Woudschoten Conference 2016
Introduction to quasi-Monte Carlo methods, with application to PDEs with random coefficients

Part 2
Application of QMC methods to PDEs with random coefficients
– a survey of analysis and implementation

Frances Kuo

f.kuo@unsw.edu.au

University of New South Wales, Sydney, Australia

~ Based on a J. FoCM survey of the same title, written jointly with Dirk Nuyens (KU Leuven, Belgium) ~
Outline

- Motivating example – a flow through a porous medium
- Quasi-Monte Carlo (QMC) methods
- Component-by-component (CBC) construction
- Application of QMC theory to PDEs with random coefficients
  - estimate the norm
  - choose the weights
- Software
- Concluding remarks
Motivating example

Uncertainty in groundwater flow
eg. risk analysis of radwaste disposal or CO$_2$ sequestration

Darcy’s law
mass conservation law

\[ q + a \nabla p = \kappa \quad \text{in } D \subset \mathbb{R}^d, d = 1, 2, 3 \]
\[ \nabla \cdot q = 0 \]

together with boundary conditions

Uncertainty in \( a(\mathbf{x}, \omega) \) leads to uncertainty in \( q(\mathbf{x}, \omega) \) and \( p(\mathbf{x}, \omega) \)
Expected values of quantities of interest

To compute the expected value of some quantity of interest:

1. Generate a number of realizations of the random field
   (Some approximation may be required)
2. For each realization, solve the PDE using e.g. FEM / FVM / mFEM
3. Take the average of all solutions from different realizations

This describes Monte Carlo simulation.

Example: particle dispersion

\[
\begin{align*}
\frac{\partial p}{\partial n} &= 0 \\
p &= 1 \\
\text{release point} &\rightarrow \\
\frac{\partial p}{\partial n} &= 0 \\
p &= 0
\end{align*}
\]
Expected values of quantities of interest

To compute the expected value of some quantity of interest:

1. Generate a number of realizations of the random field (Some approximation may be required)
2. For each realization, solve the PDE using e.g. FEM / FVM / mFEM
3. Take the average of all solutions from different realizations

This describes Monte Carlo simulation.

NOTE: expected value = (high dimensional) integral

→ use quasi-Monte Carlo methods

\[ s = \text{stochastic dimension} \]
MC v.s. QMC in the unit cube

\[ \int_{[0,1]^n} f(y) \, dy \approx \frac{1}{n} \sum_{i=1}^{n} f(t_i) \]

Monte Carlo method
\( t_i \) random uniform
\( n^{-1/2} \) convergence
order of variables irrelevant

Quasi-Monte Carlo methods
\( t_i \) deterministic
close to \( n^{-1} \) convergence or better
more effective for earlier variables and lower-order projections
order of variables very important

First 64 points of a 2D Sobol’ sequence

64 random points

A lattice rule with 64 points

use randomized QMC methods for error estimation
Two main families of QMC methods:

- \((t,m,s)\)-nets and \((t,s)\)-sequences
- lattice rules

Important developments:

- component-by-component (CBC) construction, “fast” CBC
- higher order digital nets

- Niederreiter book (1992)
- Sloan and Joe book (1994)
- Dick, K., Sloan Acta Numerica (2013)
- Nuyens and Cools (2006)
- Dick (2008)
Application of QMC to PDEs with random coefficients


Also
- Schwab (Proceedings of MCQMC 2012)
- Le Gia (Proceedings of MCQMC 2012)
- Dick, Le Gia, Schwab (in review, in review, in progress)
- Gantner, Schwab (Proceedings of MCQMC 2014)
- Robbe, Nuyens, Vandewalle (in review)
- Scheichl, Stuart, Teckentrup (in review)
- Gilbert, Graham, K., Scheichl, Sloan (in progress)
- Graham, Parkinson, Scheichl (in progress)
- etc.
### Application of QMC to PDEs with random coefficients

Three different QMC theoretical settings:

- **[1,2]** Weighted Sobolev space over $[0, 1]^s$ and "randomly shifted lattice rules"
- **[3,4]** Weighted space setting in $\mathbb{R}^s$ and "randomly shifted lattice rules"
- **[5,6]** Weighted space of smooth functions over $[0, 1]^s$ and (deterministic) "interlaced polynomial lattice rules"

- K., Nuyens (J. FoCM) – a survey of [1,2], [3,4], [5,6] in a unified view

<table>
<thead>
<tr>
<th>Setting</th>
<th>Uniform</th>
<th>Lognormal KL expansion</th>
<th>Lognormal Circulant embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiments only</td>
<td></td>
<td></td>
<td>[0]*</td>
</tr>
<tr>
<td>First order, single-level</td>
<td>[1]</td>
<td>[3]</td>
<td>[7]*</td>
</tr>
<tr>
<td>First order, multi-level</td>
<td>[2]</td>
<td>[4]</td>
<td></td>
</tr>
<tr>
<td>Higher order, single-level</td>
<td>[5]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Higher order, multi-level</td>
<td>[6]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The * indicates there are accompanying numerical results
Common features among all three QMC theoretical settings:

- Separation in error bound
  \[ \text{(rms) QMC error} \leq \text{(rms) worst case error} \times \text{norm of integrand} \]
- Weighted spaces
- Pairing of QMC rule with function space
- Optimal rate of convergence
- Rate and constant independent of dimension
- **Fast component-by-component (CBC) construction**
- Extensible or embedded rules

Application of QMC theory

- **Estimate the norm** (critical step)
- **Choose the weights**
- Weights as input to the CBC construction
Lattice rules

**Rank-1 lattice rules** have points

\[
\mathbf{t}_i = \frac{\mathbf{i}}{\mathbf{n}}, \quad i = 1, 2, \ldots, n
\]

\(z \in \mathbb{Z}^s\) – the *generating vector*, with all components *coprime* to \(n\)

\(\text{frac}(\cdot)\) – means to take the fractional part of all components

\(~\) quality determined by the choice of \(z\) \(~\)

\[
n = 64 \quad z = (1, 19) \quad \mathbf{t}_i = \frac{\mathbf{i}}{64}(1, 19)
\]
Randomly shifted lattice rules

**Shifted rank-1 lattice rules** have points

\[ t_i = \text{frac} \left( \frac{i}{n} z + \Delta \right), \quad i = 1, 2, \ldots, n \]

\[ \Delta \in [0, 1)^s \text{ – the shift} \]

\[ \sim \text{ use a number of random shifts for error estimation} \sim \]

A lattice rule with 64 points

A shifted lattice rule with 64 points

shifted by \( \Delta = (0.1, 0.3) \)
Component-by-component construction

Want to find $z$ for which some error criterion is as small possible

~ Exhaustive search is practically impossible - too many choices! ~

CBC algorithm  

[Korobov (1959); Sloan, Reztsov (2002); Sloan, K., Joe (2002),...]

1. Set $z_1 = 1$.

2. With $z_1$ fixed, choose $z_2$ to minimize the error criterion in 2D.

3. With $z_1, z_2$ fixed, choose $z_3$ to minimize the error criterion in 3D.

4. etc.

[K. (2003); Dick (2004)]

Optimal rate of convergence $O\left(n^{-1+\delta}\right)$ in “weighted Sobolev space”, independently of $s$ under an appropriate condition on the weights

~ Averaging argument: there is always one choice as good as average! ~

[Nuyens, Cools (2006)]

Cost of algorithm for “product weights” is $O(n \log n s)$ using FFT

[Hickernell, Hong, L’Ecuyer, Lemieux (2000); Hickernell, Niederreiter (2003)]

[Dick, Pillichshammer, Waterhouse (2007)]

Setting 1: standard QMC for unit cube

Worst case error bound

\[
\left| \int_{[0,1]^s} f(y) \, dy - \frac{1}{n} \sum_{i=1}^{n} f(t_i) \right| \leq e_{\text{wor}}(t_1, \ldots, t_n) \| f \|_{\gamma}
\]

Weighted Sobolev space

\[
\| f \|_{\gamma}^2 = \sum_{u \subseteq \{1:s\}} \frac{1}{\gamma_u} \int_{[0,1]^{|u|}} \left| \frac{\partial^{|u|} f}{\partial y_u} (y_u; 0) \right|^2 \, dy_u
\]

\(2^s\) subsets

Mixed first derivatives are square integrable

Small weight \(\gamma_u\) means that \(f\) depends weakly on the variables \(y_u\)

Pair with randomly shifted lattice rules

Choose weights to minimize the error bound

\[
\left( \frac{2}{n} \sum_{u \subseteq \{1:s\}} \gamma_u^\lambda a_u \right)^{1/(2\lambda)} \left( \sum_{u \subseteq \{1:s\}} \frac{b_u}{\gamma_u} \right)^{1/2} \Rightarrow \gamma_u = \left( \frac{b_u}{a_u} \right)^{1/(1+\lambda)}
\]

Construct points (CBC) to minimize the worst case error
Setting 2: QMC integration over $\mathbb{R}^s$

- Change of variables
- Norm with weight function
- Pair with randomly shifted lattice rules
Setting 3: smooth integrands in the cube

- Norm involving **higher order mixed derivatives**
- Pair with **higher order digital nets**

- Interlaced polynomial lattice rule
Uniform v.s. lognormal

\[- \nabla \cdot (a(x, y) \nabla u(x, y)) = \kappa(x), \quad x \in D \subset \mathbb{R}^d, \ d = 1, 2, 3 \]

\[u(x, y) = 0, \quad x \in \partial D\]

**Uniform**

- \(a(x, y) = \overline{a}(x) + \sum_{j \geq 1} y_j \psi_j(x), \quad y_j \sim \mathcal{U}[-\frac{1}{2}, \frac{1}{2}]\)

- **Goal:**
  \[\int_{(-\frac{1}{2}, \frac{1}{2})^N} G(u(\cdot, y)) \, dy\]
  \(G\) – bounded linear functional

**Lognormal**

- \(a(x, y) = \exp(Z(x, y))\)

- **Karhunen-Loève expansion:**
  \(Z(x, y) = \overline{Z}(x) + \sum_{j \geq 1} y_j \sqrt{\mu_j} \xi_j(x), \quad y_j \sim \mathcal{N}(0, 1)\)

- **Goal:**
  \[\int_{\mathbb{R}^N} G(u(\cdot, y)) \prod_{j \geq 1} \phi_{\text{nor}}(y_j) \, dy\]

[Cohen, DeVore, Schwab (2011)]
Single-level v.s. multi-level (the uniform case)

\[ I_\infty = \int_{(-\frac{1}{2}, \frac{1}{2})^N} G(u(\cdot, y)) \, dy \]

**Single-level algorithm**

- **Deterministic:** 
  \[ \frac{1}{n} \sum_{i=1}^{n} G(u_h^s(\cdot, t_i - \frac{1}{2})) \]

- **Randomized:**

**Multi-level algorithm**

\[ I_\infty = (I_\infty - I_L) + \sum_{\ell=0}^{L} (I_\ell - I_{\ell-1}), \quad I_{-1} := 0 \]

**Deterministic:** 
\[ \sum_{\ell=0}^{L} \left( \frac{1}{n_{\ell}} \sum_{i=1}^{n_{\ell}} G((u_{h\ell}^{s_{\ell}} - u_{h\ell-1}^{s_{\ell-1}})(\cdot, t_{\ell,i} - \frac{1}{2})) \right) \]

**Randomized:**

*Benefits of randomization: unbiased estimator, practical error estimate*
Single-level v.s. multi-level \textit{(the lognormal case)}

\[ I_\infty = \int_{\mathbb{R}^N} G(u(\cdot, y)) \prod_{j \geq 1} \phi_{\text{nor}}(y_j) \, dy \]

**Single-level algorithm**

\begin{itemize}
  \item Deterministic: \[ \frac{1}{n} \sum_{i=1}^{n} G(u^s_h(\cdot, \Phi^{-1}(t_i))) \]
  \item Randomized: \\
\end{itemize}

**Multi-level algorithm**

\begin{itemize}
  \item Deterministic: \[ \sum_{\ell=0}^{L} \left( \frac{1}{n_\ell} \sum_{i=1}^{n_\ell} G((u^s_{h_{\ell}} - u^s_{h_{\ell-1}})(\cdot, \Phi^{-1}(t_{\ell,i}))) \right) \]
  \item Randomized: \\
\end{itemize}

*Benefits of randomization: unbiased estimator, practical error estimate*
Estimate the norm

**Single-level algorithm**

\[ \| G(u_h^s) \|_\gamma \]

**Multi-level algorithm**

\[ \| G'(u_{h\ell}^{s\ell} - u_{h\ell-1}^{s\ell-1}) \|_\gamma \]
TUTORIAL: Differentiate the PDE (the uniform case)

1. Variational formulation
\[ \int_D a(x, y) \nabla u(x, y) \cdot \nabla w(x) \, dx = \int_D \kappa(x) w(x) \, dx \quad \forall w \in H_0^1(D) \]

2. Mixed derivatives in \( y \) with multi-index \( \nu \neq 0 \)
\[ \int_D \partial^\nu \left( a(x, y) \nabla u(x, y) \cdot \nabla w(x) \right) \, dx = 0 \quad \forall w \in H_0^1(D) \]

3. Leibniz product rule
\[ \int_D \left( \sum_{m \leq \nu} \binom{\nu}{m} (\partial^m a)(x, y) \nabla (\partial^{\nu-m} u)(x, y) \cdot \nabla w(x) \right) \, dx = 0 \quad \forall w \in H_0^1(D) \]

4. Uniform case
\[ a(x, y) = \overline{a}(x) + \sum_{j \geq 1} y_j \psi_j(x) \implies (\partial^m a)(x, y) = \begin{cases} a(x, y) & \text{if } m = 0, \\ \psi_j(x) & \text{if } m = e_j, \\ 0 & \text{otherwise} \end{cases} \]

5. Separate out the \( m = 0 \) term
\[ \int_D a(x, y) \nabla (\partial^\nu u)(x, y) \cdot \nabla w(x) \, dx = - \sum_{j \geq 1} \nu_j \int_D \psi_j(x) \nabla (\partial^{\nu-e_j} u)(x, y) \cdot \nabla w(x) \, dx \quad \forall w \in H_0^1(D) \]
5. Separate out the \( m = 0 \) term

\[
\int_D a(x, y) \nabla (\partial^\nu u)(x, y) \cdot \nabla w(x) \, dx
\]

\[
= - \sum_{j \geq 1} \nu_j \int_D \psi_j(x) \nabla (\partial^\nu - e_j u)(x, y) \cdot \nabla w(x) \, dx \quad \forall w \in H^1_0(D)
\]

6. Take \( w = (\partial^\nu u)(\cdot, y) \) and apply Cauchy-Schwarz

\[
a_{\min} \int_D |\nabla (\partial^\nu u)(x, y)|^2 \, dx
\]

\[
\leq \sum_{j \geq 1} \nu_j \|\psi_j\|_{L_\infty} \left( \int_D |\nabla (\partial^\nu - e_j u)(x, y)|^2 \, dx \right)^{1/2} \left( \int_D |\nabla (\partial^\nu u)(x, y)|^2 \, dx \right)^{1/2}
\]

7. Cancel one common factor

\[
\|\nabla (\partial^\nu u)(\cdot, y)\|_{L_2} \leq \sum_{j \geq 1} \nu_j b_j \|\nabla (\partial^\nu - e_j u)(\cdot, y)\|_{L_2}, \quad b_j = \frac{\|\psi_j\|_{L_\infty}}{a_{\min}}
\]

8. Induction hypothesis

\[
\|\nabla (\partial^\nu u)(\cdot, y)\|_{L_2} \leq \sum_{j \geq 1} \nu_j b_j |\nu - e_j| ! b^{\nu - e_j} \frac{\|\kappa\|_{H^{-1}}}{a_{\min}} = |\nu| ! b^{\nu} \frac{\|\kappa\|_{H^{-1}}}{a_{\min}}
\]
8. **Bound on mixed derivatives by induction**

\[
\| (\partial^\nu u_h^s)(\cdot, y) \|_{H^1_0} = \| \nabla (\partial^\nu u_h^s)(\cdot, y) \|_{L^2} \leq |\nu|! b^\nu \frac{\|\kappa\|_{H^{-1}}}{} a_{\min}, \quad b_j := \frac{\|\psi_j\|_{L^\infty}}{a_{\min}}
\]

9. **First order mixed derivatives**

\[
\left\| \frac{\partial |u|}{\partial y_u} u_h^s(\cdot, y) \right\|_{H^1_0} \leq |u|! \left( \prod_{j \in u} b_j \right) \frac{\|\kappa\|_{H^{-1}}}{a_{\min}}, \quad u \subseteq \{1 : s\}
\]

10. **Linearity and boundedness of \(G\)**

\[
\left| \frac{\partial |u|}{\partial y_u} G(u_h^s(\cdot, y)) \right| = \left| G \left( \frac{\partial |u|}{\partial y_u} u_h^s(\cdot, y) \right) \right| \leq \|G\|_{H^{-1}} \left\| \frac{\partial |u|}{\partial y_u} u_h^s(\cdot, y) \right\|_{H^1_0}
\]

11. **Weighted norm**

\[
\|G(u_h^s)\|_{\gamma} \leq \frac{\|\kappa\|_{H^{-1}} \|G\|_{H^{-1}}}{a_{\min}} \left( \sum_{u \subseteq \{1 : s\}} \frac{(|u|!)^2 \prod_{j \in u} b_j^2}{\gamma_u} \right)^{1/2}
\]
11. Weighted norm

\[ \|G(u_h^s)\|_\gamma \lesssim \left( \sum_{u \subseteq \{1:s\}} \frac{(|u|)!^2 \prod_{j \in u} b_j^2}{\gamma_u} \right)^{1/2}, \quad b_j := \frac{\|\psi_j\|_{L_\infty}}{a_{\min}} \]

12. CBC error bound × bound on norm

ms-error \lesssim \left( \frac{2}{n} \sum_{u \subseteq \{1:s\}} \gamma_u^{\lambda} [\rho(\lambda)]^{|u|} \right)^{1/\lambda} \left( \sum_{u \subseteq \{1:s\}} \frac{(|u|)!^2 \prod_{j \in u} b_j^2}{\gamma_u} \right) \quad \forall \lambda \in (\frac{1}{2}, 1]

13. Minimize the bound to yield POD weights

\[ \gamma_u = \left( |u|! \prod_{j \in u} b_j \right)^{2/(1+\lambda)} \]

14. Choose \( \lambda \) to ensure dimension independence

- If \( \sum_{j \geq 1} b_j^p < \infty \), take \( \lambda = \begin{cases} \frac{1}{2 - 2 \delta} & \text{for } \delta \in (0, \frac{1}{2}) \text{ when } p \in (0, \frac{2}{3}], \\ \frac{p}{2 - p} & \text{when } p \in (\frac{2}{3}, 1). \end{cases} \)

- Convergence rate is \( n - \min\left(\frac{1}{p} - \frac{1}{2}, 1 - \delta\right) \)
- Implied constant is independent of \( s \)
Estimate the norm (the uniform case)

**Single-level algorithm**
\[ \|G(u_s^h)\|_\gamma \]

- **Lemma 1**
\[ \|\nabla \partial^\nu u(\cdot, y)\|_{L_2} \leq |\nu|! b^\nu \frac{\|\kappa\|_{H^{-1}}}{a_{\min}} \]

- **Lemma 2**
\[ \|\Delta \partial^\nu u(\cdot, y)\|_{L_2} \]

- **Lemma 3**
\[ \|\nabla \partial^\nu (u - u_h)(\cdot, y)\|_{L_2} \]

- **Lemma 4**
\[ |\partial^\nu G((u - u_h)(\cdot, y))| \]

**Multi-level algorithm**
\[ \|G(u_{s_\ell}^h - u_{s_{\ell-1}}^h)\|_\gamma \]

- **Lemma 2**
\[ \|\Delta \partial^\nu u(\cdot, y)\|_{L_2} \]

- **Lemma 3**
\[ \|\nabla \partial^\nu (u - u_h)(\cdot, y)\|_{L_2} \]

- **Lemma 4**
\[ |\partial^\nu G((u - u_h)(\cdot, y))| \]

Bounds of the form \((|\nu| + a_1)! \bar{b}^\nu\)

- **Lemma 1**
\[ b_j := \frac{\|\psi_j\|_{L_\infty}}{a_{\min}} \]

- **Lemma 2**
\[ \bar{b}_j := \frac{\max(\|\psi_j\|_{L_\infty}, \|\nabla \psi_j\|_{L_\infty})}{a_{\min}} \]

- **Lemma 3**
\[ O(h) \]

- **Lemma 4**
\[ O(h^2) \text{ by duality argument} \]
Estimate the norm (the lognormal case)

Single-level algorithm
\[ \| G(u^s_h) \| \gamma \]

- Lemma 5

Multi-level algorithm
\[ \| G(u'^{s\ell}_{h}\ell - u^{s\ell-1}_{h\ell-1}) \| \gamma \]

- Lemma 6
- Lemma 7
- Lemma 8

Bounds of the form \( T(y) (|\nu| + a_1)! \beta^\nu \)
Assumptions:
\[ \kappa \in H^{-1+t} \]
\[ G \in H^{-1+t'} \]
\[ \sum_{j \geq 1} \| \psi_j \|_{L^\infty}^{p_0} < \infty \]
\[ \sum_{j \geq 1} \| \nabla \psi_j \|_{L^\infty}^{p_1} < \infty \]
\[ \sum_{j \geq 1} \| \psi_j \|_{x_t}^{p_t} < \infty \]

**First-order single-level**
[K., Schwab, Sloan (2012)]
\[ s^{-2(\frac{1}{p_0} - 1)} + h^{t+t'} + \]
\[ n - \min(\frac{1}{p_0} - \frac{1}{2}, 1-\delta) \]

**First-order multi-level**
[K., Schwab, Sloan (2015)]
\[ s_L^{-2(\frac{1}{p_0} - 1)} + h_L^{t+t'} + \]
\[ \sum_{\ell=0}^{L} n_\ell^{-\min(\frac{1}{p_1} - \frac{1}{2}, 1-\delta)} \left( s_{\ell-1}^{-\left(\frac{1}{p_0} - \frac{1}{p_1}\right)} + h_{\ell-1}^{t+t'} \right) \]

**Higher-order single-level**
[Dick, K., Le Gia, Nuyens, Schwab (2014)]
\[ s^{-2(\frac{1}{p_0} - 1)} + h^{t+t'} + \]
\[ n - \frac{1}{p_0} \]

**Higher-order multi-level**
[Dick, K., Le Gia, Schwab (2016)]
\[ s_L^{-2(\frac{1}{p_0} - 1)} + h_L^{t+t'} + \]
\[ \sum_{\ell=0}^{L} n_\ell^{-\frac{1}{p_t}} \left( s_{\ell-1}^{-\left(\frac{1}{p_0} - \frac{1}{p_t}\right)} + h_{\ell-1}^{t+t'} \right) \]

**HO-ML does not always win!**
High-level view of analysis (the uniform case)

- **QMC** (deterministic or randomized) **convergence** (rate and constant) is independent of truncation dimension
  - Achieved by working in the **weighted** function space setting
  - Need to estimate the regularity of the PDE solution with respect to \( y \)
  - Function space weights are chosen to reduce the error bound
  - The chosen weights enter the fast CBC construction of QMC points

- **Combined error** – *truncation error, discretization error, quadrature error*
  - **SL**: sum of three errors
  - **ML**: multiplicative effect between FE and QMC, but need stronger regularity in \( x \), and regularity estimate simultaneously in \( x \) and \( y \)
  - **FO**: randomly shifted lattice rules; root-mean-square error bound; Hilbert space setting
  - **HO**: interlaced polynomial lattice rules; deterministic error bound; non-Hilbert space setting (gain an extra factor of \( n^{-1/2} \))

- **Error v.s. cost analysis** ... *depends crucially on the cost assumption*
  - **SL**: choose \( n, s, h \) to balance the three errors
  - **ML**: choose \( n_\ell, s_\ell, h_\ell \) to minimize total cost for a fixed total error (Lagrange multipliers)
**MLQMC numerical results** *(the lognormal case)*

- \( D = (0, 1)^2, \kappa \equiv 1, \) homogeneous Dirichlet boundary
- \( G(u(\cdot, y)) = \frac{1}{D^*} \int_{D^*} u(x, y) \, dx, \quad D^* = (\frac{3}{4}, \frac{7}{8}) \times (\frac{7}{8}, 1) \)
- Piecewise linear finite elements, off-the-shelf randomly shifted lattice rule
- Lognormal case with Matérn kernel: smoothness \( \nu \), variance \( \sigma^2 \), correlation length scale \( \lambda \)

\[
\nu = 2.5, \ \sigma^2 = 0.25, \ \lambda = 1
\]

\[
\nu = 1.5, \ \sigma^2 = 1, \ \lambda = 0.1
\]

[K., Scheichl, Schwab, Sloan, Ullmann (to appear)]
Software


Construction script in Python

- **Input:** generic bound on mixed derivatives (weights are set automatically)

\[
|\partial^\nu F(y)| \lesssim ((|\nu| + a_1)!)^{d_1} \prod_{j=1}^{s} (a_2 B_j)^{\nu_j} \exp(a_3 B_j |y_j|)
\]

\[
F(y) = \begin{cases} 
G(u_h^s) & \text{single-level} \\
G(u_{h_\ell}^s - u_{h_{\ell-1}}^s) & \text{multi-level}
\end{cases}
\]

- **Output:** generating vector for lattice sequence \texttt{z.txt}

```
# uniform case, 100-dim rule, 2^{10} points, with specified bounds b:
./lat-cbc.py --s=100 --m=10 --d2=3 --b="0.1 * j**-3 / log(j+1)"
```

```
# lognormal case, 100-dim rule, 2^{10} points, with algebraic decay:
./lat-cbc.py --s=100 --m=10 --a2="1/log(2)" --a3=1 --d2=3 --c=0.1
```

- **Output:** generating matrices for interlaced polynomial lattice rule \texttt{Bs53.col}

```
# 100-dim rule, 2^{10} points, interlacing 3, with bounds from file:
./polylat-cbc.py --s=100 --m=10 --alpha=3 --a1=5 --b_file=bounds.txt
```
Software


Point generators in Matlab/Octave (also C++ and Python)

- Lattice sequences

  ```matlab
  load z.txt                       % load generating vector
  latticeseq_b2('init0', z)       % initialize the procedural generator
  Pa = latticeseq_b2(20, 512);    % first 512 20-dimensional points
  Pb = latticeseq_b2(20, 512);    % next 512 20-dimensional points
  ```

- Interlaced polynomial lattice rules

  ```matlab
  load Bs53.col                    % load generating matrices
  digitalseq_b2g('init0', Bs53)    % initialize the procedural generator
  Pa = digitalseq_b2g(100, 512);   % first 512 100-dimensional points
  Pb = digitalseq_b2g(100, 512);   % next 512 100-dimensional points
  ```

- Interlaced Sobol’ sequences

  ```matlab
  load sobol_alpha3 Bs53.col        % load generating matrices
  digitalseq_b2g('init0', sobol_alpha3 Bs53) % initialize the procedural generator
  Pa = digitalseq_b2g(50, 512);     % first 512 50-dimensional points
  Pb = digitalseq_b2g(50, 512);     % next 512 50-dimensional points
  ```
Concluding remarks

- Three weighted space settings, **fast CBC construction**
- QMC error bound – **rate and constant** – can be **independent of dimension**
- Application of QMC theory: **estimate the norm, choose the weights**
- Pairing of method and setting: **best rate with minimal assumption**
- Theory versus practice: **generic QMC may work just fine!**
- Other cost reduction strategies: multi-index, MDM, ...

My wish list:
- Sharper dimension truncation estimate for ML?
- Weaker regularity requirement for higher order ML?
- Error as a triple product?
- Product weights instead of POD weights or SPOD weights?
- Better choice of weight function for lognormal?
- Higher order QMC for lognormal (integration over \(\mathbb{R}^s\))?
Thanks to my collaborators

<table>
<thead>
<tr>
<th>AU</th>
<th>Austria</th>
<th>Belgium</th>
<th>Ecuador</th>
<th>Germany</th>
<th>NZ</th>
<th>Poland</th>
<th>Switzerland</th>
<th>UK</th>
<th>USA</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="AU" /></td>
<td><img src="image2.png" alt="Austria" /></td>
<td><img src="image3.png" alt="Belgium" /></td>
<td><img src="image4.png" alt="Ecuador" /></td>
<td><img src="image5.png" alt="Germany" /></td>
<td><img src="image6.png" alt="NZ" /></td>
<td><img src="image7.png" alt="Poland" /></td>
<td><img src="image8.png" alt="Switzerland" /></td>
<td><img src="image9.png" alt="UK" /></td>
<td><img src="image10.png" alt="USA" /></td>
</tr>
</tbody>
</table>

Frances Kuo @ UNSW Australia