1}^{\alpha} r_i \sum_{\substack{\mathbf{k} \in P_i^{\perp} \setminus \{\mathbf{0}\} \\ b^{m-\alpha+i} \nmid \mathbf{k}}} \hat{f}(\mathbf{k}) + O(b^{-\alpha m}).$$

- The remaining task is to find P_i's such that the first term is small. This strategy leads to
Extrapolated polynomial lattice rule (D., Goda, Yoshiki, 2018+)
- The r_i are independent of the number of points $N = b^{m-\alpha+1} + \cdots + b^m$ and

$$\sum_{i=1}^{\alpha} |r_i| \le \prod_{i=1}^{\alpha-1} \frac{b^i + 1}{b^i - 1}.$$

Polynomial lattice point set

Definition (Niederreiter, 1992)

- For $m, s \in \mathbb{N}$, let $p \in \mathbb{F}_b[x]$ with $\deg(p) = m$ and let $\mathbf{q} \in (\mathbb{F}_b[x])^s$.
- $P(p, \mathbf{q})$ is a digital net with square generating matrices

$$C_j = \begin{pmatrix} a_1^{(j)} & a_2^{(j)} & \cdots & a_m^{(j)} \\ a_2^{(j)} & a_3^{(j)} & \cdots & a_{m+1}^{(j)} \\ \vdots & \vdots & \ddots & \vdots \\ a_m^{(j)} & a_{m+1}^{(j)} & \ddots & a_{2m-1}^{(j)} \end{pmatrix} \in \mathbb{F}_b^{m \times m},$$

where

$$\frac{q_j(x)}{p(x)} = \sum_{i=1}^{\infty} a_i^{(j)} x^{-i}.$$

Sobolev spaces $W_{s,\alpha,q,\gamma}$

- The function space $W_{s,\alpha,q,\gamma}$ consists of functions f having partial mixed derivatives $f^{(\tau_1,\ldots,\tau_s)}$ for $0 \le \tau_1, \ldots \tau_s \le \alpha$ with the finite norm

$$\|f\|_{s,\alpha,q,\gamma} := \sup_{u \subseteq \{1,\ldots,s\}} \gamma_u^{-1}$$
$$\times \left(\sum_{v \subseteq u} \sum_{\boldsymbol{\tau} \in \{1,\ldots,\alpha\}^{|u \setminus v|}} \int_{[0,1)^{|v|}} \left| \int_{[0,1)^{s-|v|}} f^{(\boldsymbol{\tau}_{u\setminus v}, \boldsymbol{\alpha}_v, \mathbf{0})}(\mathbf{x}) \, \mathrm{d}\mathbf{x}_{-v} \right|^q \mathrm{d}\mathbf{x}_v \right)^{1/q}.$$

- This function space has been introduced in the context of PDEs with random coefficients (D., Kuo, Le Gia, Nuyens & Schwab, 2014).

Worst-case error bound

Theorem (D., Goda, Yoshiki, 2018+)

- *For* $m, s \in \mathbb{N}$, $m \geq \alpha$, *let* $P(p_i, \mathbf{q}_i)$ *be a polynomial lattice point set with* $\deg(p_i) = m - \alpha + i$ *for* $i = 1, \ldots, \alpha$.
- *The worst-case error of an extrapolated polynomial lattice rule is bounded by*

$$\sup_{\|f\| \leq 1} \left| \sum_{i=1}^{\alpha} r_i Q_{P(p_i, \mathbf{q}_i)}(f) - I(f) \right| \leq \sum_{i=1}^{\alpha} |r_i| B_{s,\gamma}(P(p_i, \mathbf{q}_i)) + O(N^{-\alpha})$$

where $N = b^{m-\alpha+1} + \cdots + b^m$, *and* $B_{s,\gamma}$ *is a computable quality criterion and independent of* q.

CBC construction

Algorithm (Component-by-component)

1. *For* $i = 1, \ldots, \alpha$, *do the following:*
2. *Let* p_i *be irreducible with* $\deg(p_i) = m - \alpha + i$ *and* $q_{i,1}^* = 1 \in \mathbb{F}_b[x]$.
3. *For* $j = 2, \ldots, s$, *find* $q_{i,j}^* \in \mathbb{F}_b[x]$ *which minimizes*

$$B_{j,\gamma}(P(p_i, (q_{i,1}^*, \ldots, q_{i,j-1}^*, q_{i,j})))$$

as a function of $q_{i,j}$ *where* $\deg(q_{i,j}) < m - \alpha + i$.

- In case of product weights $\gamma_u = \prod_{j \in u} \gamma_j$, the CBC algorithm can find $\mathbf{q}_i$ such that $B_{s,\gamma}(P(p_i, \mathbf{q}_i)) = O(N^{-\alpha+\varepsilon})$ for $i = 1, \ldots, \alpha$.
- Therefore, an extrapolated polynomial lattice rule can achieve the almost optimal rate of convergence.

CBC construction

Algorithm (Component-by-component)

1. *For* $i = 1, \ldots, \alpha$, *do the following:*
2. *Let* p_i *be irreducible with* $\deg(p_i) = m - \alpha + i$ *and* $q^*_{i,1} = 1 \in \mathbb{F}_b[x]$.
3. *For* $j = 2, \ldots, s$, *find* $q^*_{i,j} \in \mathbb{F}_b[x]$ *which minimizes*

$$B_{j,\gamma}(P(p_i, (q^*_{i,1}, \ldots, q^*_{i,j-1}, q_{i,j})))$$

as a function of $q_{i,j}$ *where* $\deg(q_{i,j}) < m - \alpha + i$.

- In case of product weights $\gamma_u = \prod_{j \in u} \gamma_j$, the CBC algorithm can find $\mathbf{q}_i$ such that $B_{s,\gamma}(P(p_i, \mathbf{q}_i)) = O(N^{-\alpha+\varepsilon})$ for $i = 1, \ldots, \alpha$.
- Therefore, an extrapolated polynomial lattice rule can achieve the almost optimal rate of convergence.
- Many other methods also achieve this convergence rate: Frolov (-Ullrich) rules, sparse grids, higher order nets, interlaced polynomial lattice rules, ... But is there a fast matrix vector product for such rules?

Construction cost

- Possible to use fast CBC algorithm due to Nuyens & Cools (2006).
- Each $\mathbf{q}_i$ can be found by the cost $O((s+\alpha)\deg(p_i) b^{\deg(p_i)})$ with $O(b^{\deg(p_i)})$ memory. In total, we need the cost of order

$$\begin{aligned}\sum_{i=1}^{\alpha}(s+\alpha)\deg(p_i) b^{\deg(p_i)} &\le (s+\alpha)\deg(p_1)\sum_{i=1}^{\alpha} b^{\deg(p_i)} \\ &= (s+\alpha)\deg(p_1) N \le (s+\alpha) N \log_b N\end{aligned}$$

  This compares favorably with ...
- Interlaced polynomial lattice rule: $O(s\alpha N \log N)$
  - We need to search for αs components.
- Higher order polynomial lattice rule: $O(s\alpha N^{\alpha} \log N)$
  - The search space of generating vectors is exponentially larger in α.

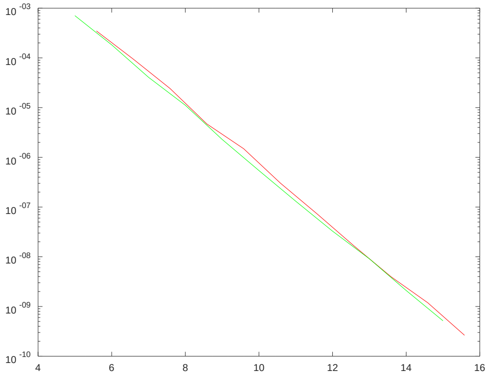
Numerical experiment ($\alpha = 2$)

- Consider the test function:

$$f(\mathbf{x}) = \prod_{j=1}^{s} \left[1 + \frac{\gamma_j}{1 + \gamma_j x_j} \right].$$

for $s = 100$ and $\gamma_j = j^{-2}$.

- Extrapolated rule with $\alpha = 2$ (green), interlaced rule with $\alpha = 2$ (red)



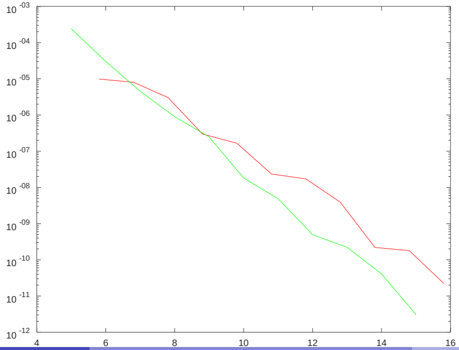
Numerical experiment ($\alpha = 3$)

- Consider the test function:

$$f(\mathbf{x}) = \prod_{j=1}^{s} \left[1 + \frac{\gamma_j}{1 + \gamma_j x_j} \right].$$

for $s = 100$ and $\gamma_j = j^{-2}$.

- Extrapolated rule with $\alpha = 3$ (green) is slightly worse than interlaced rule with $\alpha = 3$ (red)



Thank you for your attention!

Digital net

Definition

- For prime b, $\mathbb{F}_b$ denotes the b-element field.
- For $s, m, n \in \mathbb{N}$, let $C_1, \ldots, C_s \in \mathbb{F}_b^{n \times m}$.
- Denote the b-adic expansion of $0 \le h < b^m$ by $h = (\eta_{m-1} \ldots \eta_0)_b$.
- Set $\mathbf{x}_h = (x_{h,1}, \ldots, x_{h,s}) \in [0,1]^s$, where

$$x_{h,j} = (0.\xi_{1,h,j} \ldots \xi_{n,h,j})_b \in [0,1]$$

  with

$$(\xi_{1,h,j}, \ldots, \xi_{n,h,j})^\top = C_j \cdot (\eta_0, \ldots, \eta_{m-1})^\top.$$

Then we call $P = \{\mathbf{x}_h \colon 0 \le h < b^m\}$ a *digital net* over $\mathbb{F}_b$.

The parameter m determines the size of point set, and n the precision.

Dual net

Definition

For a digital net P, the dual net is defined by

$$P^{\perp} := \left\{ \mathbf{k} = (k_1, \ldots, k_s) \in \mathbb{N}_0^s \colon C_1^{\top} \vec{k}_1 \oplus \cdots \oplus C_s^{\top} \vec{k}_s = \mathbf{0} \in \mathbb{F}_b^m \right\} \subset \mathbb{N}_0^s,$$

where $\vec{k} = (\kappa_0, \ldots, \kappa_{n-1})^{\top}$ for $k = (\ldots \kappa_1 \kappa_0)_b$.

- $P^{\perp}$ includes every $\mathbf{k} = (k_1, \ldots, k_s)$ such that $b^n \mid \mathbf{k}$, i.e., $b^n \mid k_j$ holds for all j. This means

$$P^{\perp} \supset \{\mathbf{k} \in \mathbb{N}_0^s \colon b^n \mid \mathbf{k}\} = P^{\perp}_{\mathrm{grid},n}$$

where

$$P_{\mathrm{grid},n} = \left\{ \left(\frac{a_1}{b^n}, \ldots, \frac{a_s}{b^n} \right) : 0 \le a_1, \ldots, a_s < b^n \right\}.$$

Integration error

- For a digital net P with $C_1, \ldots, C_s \in \mathbb{F}_b^{n \times m}$, we have

$$\begin{aligned} Q_P(f) - I(f) &= \frac{1}{b^m} \sum_{h=0}^{b^m-1} f(\mathbf{x}_h) - I(f) \\ &= \sum_{\mathbf{k} \in P^\perp \setminus \{\mathbf{0}\}} \hat{f}(\mathbf{k}) \\ &= \sum_{\substack{\mathbf{k} \in P^\perp \setminus \{\mathbf{0}\} \\ b^n \nmid \mathbf{k}}} \hat{f}(\mathbf{k}) + \sum_{\substack{\mathbf{k} \in P^\perp \setminus \{\mathbf{0}\} \\ b^n \mid \mathbf{k}}} \hat{f}(\mathbf{k}) \\ &= \sum_{\substack{\mathbf{k} \in P^\perp \setminus \{\mathbf{0}\} \\ b^n \nmid \mathbf{k}}} \hat{f}(\mathbf{k}) + \sum_{\mathbf{k} \in P^\perp_{\mathrm{grid},n} \setminus \{\mathbf{0}\}} \hat{f}(\mathbf{k}) \\ &= \sum_{\substack{\mathbf{k} \in P^\perp \setminus \{\mathbf{0}\} \\ b^n \nmid \mathbf{k}}} \hat{f}(\mathbf{k}) + \left(Q_{P_{\mathrm{grid},n}}(f) - I(f)\right) \end{aligned}$$

where $\hat{f}(\mathbf{k})$ denotes the $\mathbf{k}$-th Walsh coefficient of f.

Euler-Maclaurin formula for $Q_{P_{\mathrm{grid},n}}(f)$

Lemma

If f has partial mixed derivatives up to order $\alpha \ge 2$ in each variable,

$$Q_{P_{grid,n}}(f) = I(f) + \frac{c_1(f)}{b^n} + \cdots + \frac{c_{\alpha-1}(f)}{b^{(\alpha-1)n}} + O(b^{-\alpha n}).$$

Here

$$c_\tau(f) = \sum_{\substack{(\tau_1,\ldots,\tau_s)\in\mathbb{N}_0^s \\ \tau_1+\cdots+\tau_s=\tau}} \prod_{j=1}^{s} \frac{B_{\tau_j}}{\tau_j!} \cdot I(f^{(\tau_1,\ldots,\tau_s)})$$

where B_τ denotes the τ-th Bernoulli number.