Application of Quasi-Monte Carlo Methods to Elliptic PDEs with Random Diffusion Coefficients: A Survey of Analysis and Implementation

Frances Y. Kuo · Dirk Nuyens

Received: 23 November 2015 / Revised: 18 June 2016 / Accepted: 22 June 2016 / Published online: 15 September 2016
© SFoCM 2016

Abstract This article provides a survey of recent research efforts on the application of quasi-Monte Carlo (QMC) methods to elliptic partial differential equations (PDEs) with random diffusion coefficients. It considers and contrasts the uniform case versus the lognormal case, single-level algorithms versus multi-level algorithms, first-order QMC rules versus higher-order QMC rules, and deterministic QMC methods versus randomized QMC methods. It gives a summary of the error analysis and proof techniques in a unified view, and provides a practical guide to the software for constructing and generating QMC points tailored to the PDE problems. The analysis for the uniform case can be generalized to cover a range of affine parametric operator equations.

Keywords Quasi-Monte Carlo methods · Infinite-dimensional integration · Partial differential equations with random coefficients · Uniform · Lognormal · Single-level · Multi-level · First order · Higher order · Deterministic · Randomized

Mathematics Subject Classification 65D30 · 65D32 · 65N30

Communicated by Albert Cohen.

Frances Y. Kuo
f.kuo@unsw.edu.au

Dirk Nuyens
dirk.nuyens@cs.kuleuven.be

1 School of Mathematics and Statistics, University of New South Wales, Sydney, NSW 2052, Australia
2 Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium
1 Introduction

In this article we provide a survey of recent research efforts on the application of quasi-Monte Carlo (QMC) methods to elliptic partial differential equations (PDEs) with random diffusion coefficients. Such PDE problems occur in the area of uncertainty quantification. In recent years many papers have been written on this topic, using a variety of methods (see below). QMC methods are relatively new to this application area. This article considers and contrasts different models for the randomness (uniform versus lognormal) and different algorithms (single-level versus multi-level, first-order versus higher-order, deterministic versus randomized). It gives a summary of the QMC error analysis and proof techniques in a unified view, and provides a practical guide to the software for constructing and generating QMC points tailored to the PDE problems. The analysis for the uniform case can be generalized to cover a range of affine parametric operator equations.

1.1 Motivating Example

Many physical, biological or geological models involve spatially varying input data which may be subject to uncertainty. This induces a corresponding uncertainty in the outputs of the model and in any physical quantities of interest which may be derived from these outputs. A common way to deal with these uncertainties is by considering the input data to be a random field, in which case the derived quantity of interest will in general also be a random variable or a random field. The computational goal is usually to find the expected value, higher-order moments or other statistics of these derived quantities.

A prime example is the flow of water through a disordered porous medium. Because of the near impossibility of modeling the microscopic channels through which water can flow in a porous layer, it is common engineering practice to model the porous medium as a random field. Mathematically, the flow can be modeled by Darcy’s law coupled with the mass conservation law, i.e.,

\[ q(x, \omega) + a(x, \omega) \nabla p(x, \omega) = 0, \]
\[ \nabla \cdot q(x, \omega) = 0, \]

for \( x \) in a bounded domain \( D \subset \mathbb{R}^d, \ d \in \{1, 2, 3\} \), and for almost all events \( \omega \) in the probability space \( (\Omega, \mathcal{A}, P) \). Here \( q(x, \omega) \) is the velocity (also called the specific discharge) and \( p(x, \omega) \) is the residual pressure, while \( a(x, \omega) \) is the permeability (or more precisely, the ratio of permeability to dynamic viscosity) which is modeled as a random field. Uncertainty in \( a(x, \omega) \) leads to uncertainty in \( q(x, \omega) \) and \( p(x, \omega) \). Quantities of interest include for example the breakthrough time of a plume of pollution moving through the medium.

To compute the expected value of some quantity of interest, one can generate a number of realizations of the random permeability field, for each realization solve the PDE numerically and compute the quantity of interest, and then take the average of all solutions from different realizations. This describes Monte Carlo (MC) simulation,
and it is regularly employed in such problems; see, e.g., [18,73–75,90,108]. By definition, expected values are integrals, with the dimensionality as high as the number of parameters needed to describe the randomness. This leads to the consideration of quasi-Monte Carlo (QMC) methods, which are quadrature methods for tackling high-dimensional integrals, with the hope to improve upon the slow convergence of MC simulation.

Throughout this article we refer to the number of integration variables $s$ as the “stochastic dimension,” which can be in the hundreds or thousands or more, in contrast to the “spatial dimension” $d$ which is just 1, 2 or 3.

1.2 The QMC Story

QMC methods [77,98], including the families of “lattice rules” and “digital nets,” are equal-weight quadrature rules where the quadrature points are chosen deterministically and designed cleverly to beat the random sampling of MC. They date back to the 1960s from number theorists, but the theories of that era were not adequate for very high-dimensional problems because either that fast convergence was obtained by assuming periodicity which is unrealistic in high dimensions, or that the typical error bounds for a QMC method requiring $n$ function evaluations in $s$ variables were of the form $c_s (\log n)^{s-1}/n$. In the latter case although the convergence rate appears faster than the MC rate of $1/\sqrt{n}$, the fatal flaw is that for fixed $s$, the function $(\log n)^{s-1}/n$ increases with increasing $n$ until $n \approx \exp(s)$, a number that is truly astronomical if $s$ is large.

So until perhaps the middle of the 1990s it was generally thought that QMC methods would not be effective in dimensions of more than say 20 or 30. But then a dramatically successful computational experiment of treating a 360-dimensional integral coming from Wall Street [88] changed people’s perceptions of what might be possible. At that time there was nothing in the available QMC theory that could explain the success. This led to many theoretical developments, as researchers struggled to understand how such high dimensionality could be handled successfully by QMC methods.

Modern QMC analysis takes into account that integrands from practical applications can have “low effective dimension” [9], meaning that the problem although having a very high nominal dimension may in fact depend mainly on a number of leading variables, or may be mainly affected by the interaction of a small number of variables at a time. This concept was formalized in theory through the introduction of “weighted” function spaces [102]: a set of parameters called “weights” are built into the function space norm to model the relative importance between different subsets of variables. Then “tractability” [81–83] analysis has been conducted in these settings to obtain, for example, a necessary and sufficient condition on the weights under which the integration error in the so-called “worst-case” sense is independent of the dimension $s$. Thus, rather than saying that all high-dimensional problems can be successfully tackled by QMC methods, we now know how to recognize and analyze mathematically the particular features that make some high-dimensional problems manageable.

Following the introduction of weighted spaces, a new class of constructive methods known as the “component-by-component (CBC) construction” has flourished [99, 100]. The algorithm constructs the components of the “generating vector” for lattice
rules one at a time: the \((j+1)\)th component is obtained by successive one-dimensional searches, with the previous \(j\) components kept unchanged. The inductive nature of such algorithms provides the means to obtain new methods for arbitrarily high dimensions. It has been established that this “greedy” algorithm yields lattice rules which achieve the optimal rate of convergence close to order \(n^{-1}\) in the underlying weighted function spaces, with the implied constant independent of \(s\); see [19,65]. This was followed by many further works, most notably the use of fast Fourier transform (FFT) to speed up the computation [85,86], the construction of “extensible lattice sequences” [17,32,57], the use of “tent transform” to achieve close to order \(n^{-2}\) convergence [29,56], the carrying over of the lattice technology to digital nets and sequences [26,30,87] and the revolutionary invention of “higher-order digital nets” which allow a convergence rate of order \(n^{-\alpha}, \alpha > 1\), for sufficiently smooth integrands [3–5,20–22,45]. For surveys of these recent QMC developments see [27,31,67,84,97].

By now weights of many forms have been considered in the literature. At the unrealistic extreme we have “general weights” which allow for a different weight parameter to be attached to each of the \(2^s\) subsets of the indices from 1 to \(s\). The original and simplest weights from [102] are now called “product weights,” and in between we have “finite-order” weights, “finite diameter” weights and “order-dependent” weights [33,101]. Furthermore, there are the more recent additions in the context of applying QMC to PDE problems: “POD” (for “product and order-dependent”) weights [68] and “SPOD” (for “smoothness-driven product and order-dependent”) weights [23]. The driving motivation for this flowering of possibilities has been the desire to describe in a more precise way the influence of particular combinations of the variables. For the CBC construction of lattice rules mentioned above, the weights definitely matter, since they appear as parameters in the algorithm that determine the integration rule. The philosophy is therefore to choose the weights according to the dimension structure of the practical integrands and then construct QMC methods that are tailored to the given application.

1.3 Progress on PDEs with Random Coefficients

Returning to the motivating example in Sect. 1.1, because the permeability \(a(\mathbf{x}, \omega)\) is physically positive, it is popular and natural to assume that \(a(\mathbf{x}, \omega)\) is a “lognormal” random field, that is, \(\log(a(\mathbf{x}, \omega))\) is a Gaussian random field on the spatial domain \(D\) with a specified mean and a covariance function. There is some evidence from field data that lognormality gives a reasonable representation of reality in certain cases [35,61]. See below for many recent references which considered lognormal random fields.

A common approach to represent the random field \(a(\mathbf{x}, \omega)\) is to use the Karhunen–Loève (KL) expansion [72] to write \(\log(a(\mathbf{x}, \omega))\) as an infinite series involving the eigenvalues and eigenfunctions of the integral operator associated with the covariance function, where the series is parametrised and depends linearly on a sequence \(y_j = y_j(\omega), j \geq 1\), of i.i.d. standard normal random numbers from \(\mathbb{R}\). In practical computations the infinite sum is truncated to, say, \(s\) terms, giving rise to a truncation error to be managed. While this approach can be very effective when the KL expan-
sion converges rapidly, it faces the serious challenges of high cost combined with large truncation error when the convergence of the KL expansion is slow.

Instead of sampling the continuous random field everywhere in the spatial domain by a truncated KL expansion, an alternative approach is to sample the random field only at a discrete set of grid points with respect to the covariance matrix inherited from the given covariance function of the continuous field. The random field is then represented exactly at these grid points, thus eliminating completely the truncation error. (However, interpolation would be required at the time of assembling the stiffness matrix for solving the PDE numerically, and care is needed to ensure that the interpolation error is no worse than, e.g., the quadrature error in integrating the finite element basis functions.) The resulting large matrix factorization problem can be handled by the “circulant embedding” technique; see, e.g., [34,47,66].

The unbounded parameters \( y = (y_j)_{j \geq 1} \) from the lognormal model present some challenges in the theoretical analysis. One major challenge is that the random coefficient is not uniformly bounded from above and below and so the Lax–Milgram lemma cannot be applied directly. Many researchers therefore consider the simpler “uniform” model where \( a(\mathbf{x}, \omega) \) is written as an infinite series that depends linearly on a sequence \( y_j = y_j(\omega), j \geq 1, \) of i.i.d. uniform random numbers from a bounded interval \( [-1, 1] \) or \( [-\frac{1}{2}, \frac{1}{2}] \).

There is a huge body of the literature on treating these PDEs with random coefficients. Some methods apply simultaneous approximation in both physical and probability space; these go under names such as “stochastic Galerkin,” “stochastic collation,” “polynomial chaos” or “generalized polynomial chaos”; see, e.g., [1,2,13–15,38,49,58,59,64,79,80,94–96,106]. In recent times these methods are also combined with “multi-level” schemes to reduce the computational cost without loss of accuracy; see, e.g., [6,11,12,16,51,52,103,104]. The methods are also being applied to the area of Bayesian inversion; see, e.g., [60,91,92].

Table 1 provides a summary of some recent progress on the application of QMC to PDEs with random coefficients. Firstly, comprehensive numerical experiments were carried out in [47] showing promising results for the lognormal case with the circulant embedding strategy, but without any theoretical justification. The first theoretical analysis was done in [68] for the simpler uniform case under the KL expansion framework. The analysis was then generalized to the lognormal case in [46] and extended to a multi-level scheme for the uniform case in [69]. The use of first-order QMC methods in the uniform case was then replaced by higher-order QMC methods in [23], and the corresponding multi-level analysis was done in [25]. The multi-level analysis for the lognormal case was done in [66], while the analysis for the discrete sampling method combined with circulant embedding technique is being considered in [48]. There are also other QMC-related works; see, e.g., [28,36,37,53,71,89,93].

1.4 Overview of This Article

This article surveys the results from [23,25,46,66,68,69] in a unified view. The first-order results [46,66,68,69] are based on randomly shifted lattice rules and are accompanied by probabilistic error bounds. The higher-order results [23,25] are based
Table 1  Application of QMC to PDEs with random coefficients

|                        | Uniform KL expansion | Lognormal KL expansion | Circulant embedding |
|------------------------|----------------------|------------------------|---------------------|
| Numerical experiments only |                      |                        | [47]*               |
| First-order single-level analysis | [68]                | [46]*                  | [48]*               |
| First-order multi-level analysis | [69]                | [66]*                  |                     |
| Higher-order single-level analysis | [23]                |                        |                     |
| Higher-order multi-level analysis | [25]*               |                        |                     |

The * indicates that there are accompanying numerical results on interlaced polynomial lattice rules and are accompanied by deterministic error bounds. Part of the gain in the improved convergence rates arises because of the switch from an $\ell^2$ norm in the function space setting to an $\ell^\infty$ norm. The lognormal results [46,66] require a non-standard function space setting for integrands with domain $\mathbb{R}^{s}$ for some $s$. There is as yet no QMC theory that can give higher-order convergence for the lognormal case.

We will not discuss the theory [48] for the case of circulant embedding. Also, the results in [23,25] were formulated for general affine parametric operator equations, but we will only touch on this aspect very briefly in the article.

This article includes a practical guide on how to make use of the QMC technology for this and other PDE problems. Computer programs are available from the web site http://people.cs.kuleuven.be/~dirk.nuyens/qmc4pde/.

The structure of this article is as follows. In Sect. 2 we give a parametric formulation of the PDE problem and discuss both the uniform and lognormal cases. In Sect. 3 we provide some minimum background on QMC and finite element (FE) methods as well as the dimension truncation analysis. In Sect. 4 we introduce the single-level and multi-level algorithms for both the deterministic and randomized variants. In Sect. 5 we introduce three weighted function space settings for QMC error analysis. In Sect. 6 we summarize the error analysis from [23,25,46,66,68,69] in a high-level unified way. We explain the strategies and proof techniques, deferring proofs to Sect. 9, the Appendix. In Sect. 7 we provide a practical guide on how to use the software from the Web site for constructing and generating QMC points. In Sect. 8 we end the article with some concluding remarks.

2 Uniform Versus Lognormal Coefficients

With the motivating example from Sect. 1.1 in mind, we consider throughout this article a model parametric elliptic problem with homogeneous Dirichlet condition. This model problem has been considered in many papers. It is simple enough for illustrating the kind of results that we can obtain and the corresponding proof techniques. The strategy can be extended to more general domains and boundary conditions as well as other PDE problems.
We take the view that the random coefficient \( a(\mathbf{x}, \omega) \) has been parameterized by a vector \( \mathbf{y}(\omega) = (y_1(\omega), y_2(\omega), \ldots) \), and, for fixed \( \omega \), we denote the corresponding deterministic parametric coefficient by \( a(\mathbf{x}, \mathbf{y}) \). Specifically, for a given parameter \( \mathbf{y} \) we consider the parametric elliptic Dirichlet problem

\[
-\nabla \cdot (a(\mathbf{x}, \mathbf{y}) \nabla u(\mathbf{x}, \mathbf{y})) = f(\mathbf{x}) \quad \text{for} \quad \mathbf{x} \in D, \quad u(\mathbf{x}, \mathbf{y}) = 0 \quad \text{for} \quad \mathbf{x} \text{ on } \partial D, \tag{2.1}
\]

for domain \( D \subset \mathbb{R}^d \) a bounded, convex, Lipschitz polyhedron with boundary \( \partial D \), where the spatial dimension \( d = 1, 2, \) or \( 3 \) is assumed given and fixed. The differential operators in (2.1) are understood to be with respect to the physical variable \( \mathbf{y} \) which belongs to \( D \). The parametric variable \( \mathbf{y} = (y_j)_{j \geq 1} \) belongs to either a bounded or unbounded domain, depending on which of the two popular formulations of the parametric coefficient \( a(\mathbf{x}, \mathbf{y}) \) is being considered: the “uniform” case or the “lognormal” case; see below.

### 2.1 Uniform Case

In the “uniform” case, we assume that the parameter \( \mathbf{y} \) is distributed on

\[
U := \left[ -\frac{1}{2}, \frac{1}{2} \right]^N
\]

with the uniform probability measure \( \mu(\mathrm{d}\mathbf{y}) = \otimes_{j \geq 1} \mathrm{d}y_j = \mathrm{d}\mathbf{y} \). Throughout the article \( \mathbb{N} = \{1, 2, 3, \ldots\} \) denotes the set of positive integers. The parametric coefficient \( a(\mathbf{x}, \mathbf{y}) \) is further assumed to depend linearly on the parameters \( y_j \) as follows:

\[
a(\mathbf{x}, \mathbf{y}) = a_0(\mathbf{x}) + \sum_{j \geq 1} y_j \psi_j(\mathbf{x}), \quad \mathbf{x} \in D, \quad \mathbf{y} \in U. \tag{2.2}
\]

The functions \( \psi_j \) can arise from either the eigensystem of a covariance operator or other suitable function systems in \( L^2(D) \).

We will impose a number of assumptions on \( a_0 \) and \( \psi_j \) as required. In the following, the \( L^\infty(D) \) norm is defined as the essential supremum in \( D \) as per usual, and for \( \|\nabla v\|_{L^\infty} \) we take the essential supremum of the Euclidean norm of \( \nabla v \). The \( W^{1,\infty}(D) \) norm is defined by \( \|v\|_{W^{1,\infty}} := \max\{\|v\|_{L^\infty}, \|\nabla v\|_{L^\infty}\} \).

(U1) We have \( a_0 \in L^\infty(D) \) and \( \sum_{j \geq 1} \|\psi_j\|_{L^\infty} < \infty \).

(U2) There exist \( a_{\text{max}} \) and \( a_{\text{min}} \) such that \( 0 < a_{\text{min}} \leq a(\mathbf{x}, \mathbf{y}) \leq a_{\text{max}} < \infty \) for all \( \mathbf{x} \in D \) and \( \mathbf{y} \in U \).

(U3) We have \( a_0 \in W^{1,\infty}(D) \) and \( \sum_{j \geq 1} \|\psi_j\|_{W^{1,\infty}} < \infty \).

(U4) The sequence \( \psi_j \) is ordered so that \( \|\psi_1\|_{L^\infty} \geq \|\psi_2\|_{L^\infty} \geq \cdots \).

(U5) There exists \( p_0 \in (0, 1) \) such that \( \sum_{j \geq 1} \|\psi_j\|^{p_0} \|_{L^\infty} < \infty \).

(U6) There exists \( p_1 \in (0, 1) \) such that \( \sum_{j \geq 1} \|\psi_j\| \|_{W^{1,\infty}} < \infty \).

(U7) For a non-negative integer \( t \), there exists \( p_t \in (0, 1) \) such that \( \sum_{j \geq 1} \|\psi_j\|^{p_t} \| \_{\mathcal{C}_t^t} < \infty \), where \( \mathcal{C}_t^t \) denotes a Sobolev space of functions on \( D \) with smoothness scale \( t \), with \( t \) roughly corresponding to the number of derivatives that exist in \( \mathbf{x} \).
Assumption (U1) ensures that the coefficient \( a(x, y) \) is well defined for all parameters \( y \in U \). Assumption (U2) yields the continuity and coercivity needed for the standard FE analysis, so that a unique solution exists. Assumption (U3) guarantees that the FE solutions converge to the solution of (2.1). Assumption (U4) enables the analysis for truncating the infinite sum in (2.2). Assumption (U5) implies decay of the fluctuation coefficients \( \psi_j \), with faster decay for smaller \( p_0 \); Assumptions (U6) and (U7) have similar implications. Although not explicitly specified, we are interested in those values of \( p_0, p_1 \) and \( p_t \) which are as small as possible. Typically we have \( p_0 < p_1 < p_2 < \cdots \). The values of \( p_0, p_1 \) and \( p_t \) will determine our QMC convergence rates for different algorithms.

For convenience of later analysis, we define the sequence \( b = (b_j)_{j \geq 1} \) by

\[
 b_j := \frac{\| \psi_j \|_{L^\infty}}{a_{\min}}, \quad j \geq 1, \tag{2.3}
\]

and define the sequence \( \overline{b} = (\overline{b}_j)_{j \geq 1} \) by

\[
 \overline{b}_j := \frac{\| \psi_j \|_{W^{1,\infty}}}{a_{\min}} = \frac{\max(\| \psi_j \|_{L^\infty}, \| \nabla \psi_j \|_{L^\infty})}{a_{\min}} \geq b_j, \quad j \geq 1. \tag{2.4}
\]

Our goal is to compute the integral, i.e., the expected value, with respect to \( y \in U \), of a bounded linear functional \( G \) applied to the solution \( u(\cdot, y) \) of the PDE (2.1)

\[
 I(G(u)) = \int_{[-1/2, 1/2]^N} G(u(\cdot, y)) \, dy 
 = \lim_{s \to \infty} \int_{[-1/2, 1/2]^s} G(u(\cdot, (y_1, \ldots, y_s, 0, 0, \ldots))) \, dy_1 \cdots dy_s.	ag{2.5}
\]

We remark that our analysis relies heavily on the boundedness and linearity of \( G \), especially in the duality arguments.

The uniform framework can be extended to the general framework of “affine” parametric operator equations; see [93] as well as [23, 25]. Let \( \{A_j\}_{j \geq 1} \) denote a sequence of bounded linear operators in \( L(X, Y^*) \), between suitably defined spaces \( X \) and \( Y^* \). For every \( f \in Y^* \) and every \( y \in U \), the task is to seek \( u(y) \in X \) such that \( A(y) u(y) = f \), where \( A(y) = A_0 + \sum_{j \geq 1} y_j A_j \). In this general setting, coercivity is replaced by inf–sup conditions, and the results depend, e.g., on the summability of \( \sum_{j \geq 1} \| A_j \|_{L(X, Y^*)}^{p_0} \) for \( p_0 \in (0, 1) \). We will not discuss this general framework further in this article, other than to summarize some results from [23, 25] at the end of Sect. 6.

2.2 Lognormal Case

In the “lognormal” case, we assume that the parameter \( y \) is distributed on \( \mathbb{R}^N \) according to the product Gaussian measure \( \mu_G = \bigotimes_{j \geq 1} N(0, 1) \). The parametric coefficient \( a(x, y) \) now takes the form

\[
 a(x, y) = \bigotimes_{j \geq 1} \phi_j (x) \cdot \phi_j (y). \tag{2.6}
\]
\[ a(\mathbf{x}, \mathbf{y}) = a_0(\mathbf{x}) \exp \left( \sum_{j \geq 1} y_j \sqrt{\mu_j} \xi_j(\mathbf{x}) \right), \quad \mathbf{x} \in D, \quad \mathbf{y} \in \mathbb{R}^N, \]  

(2.6)

where \( a_0(\mathbf{x}) > 0 \).

The coefficient \( a(\mathbf{x}, \mathbf{y}) \) of the form (2.6) can arise from the Karhunen–Loève (KL) expansion in the case where \( \log(a) \) is a stationary Gaussian random field with a specified mean and a covariance function. As an example we focus on the isotropic Matérn covariance \( \rho_v(|\mathbf{x} - \mathbf{x}'|) \), with

\[ \rho_v(r) := \sigma^2 \frac{2^{1-v}}{\Gamma(v)} \left( \sqrt{2r} \frac{r}{\lambda_C} \right)^v K_v \left( \sqrt{2r} \frac{r}{\lambda_C} \right), \]  

(2.7)

where \( \Gamma \) is the gamma function and \( K_v \) is the modified Bessel function of the second kind. The parameter \( v > 1/2 \) is a smoothness parameter, \( \sigma^2 \) is the variance and \( \lambda_C \) is the correlation length scale. Then \( \{ (\mu_j, \xi_j) \}_{j \geq 1} \) is the sequence of eigenvalues and eigenfunctions of the integral operator \( (Rw)(\mathbf{x}) = \int_D \rho_v(|\mathbf{x} - \mathbf{x}'|) w(\mathbf{x}') \, d\mathbf{x}' \), with eigenvalues \( \mu_j \) enumerated in non-increasing order and with eigenfunctions \( \xi_j \) normalized in \( L^2(D) \). Moreover, the sequence \( \{ \xi_j \}_{j \geq 1} \) form an orthonormal basis in \( L^2(D) \).

We define the sequence \( \mathbf{\beta} = (\beta_j)_{j \geq 1} \) by

\[ \beta_j := \sqrt{\mu_j} \| \xi_j \|_{L^\infty}, \quad j \geq 1, \]  

(2.8)

and define the set of admissible parameters

\[ U_{\mathbf{\beta}} := \left\{ \mathbf{y} \in \mathbb{R}^N : \sum_{j \geq 1} \beta_j |y_j| < \infty \right\} \subseteq \mathbb{R}^N. \]

We also define the sequence \( \mathbf{\overline{\beta}} = (\overline{\beta}_j)_{j \geq 1} \) by

\[ \overline{\beta}_j := \sqrt{\mu_j} \| \xi_j \|_{W^{1,\infty}} = \sqrt{\mu_j} \max\{ \| \xi_j \|_{L^\infty}, \| \nabla \xi_j \|_{L^\infty} \} \geq \beta_j, \quad j \geq 1, \]  

(2.9)

and define analogously \( U_{\mathbf{\overline{\beta}}} \subseteq U_{\mathbf{\beta}} \subseteq \mathbb{R}^N \).

Similarly to the uniform case, we will impose a number of assumptions in the lognormal case as required. We follow the setting of [66], with the exception of Assumption (L4) which came from [46].

(L1) We have \( a_0 \in L^\infty(D) \) and \( \sum_{j \geq 1} \beta_j < \infty. \)

(L2) For every \( \mathbf{y} \in U_{\mathbf{\beta}} \), the expressions \( a_{\max}(\mathbf{y}) := \max_{\mathbf{x} \in D} a(\mathbf{x}, \mathbf{y}) \) and \( a_{\min}(\mathbf{y}) := \min_{\mathbf{x} \in D} a(\mathbf{x}, \mathbf{y}) \) are well defined and satisfy \( 0 < a_{\min}(\mathbf{y}) \leq a(\mathbf{x}, \mathbf{y}) \leq a_{\max}(\mathbf{y}) < \infty \).

(L3) We have \( a_0 \in W^{1,\infty}(D) \) and \( \sum_{j \geq 1} \overline{\beta}_j < \infty. \)

(L4) There exist \( C_1, C_2 > 0, \Theta > 1 \) and \( \varepsilon \in \left[ \Theta^{-1}, \frac{\Theta+1}{2\Theta} \right] \) such that \( \mu_j \leq C_1 j^{-\Theta} \) and \( \| \xi_j \|_{C^0(\overline{D})} + \mu_j \| \nabla \xi_j \|_{C^0(\overline{D})} \leq C_2 \mu_j^{-\varepsilon} \) for \( j \geq 1. \)
There exists \( p_0 \in (0, 1) \) such that \( \sum_{j \geq 1} \beta_j^{p_0} < \infty \).

There exists \( p_1 \in (0, 1) \) such that \( \sum_{j \geq 1} \beta_j^{p_1} < \infty \).

Assumption (L1) ensures that the series (2.6) converges in \( L^\infty(D) \) for every \( y \in U_\beta \) and that \( \mu_G(U_\beta) = 1 \); see [94, Lemma 2.28]. Moreover, Assumption (L1) implies Assumption (L2), which in turn yields the continuity and coercivity of the bilinear form (see (3.6) below) for every \( y \in U_\beta \). Assumption (L3) ensures that the series (2.6) converges in \( W^{1,\infty}(D) \) for every \( y \in U_\beta \) and that \( \mu_G(U_\beta) = 1 \); it also guarantees that for every \( y \in U_\beta \) the FE solutions converge to the solution of (2.1). Assumption (L4) is needed for the dimension truncation result. Assumptions (L5) and (L6) play analogous roles to Assumptions (U5) and (U6) in the uniform case. Typically we have \( p_0 < p_1 \).

Our goal is again to compute the integral of a bounded linear functional \( G \) applied to the solution \( u(\cdot,y) \) of the PDE, but now the integral is over \( y \in \mathbb{R}^N \) with a countable product Gaussian measure \( \mu_G(dy) \). Recall that we restrict ourselves to \( y \in U_\beta \) with full measure \( \mu_G(U_\beta) = 1 \). Abusing the standard notation in measure and integration theory, we write this integral simply as

\[
I(G(u)) = \int_{\mathbb{R}^N} G(u(\cdot,y)) \prod_{j \geq 1} \phi(y_j) \, dy = \int_{[0,1]^N} G(u(\cdot,\Phi^{-1}(w))) \, dw. \tag{2.10}
\]

Here \( \phi(y) := \exp(-y^2/2)/\sqrt{2\pi} \) is the univariate standard normal probability density function. Denoting the corresponding cumulative distribution function by \( \Phi(y) := \int_{-\infty}^y \phi(t) \, dt \) and its inverse by \( \Phi^{-1} \), we can apply the change of variables componentwise as follows:

\[
y = \Phi^{-1}(w) = (\Phi^{-1}(w_1), \Phi^{-1}(w_2), \ldots) \in \mathbb{R}^N \quad \text{for} \quad w \in (0, 1)^N.
\]

This leads to the transformed integral over the unit cube on the right-hand side of (2.10). The equivalence of the integrals in (2.10) follows from Kakutani’s theorem on the equivalence of infinite product measures (see, e.g., [7]).

3 Quadrature, Spatial Discretization, Dimension Truncation

3.1 QMC Quadrature

For \( F \) a general real-valued function defined over the \( s \)-dimensional unit cube \([0, 1]^s\), with \( s \) finite and fixed, we consider the integral

\[
I(F) = \int_{[0,1]^s} F(y) \, dy.
\]

An \( n \)-point quasi-Monte Carlo (QMC) approximation to this integral is an equal-weight quadrature rule of the form
\[ Q(F) = \frac{1}{n} \sum_{i=1}^{n} F(t_i), \quad (3.1) \]

with carefully chosen points \( t_1, \ldots, t_n \in [0, 1]^s \). See [27] for a comprehensive survey of recent developments on QMC methods. In this article we will consider two families of QMC methods: randomly shifted lattice rules and (deterministic) interlaced polynomial lattice rules. More details about these QMC methods and their error analysis will be given in Sect. 5. Here we provide only a brief overview of the general framework.

We define the worst-case error for QMC integration in some Banach space of functions \( \mathcal{H} \) to be

\[ e_{\text{wor}}(t_1, \ldots, t_n) := \sup_{F \in \mathcal{H}, \|F\|_{\mathcal{H}} \leq 1} |I(F) - Q(F)|. \]

Then we have the error bound

\[ |I(F) - Q(F)| \leq e_{\text{wor}}(t_1, \ldots, t_n) \|F\|_{\mathcal{H}} \quad \text{for all } F \in \mathcal{H}. \]

An error bound of this form conveniently separates the dependence on the QMC point set from the dependence on the integrand. For a given integrand \( F \), the general idea is to choose a suitable function space \( \mathcal{H} \) so that the norm \( \|F\|_{\mathcal{H}} \) is finite, and then to construct QMC points \( t_1, \ldots, t_n \) to make the worst-case error \( e_{\text{wor}}(t_1, \ldots, t_n) \) as small as possible. There could be a trade-off between these two quantities, but the ultimate goal is to make the product of the two quantities as small as possible.

The advantages of deterministic QMC methods include the exact reproducibility and the fully deterministic theoretical error bound; these properties might be favored by practitioners. However, one may also argue that deterministic QMC methods have the drawback of being biased and lacking a practical error estimate. In contrast, randomized QMC methods are unbiased and a practical error estimate can be easily obtained. Here we discuss only the simplest kind of randomization, namely random shifting. For a fixed shift \( \Delta \in [0, 1]^s \), the \( \Delta \)-shift of the QMC rule (3.1) is

\[ Q(F; \Delta) = \frac{1}{n} \sum_{i=1}^{n} F((t_i + \Delta)), \quad (3.2) \]

where the braces around a vector indicate that we take the fractional part of each component in the vector. Essentially, we move all the QMC points by the same amount, and if any point falls outside of the unit cube it is simply “wrapped” back in from the opposite side. If the shift \( \Delta \) is generated randomly from the uniform distribution on \([0, 1]^s\), then it is easy to verify that \( \mathbb{E}[Q(F; \cdot)] = \int_{[0,1]^s} Q(F; \Delta) \, d\Delta = I(F) \), that is, a randomly shifted QMC rule provides an unbiased estimate of the integral, and in turn, the variance of \( Q(F; \cdot) \) is precisely the mean-square error \( \mathbb{E}[(I(F) - Q(F; \cdot))^2] \).

A probabilistic error bound for a randomly shifted QMC rule in \( \mathcal{H} \) is

\[ \sqrt{\mathbb{E}[|I(F) - Q(F; \cdot)|^2]} \leq e_{\text{sh}}^{\text{wor}}(t_1, \ldots, t_n) \|F\|_{\mathcal{H}} \quad \text{for all } F \in \mathcal{H}, \quad (3.3) \]

where the quantity \( e_{\text{sh}}^{\text{wor}}(t_1, \ldots, t_n) := (\int_{[0,1]^s} (e_{\text{wor}}((t_1 + \Delta), \ldots, (t_n + \Delta))^2 \, d\Delta)^{1/2} \) is known as the shift-averaged worst-case error.
The idea is then to construct QMC points $t_1, \ldots, t_n$ to make the shift-averaged worst-case error as small as possible. In practice, we can take a number of independent random shifts $\Delta_1, \ldots, \Delta_r$ drawn from the uniform distribution on $[0, 1]^s$ and use the average

$$Q_{\text{ran}}(F) = \frac{1}{r} \sum_{k=1}^{r} Q(F; \Delta_k)$$

as the approximation to the integral. Notice our use of the subscript “ran” to denote that this is a randomized rule. Since $Q(F; \Delta_1), \ldots, Q(F; \Delta_r)$ are i.i.d. random variables with mean $I(F)$, the variance of their average $Q_{\text{ran}}(F)$ is precisely the variance of a single $Q(F; \Delta_k)$ divided by $r$. This together with (3.3) gives

$$\sqrt{E[|I(F) - Q_{\text{ran}}(F)|^2]} \leq r^{-1/2} e_{\text{wor}}^\text{sh}(t_1, \ldots, t_n) \|F\|_{\mathcal{H}} \quad \text{for all } F \in \mathcal{H}.$$ 

A practical estimate of the standard error can be obtained by calculating

$$\sqrt{\frac{1}{r(r-1)} \sum_{k=1}^{r} (Q(F; \Delta_k) - Q_{\text{ran}}(F))^2},$$

from which a confidence interval for $Q_{\text{ran}}(F)$ can be deduced.

A QMC approximation to an integral which is formulated over the Euclidean space $\mathbb{R}^s$ can be done by first mapping the integral to the unit cube as follows:

$$I(F) = \int_{[0,1]^s} F(\Phi^{-1}(w)) \, dw = \frac{1}{n} \sum_{i=1}^{n} F(\Phi^{-1}(t_i)) = Q(F).$$

Here $\phi$ can be any general univariate probability density function, and $\Phi^{-1}$ denotes the componentwise application of the inverse of the cumulative distribution function corresponding to $\phi$. Note that in many practical applications we need to first apply some clever transformation to formulate the integral in the above form; some examples are discussed in [76].

### 3.2 FE Discretization

In the variational setting, we consider the Sobolev space $V = H_0^1(D)$ of functions with vanishing trace on the boundary, with norm $\|v\|_V := \|\nabla v\|_{L^2}$, together with its dual space $V^* = H^{-1}(D)$ and pivot space $L^2(D)$. We now discuss the weak formulation of (2.1). For $f \in V^*$ and $y \in U$ (or $y \in U_\beta$ in the lognormal case), find $u(\cdot, y) \in V$ such that
\[ \mathcal{A}(y; u(\cdot, y), v) = \langle f, v \rangle \quad \text{for all} \quad v \in V, \quad (3.5) \]

where the parametric bilinear form is given by

\[ \mathcal{A}(y; w, v) := \int_D a(x, y) \nabla w(x) \cdot \nabla v(x) \, dx \quad \text{for all} \quad w, v \in V, \quad (3.6) \]

and \( \langle \cdot, \cdot \rangle \) denotes the duality pairing between \( V \) and \( V^* \).

In the uniform case under Assumptions (U1) and (U2), it follows that for all \( y \in U \) the bilinear form is continuous and coercive on \( V \times V \), and we may infer from the Lax–Milgram lemma that for every \( f \in V^* \) and every \( y \in U \) there exists a unique solution \( u(\cdot, y) \in V \) to (3.5) satisfying the standard \textit{a priori} estimate

\[ \|u(\cdot, y)\|_V \leq \frac{\|f\|_{V^*}}{a_{\min}}. \quad (3.7) \]

In addition, if Assumption (U3) holds and if we assume that the representer \( f \) of the right-hand side of (3.5) is in \( L^2(D) \), then we have

\[ \|\Delta u(\cdot, y)\|_{L^2} \lesssim \|f\|_{L^2}. \quad (3.8) \]

Throughout this article the notation \( P \lesssim Q \) indicates \( P \leq CQ \) for some constant \( C > 0 \) which is independent of all relevant parameters.

We denote by \( \{V_h\}_{h>0} \) a family of subspaces \( V_h \subset V \) of finite dimension \( M_h \). For example, \( V_h \) can be the space of continuous piecewise linear finite elements on a sequence of regular triangulations of \( D \) with meshwidth \( h > 0 \). We define the parametric finite element (FE) approximation as follows: for \( f \in V^* \) and \( y \in U \) (or \( y \in U_B \) in the lognormal case), find \( u_h(\cdot, y) \in V_h \) such that

\[ \mathcal{A}(y; u_h(\cdot, y), v_h) = \langle f, v_h \rangle \quad \text{for all} \quad v_h \in V_h. \]

Then, in the uniform case under Assumptions (U1) and (U2), it is known that the FE approximation \( u_h(\cdot, y) \) of \( u(\cdot, y) \) is stable, that is, (3.7) holds with \( u(\cdot, y) \) replaced by \( u_h(\cdot, y) \). Recall that \( G \in V^* \) is the linear functional considered in (2.5) and (2.10). In the same way as we did with \( f \in V^* \), we use the same notation to denote the representer of \( G \). In addition, if Assumption (U3) holds and \( f \in L^2(D) \) and \( G \in L^2(D) \), then as \( h \to 0 \) we have

\[ \|u(\cdot, y) - u_h(\cdot, y)\|_V \lesssim h \|\Delta u(\cdot, y)\|_{L^2} \lesssim h \|f\|_{L^2}, \quad (3.9) \]

\[ |G(u(\cdot, y)) - G(u_h(\cdot, y))| \lesssim h^2 \|f\|_{L^2} \|G\|_{L^2}, \quad (3.10) \]

\[ |I(G(u)) - I(G(u_h))| \lesssim h^2 \|f\|_{L^2} \|G\|_{L^2}. \quad (3.11) \]

In the lognormal case under Assumptions (L1) and (L2), the \textit{a priori} estimate (3.7) is replaced by
\[ \| u(\cdot, y) \|_V \leq \frac{\| f \|_{V^*}}{a_{\min}(y)} \text{ for all } y \in U_\beta. \]  

Adding also Assumption (L3) and \( f \in L^2(D) \), the bound (3.8) is replaced by

\[ \| \Delta u(\cdot, y) \|_{L^2} \lesssim T(y) \| f \|_{L^2} \text{ for all } y \in U_\beta, \]

where

\[ T(y) := \frac{\| \nabla a(\cdot, y) \|_{L^\infty}}{a_{\min}(y)^2} + \frac{1}{a_{\min}(y)}, \]  

while (3.9) is generalized to

\[ \| a^{1/2}(\cdot, y) \nabla (u - u_h)(\cdot, y) \|_{L^2} \lesssim h a_{\max}(y)^{1/2} \| \Delta u(\cdot, y) \|_{L^2} \text{ for all } y \in U_\beta; \]

see, e.g., [66]. Furthermore, if \( G \in L^2(D) \), then we also obtain an analogous result to (3.11) for the lognormal case; see, e.g., [46, Theorem 6].

3.3 Dimension Truncation

For both the uniform and lognormal cases, we observe that truncating the infinite sum in (2.2) and (2.6) to \( s \) terms is the same as setting \( y_j = 0 \) for \( j > s \). We denote the corresponding weak solution for the truncated case of (3.5) by \( u_h(x, y) := u(x, y_{\{1:s\}}) \). Throughout this article we refer to the value of \( s \) as the “truncation dimension.”

In the uniform case under Assumptions (U1) and (U2), for every \( f \in V^* \), every \( G \in V^* \), every \( y \in U \) and every \( s \in \mathbb{N} \), we have from [68, Theorem 5.1] that, with \( b_j \) defined in (2.3),

\[ \| u(\cdot, y) - u_h(\cdot, y) \|_{\mathcal{X}_0^*} \lesssim h^s \| f \|_{\mathcal{X}_s^*}, \]

\[ |G(u(\cdot, y)) - G(u_h(\cdot, y))| \lesssim h^{s+t'} \| f \|_{\mathcal{X}_{s+t'}^*} \| G \|_{\mathcal{X}_{s+t'}^*}. \]
\[ \|u(\cdot, y) - u^s(\cdot, y)\|_{V} \lesssim \|f\|_{V^*} \sum_{j \geq s+1} b_j, \]

\[ |I(G(u)) - I(G(u^s))| \lesssim \|f\|_{V^*} \|G\|_{V^*} \left( \sum_{j \geq s+1} b_j \right)^2. \quad (3.16) \]

In addition, if Assumptions (U4) and (U5) hold, then

\[ \sum_{j \geq s+1} b_j \leq \min \left( \frac{1}{1/p_0 - 1}, 1 \right) \left( \sum_{j \geq 1} b_j^{p_0} \right)^{1/p_0} s^{-(1/p_0 - 1)}. \quad (3.17) \]

Dimension truncation analysis in the lognormal case is more complicated. We summarize here the results from [46] which make use of [10]. Under Assumption (L4), we know from [46, Theorem 8] that for \( f \in V^* \), \( G \in V^* \), \( s \in \mathbb{N} \) and \( h > 0 \),

\[ |I(G(u_h)) - I(G(u^s_h))| \lesssim \|f\|_{V^*} \|G\|_{V^*} s^{-\chi}, \quad 0 < \chi < \left( \frac{1}{2} - \varepsilon \right) \Theta - \frac{1}{2}. \quad (3.18) \]

For the Matérn covariance (2.7) with \( \nu > d/2 \), we know from [46, Proposition 9] that Assumption (L4) holds with \( \Theta = 1 + 2\nu/d \) and \( \varepsilon \in \left( \frac{1}{2\Theta}, \frac{\Theta - 1}{2\Theta} \right) \). This implies that (3.18) holds for all \( 0 < \chi < \nu/d - 1/2 \).

4 Single-Level Versus Multi-level Algorithms

4.1 Single-Level Algorithms

We approximate the integral (2.5) or (2.10) in three steps:

1. Dimension truncation: the infinite sum in (2.2) or (2.6) is truncated to \( s \) terms.
2. FE discretization: the PDE in weak formulation (3.5) is solved using the piecewise linear FE method with meshwidth \( h \).
3. QMC quadrature: the integral of the FE solution for the truncated problem is estimated using a deterministic or randomized QMC method.

The deterministic version of this algorithm is therefore

\[ A_{\text{det}}^{\text{SL}}(G(u)) := Q \left( G \left( u_h^s \right) \right) = \frac{1}{n} \sum_{i=1}^{n} G(u_h^s(\cdot, y_i)), \quad y_i = \begin{cases} t_i - \frac{1}{2} & \text{for uniform,} \\ \Phi^{-1}(t_i) & \text{for lognormal,} \end{cases} \]

where \( t_1, \ldots, t_n \in [0, 1]^s \) are \( n \) QMC points from the \( s \)-dimensional standard unit cube. In the uniform case, these points are translated to the unit cube \( \left[ -\frac{1}{2}, \frac{1}{2} \right]^s \). In the lognormal case, these points are mapped to the Euclidean space \( \mathbb{R}^s \) by applying the inverse of the cumulative normal distribution function componentwise.
A randomized version of this algorithm with random shifting is then given by

$$A_{\text{ran}}(G(u)) := Q_{\text{ran}}(G(u_h^x)) = \frac{1}{r} \sum_{k=1}^{r} \frac{1}{n} \sum_{i=1}^{n} G(u_h^x(\cdot, y_{i,k})),$$

where $t_1, \ldots, t_n \in [0, 1]^s$ are $n$ QMC points as above, and $\Delta_1, \ldots, \Delta_r \in [0, 1]^s$ are $r$ independent random shifts generated from the uniform distribution on $[0, 1]^s$. Recall that the braces in $\{t_i + \Delta_k\}$ mean that we take the fractional part of each component in the vector $t_i + \Delta_k$. The total number of evaluations of the integrand is $rn$.

We assume that the cost for assembling the stiffness matrix is $O(s h^{-d})$ operations, and further assume that this is higher than the FE solve itself. Thus, the overall cost for the deterministic algorithm is $O(n s h^{-d})$ operations, while for the randomized algorithm it is $O(r n s h^{-d})$ operations. In practice we assume that $r$ is a fixed small constant, e.g., $r = 10$ or 20.

We sometimes refer to these algorithms as “single-level” algorithms, in contrast to “multi-level” algorithms to be discussed next.

### 4.2 Multi-Level Algorithms

The idea of multi-level algorithms in MC simulation originated from [54, 55] and was reinvented in [39, 40]; see also [41]. The general concept is quite easy to explain: If we denote the integral (2.5) or (2.10) by $I_\infty$ and define a sequence $I_0, I_1, \ldots$ of approximations converging to $I_\infty$, then we can write $I_\infty$ as a telescoping sum

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

and then apply different quadrature rules to the differences $I_{\ell} - I_{\ell-1}$, which we anticipate to get smaller as $\ell$ increases.

In our case, for each $\ell \geq 0$ we define $I_{\ell}$ to be the integral of $G(u_h^{x_\ell})$ corresponding to the FE solution with meshwidth $h_\ell$ of the truncation problem with $s_\ell$ terms. We assume that

$$1 \leq s_0 \leq s_1 \leq s_2 \leq \cdots \leq s_L \leq \cdots \quad \text{and} \quad h_0 \geq h_1 \geq h_2 \geq \cdots \geq h_L \geq \cdots > 0,$$

so that $I_\ell$ becomes a better approximation to $I_\infty$ as $\ell$ increases. For convenience we take $h_\ell \asymp 2^{-\ell}$. Throughout this article the notation $P \asymp Q$ means that we have $P \lesssim Q$ and $Q \lesssim P$.

The deterministic version of our multi-level algorithm takes the form (remembering the linearity of $G$)
A_{\text{ML}} (G(u)) := \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, y_i^{\ell})) \right),

\begin{align*}
y_i^{\ell} = \begin{cases}
t_i^{\ell} - \frac{1}{2} & \text{for uniform}, \\
\Phi^{-1}(t_i^{\ell}) & \text{for lognormal},
\end{cases}
\end{align*}

where we apply an $s_\ell$-dimensional QMC rule with $n_\ell$ points $t_1^{\ell}, \ldots, t_{n_\ell}^{\ell} \in [0, 1]^{s_\ell}$ to the integrand $G(u_{h_\ell}^{s_\ell} - u_{h_{\ell-1}}^{s_{\ell-1}})$, and we define $u_{h_{-1}}^{s_{-1}} := 0$. The total number of evaluations of the integrand is $O(\sum_{\ell=0}^{L} n_\ell)$.

We can also use $r_\ell$ random shifts at each level, noting that the shifts should all be independent. Then a randomized version of our multi-level algorithm with random shifting takes the form

$$A_{\text{ML ran}} (G(u)) := \sum_{\ell=0}^{L} \left( \frac{1}{r_\ell} \sum_{k=1}^{r_\ell} \frac{1}{n_\ell} \sum_{i=1}^{n_\ell} G((u_{h_\ell}^{s_\ell} - u_{h_{\ell-1}}^{s_{\ell-1}})(\cdot, y_{i,k}^{\ell})) \right),$$

$$y_{i,k}^{\ell} = \begin{cases}
(t_i^{\ell} + \Delta_i^{\ell,k}) - \frac{1}{2} & \text{for uniform}, \\
\Phi^{-1}((t_i^{\ell} + \Delta_i^{\ell,k})) & \text{for lognormal}.
\end{cases}$$

In this case the total number of evaluations of the integrand is $O(\sum_{\ell=0}^{L} n_\ell r_\ell)$.

We assume that the overall cost for the multi-level algorithm is $O(\sum_{\ell=0}^{L} n_\ell s_\ell h_{\ell}^{-d})$ operations for the deterministic version and $O(\sum_{\ell=0}^{L} n_\ell r_\ell s_\ell h_{\ell}^{-d})$ operations for the randomized version.

5 First-Order Versus Higher-Order Methods

Contemporary analysis of QMC methods is often carried out in the setting of weighted spaces following [33, 101, 102]. The general concept is the observation that in many practical examples, not all the integration variables are of equal importance, and furthermore, there could be a difference in importance associated with each different subset of variables. Under appropriate conditions, it is known that QMC methods can be constructed with error bounds that are independent of the dimension. In this section we briefly summarize known results for randomly shifted lattice rules (first order) and interlaced polynomial lattice rules (higher order) in suitably weighted spaces. Construction of these point sets are surveyed in [84], and we provide practical pointers to a software implementation in Sect. 7. We note that there usually is a tight bond between the QMC method and the chosen function space setting.

5.1 Weighted Sobolev Spaces over $[0, 1]^s$ and Randomly Shifted Lattice Rules

An $n$-point lattice rule in the unit cube $[0, 1]^s$ is a QMC method (3.1) with points
where $\mathbf{z} \in \mathbb{Z}^s$ is known as the generating vector and the braces indicate that we take the fractional parts of a vector, as in (3.2). The quality of a lattice rule is determined by the choice of the generating vector.

We analyze randomly shifted lattice rules in a weighted and unanchored Sobolev space which is a Hilbert space containing functions defined over the unit cube $[0, 1]^s$, with square integrable mixed first (weak) derivatives. The norm is given by

$$
\| F \|_{s, \gamma} = \left[ \sum_{u \subseteq \{1:s\}} \frac{1}{\gamma_u} \int_{[0,1]^{|u|}} \left( \int_{[0,1]^{s-|u|}} \frac{\partial^{|u|} F}{\partial \mathbf{y}_u} (\mathbf{y}_u; \mathbf{y}_{\{1:s\}\backslash u}) \, d\mathbf{y}_u \right)^2 \, d\mathbf{y}_u \right]^{1/2},
$$

(5.2)

where $\{1: s\}$ is a shorthand notation for the set of indices $\{1, 2, \ldots, s\}$, $(\partial^{|u|} F)/(\partial \mathbf{y}_u)$ denotes the mixed first derivative of $F$ with respect to the “active” variables $\gamma_u = (y_j)_{j \in u}$, while $\mathbf{y}_{\{1:s\}\backslash u} = (y_j)_{j \in \{1:s\}\backslash u}$ denotes the “inactive” variables. This norm is said to be “unanchored” because the inactive variables are integrated out, as opposed to being “anchored” at some fixed value, say, 0.

There is a weight parameter $\gamma_u \geq 0$ associated with each subset of variables $\mathbf{y}_u$. A small $\gamma_u$ means that $F$ depends weakly on the set of variables $\mathbf{y}_u$. If $\gamma_u = 0$, then it is understood that the corresponding mixed first derivative is also zero, and then, the convention $0/0 = 0$ is used. We denote by $\mathbf{y}$ the set of all weights $\gamma_u$, and we take $\gamma_{\emptyset} = 1$. There are in general $2^s$ weights in $s$ dimensions, far too many for practical purposes. Special forms of weights have been considered in the literature, including the so-called “product weights” and “order-dependent weights”; see, e.g., [33,101,102]. Later we will show that a hybrid of these two forms of weights, called “product and order-dependent weights” or “POD weights” for short, naturally arise in the context of PDE applications. They take the form

$$
\gamma_u = \Gamma_{|u|} \prod_{j \in u} \Upsilon_j,
$$

(5.3)

which is specified by two sequences $\Upsilon_1 \geq \Upsilon_2 \geq \cdots > 0$ and $\Gamma_0 = \Gamma_1 = 1$, $\Gamma_2$, $\Gamma_3$, $\ldots \geq 0$. In this context the cardinality $|u|$ of the set $u$ is commonly referred to as the “order.” Here the factor $\Gamma_{|u|}$ is said to be order-dependent because it is determined solely by the cardinality of $u$ and not the precise indices in $u$. The dependence of the weight $\gamma_u$ on the indices $j \in u$ is controlled by the product of terms $\Upsilon_j$. Each term $\Upsilon_j$ in the sequence corresponds to one coordinate direction; the sequence being non-increasing indicates that successive coordinate directions become less important.

For randomly shifted lattice rules in the unanchored Sobolev space, we have the root-mean-square error bound (3.3) where an explicit expression for the shift-averaged worst-case error is known, allowing it to be analyzed in theory and computed in practice. It has been proved that a good generating vector $\mathbf{z}$ for an $n$-point rule can be constructed to achieve the optimal convergence rate of $O(n^{-1+\delta})$, $\delta > 0$, and the
implied constant can be independent of the dimension $s$ under appropriate conditions on the weights $\gamma$. The construction is by a component-by-component (CBC) algorithm: the components of the generating vector $z$ are obtained one at a time while keeping previously chosen components fixed. Fast CBC algorithms (using FFT) can construct an $n$-point rule in $s$ dimensions in $\Theta(sn \log n)$ operations in the case of product weights, and in $\Theta(sn \log n + s^2 n)$ operations in the case of POD weights.

We summarize the error bound in the theorem below. In the following, $\zeta(x) := \sum_{k=1}^{\infty} k^{-x}$ denotes the Riemann zeta function.

**Theorem 5.1** Let $F$ belong to the unanchored Sobolev space defined over $[0, 1]^s$ with weights $\gamma$. A randomly shifted lattice rule with $n = 2^m$ points in $s$ dimensions can be constructed by a CBC algorithm such that for $r$ independent shifts and for all $\lambda \in (1/2, 1]$, 

$$\sqrt{\mathbb{E}[|I(F) - Q_{\text{ran}}(F)|^2]} \leq \frac{1}{\sqrt{r}} \left( \frac{2}{n} \sum_{\emptyset \neq u \subseteq \{1:s\}} \gamma_u^\lambda |\varrho(\lambda)|^{|u|} \right)^{1/(2\lambda)} \|F\|_{s, \gamma},$$

where

$$\varrho(\lambda) = \frac{2\zeta(2\lambda)}{(2\pi^2)^\lambda}.$$ 

In the theorem above we have restricted ourselves to the case where $n$ is a power of 2, as this is the most convenient setting for generating the points on a computer. For general $n$, the factor $2/n$ should be replaced by $1/\varphi_{\text{tot}}(n)$, where $\varphi_{\text{tot}}(n)$ is the Euler totient function, i.e., the number of integers between 1 and $n - 1$ that are relatively prime to $n$. When $n$ is a power of a prime, we have $1/\varphi_{\text{tot}}(n) \leq 2/n$, and hence, the theorem as stated also holds in this case.

The best rate of convergence clearly comes from choosing $\lambda$ close to $1/2$, but the advantage is offset by the fact that $\zeta(2\lambda) \to \infty$ as $\lambda \to (1/2)_+$. 

The CBC construction yields a lattice rule which is “extensible” in dimension $s$, meaning that a generating vector constructed for dimension $s$ can be used in lower dimensions by taking only the initial components and that components for higher dimensions can be appended to existing components by continuing with the construction.

It is also possible to construct “lattice sequences” which are extensible or embedded in the number of points $n$, meaning that the same generating vector can be used to generate more points or less points without having to construct the existing points anew. This extensibility in $n$ can be achieved at the expense of increasing the implied constant in the error bound; see, e.g., [17,32].

### 5.2 Weighted Space Setting in $\mathbb{R}^s$ and Randomly Shifted Lattice Rules

For an integral formulated over the Euclidean space $\mathbb{R}^s$ as in (3.4), the transformed integrand $F \circ \Phi^{-1}$ arising from practical applications typically does not belong to the Sobolev space defined over the unit cube due to the integrand being unbounded near the
boundary of the cube, or because the mixed derivatives of the transformed integrand do not exist or are unbounded. Thus, most QMC theories, including Theorem 5.1, generally do not apply in practice. Here we summarize a special weighted space setting in \( \mathbb{R}^s \) for which randomly shifted lattice rules have been shown to achieve nearly the optimal convergence rate of order one; see [70,76]. The norm in this setting is given by

\[
\|F\|_{s,\gamma} = \left[ \sum_{u \subseteq \{1:s\}} \frac{1}{\gamma_u} \int_{\mathbb{R}^{|u|}} \left( \int_{\mathbb{R}^{s-|u|}} \frac{\partial |u|}{\partial y_u} F(y_u; y_{\{1:s\}\setminus u}) \phi(y_j) dy_{\{1:s\}\setminus u} \prod_{j \in \{1:s\}\setminus u} \phi(y_j) \right)^2 \right]^{1/2},
\]

where

\[
\varphi_j(\lambda) = \exp(-2 \alpha_j |y_j|), \quad \alpha_j > 0.
\]

We have the following result from [46, Theorem 15] for randomly shifted lattice rules in this setting; see [76, Theorem 8] for results on general functions \( \varphi \) and \( \varphi_j \). We state the result again for \( n \) a power of 2, but it holds when \( n \) is a power of any prime.

**Theorem 5.2** Let \( F \) belong to the weighted function space over \( \mathbb{R}^s \) with weights \( \gamma \), with \( \phi \) being the standard normal density, and with weight functions \( \varphi_j \) given by (5.5). A randomly shifted lattice rule with \( n = 2^m \) points in \( s \) dimensions can be constructed by a CBC algorithm such that for \( r \) independent shifts and for all \( \lambda \in (1/2, 1] \),

\[
\sqrt{\mathbb{E}[|I(F) - Q_{\text{ran}}(F)|^2]} \leq \frac{1}{\sqrt{r}} \left( \frac{2}{n} \sum_{\emptyset \neq u \subseteq \{1:s\}} \gamma_u \prod_{j \in u} \varphi_j(\lambda) \right)^{1/(2\lambda)} \|F\|_{s,\gamma},
\]

where

\[
\varphi_j(\lambda) = 2 \left( \frac{\sqrt{2\pi} \exp(\alpha_j^2/\eta_*)}{\pi^{2-2\eta_*}(1-\eta_*)\eta_*} \right)^{\lambda} \zeta(\lambda + 1/2), \quad \text{and} \quad \eta_* = \frac{2\lambda - 1}{4\lambda}.
\]

Alternatively, for the lognormal case it is also possible to choose the weight functions \( \varphi_j \) to take the form of a normal density that decays much slower than \( \phi \). The corresponding result is given below, which can be obtained from [76, Theorem 8] together with [70, Example 4].
Theorem 5.3 If we replace the weight functions \( \varpi_j \) in Theorem 5.2 by \( \varpi_{\text{alt}} \), given by
\[
\varpi_{\text{alt}}(y_j) = \exp(-\alpha y_j^2)
\]
with \( \alpha < \frac{1}{2} \), then the root-mean-square error bound in
Theorem 5.2 holds for all \( \lambda \in (1/(2 - 2\alpha), 1] \), but with \( \rho_j(\lambda) \) replaced by
\[
\rho_{\text{alt}}(\lambda) = \frac{\sqrt{2\pi}}{\pi^{2 - 2\alpha}(1 - \alpha)\alpha} \lambda (2(1 - \alpha)\lambda).
\]

5.3 Weighted Space of Smooth Functions over \([0, 1]^s\) and Interlaced Polynomial Lattice Rules

We now introduce interlaced polynomial lattice rules which are a special family of higher-order QMC rules; see, e.g., [20, 21, 23, 43, 44]. We limit ourselves to the case where the polynomials are over the finite field \( \mathbb{Z}_2 \) with two elements. This has major advantages in the computer implementations for constructing these rules and generating the points, and simplifies the presentation.

Let \( \mathbb{N}_0 = \{0, 1, 2, \ldots\} \). We will identify numbers \( x \in [0, 1) \) and \( i \in \mathbb{N}_0 \) that have finite base 2 representations (i.e., having only a finite number of bits being 1) by a vector enumerating the bits, denoted by \( \bar{x} \) and \( \bar{i} \), or by a polynomial in the formal variable \( \varsigma \), denoted by \( x(\varsigma) \) and \( i(\varsigma) \). We will use the common notation \( \mathbb{Z}_2[\varsigma] \) for formal polynomials in \( \varsigma \) and \( \mathbb{Z}_2((\varsigma^{-1})) \) for formal polynomials in \( \varsigma^{-1} \) as well as \( \mathbb{Z}_2(\varsigma) \) for formal Laurent series (with powers going in both directions). Then, for \( x \in [0, 1) \), with \( x_\ell \in \mathbb{Z}_2 \) for \( \ell \geq 1 \) denoting the base 2 digits of \( x \), we have in this multitude of notations

\[
x = \sum_{\ell \geq 1} x_\ell 2^{-\ell} = (0.x_1x_2\ldots)_2 \in \mathbb{R} \simeq (x_1, x_2, \ldots)^\top =: \bar{x} \in \mathbb{Z}_2^\infty \]
\[
\simeq x_1 \varsigma^{-1} + x_2 \varsigma^{-2} + \cdots =: x(\varsigma) \in \mathbb{Z}_2((\varsigma^{-1})).
\]

Similarly for integers \( i \in \mathbb{N}_0 \) where the bit expansion goes in the other direction, with \( i_\ell \in \mathbb{Z}_2 \) for \( \ell \geq 0 \) denoting the base 2 digits of \( i \), we have

\[
i = \sum_{\ell \geq 0} i_\ell 2^\ell = (\ldots i_2i_1i_0)_2 \in \mathbb{N}_0 \simeq (i_0, i_1, i_2, \ldots)^\top =: \bar{i} \in \mathbb{Z}_2^\infty \]
\[
\simeq i_0 \varsigma^0 + i_1 \varsigma^1 + i_2 \varsigma^2 + \cdots =: i(\varsigma) \in \mathbb{Z}_2[\varsigma].
\]

We need one further notation to limit the number of bits to \( m \); this will be denoted by \( [x(\varsigma)]_m \) and \( [\bar{x}]_m \), or \( [i(\varsigma)]_m \) and \( [\bar{i}]_m \).

With this notation in place, the points of a polynomial lattice rule are given by

\[
t_i \simeq [\mathbf{r}_i(\varsigma)]_m, \quad \mathbf{r}_i(\varsigma) = \frac{i(\varsigma) \mathbf{z}(\varsigma) \mod P(\varsigma)}{P(\varsigma)} \in (\mathbb{Z}_2((\varsigma^{-1})))^s, \quad i = 0, \ldots, 2^m - 1,
\]

\[\text{ Springer}\]
where $P(\varsigma)$ is an irreducible polynomial of degree $m$ over $\mathbb{Z}_2$, known as the modulus, and the vector of polynomials $\mathbf{z}(\varsigma) = (z_1(\varsigma), \ldots, z_s(\varsigma)) \in \mathbb{Z}[\varsigma]^s$ is known as the generating vector. We see that polynomial lattice rules take the same form as lattice rules in (5.1), but the integers are replaced by polynomials, and thus, the multiplication and division are replaced by their polynomial equivalents over $\mathbb{Z}_2(\varsigma)$. We remark that the points are here indexed from 0 to $2^m - 1$ which is different from the convention we used in the rest of this article.

Alternatively, the polynomial multiplication and division can be written in matrix–vector notation over $\mathbb{Z}_2$ by identifying a Hankel matrix $C_j = (a_j, r + t - 1)^r, t \geq 1$ with the division $z_j(\varsigma)/P(\varsigma) = \sum_\ell a_j, \ell \varsigma^{-\ell}$ for $j = 1, \ldots, s$. In principle these matrices are in $\mathbb{Z}_2^{\infty \times \infty}$ but we restrict them to be of size $m \times m$. In terms of these “generating matrices” $C_1, \ldots, C_s \in \mathbb{Z}_2^{m \times m}$ we can then write the points of the polynomial lattice rule as

$$t_i \simeq (C_1 [\vec{i}]_m, \ldots, C_s [\vec{i}]_m)^\top, \quad i = 0, \ldots, 2^m - 1. \quad (5.7)$$

The matrix–vector form is to be preferred when generating the actual points as point generation then boils down to simple bit operations. In fact, if the order in which the points are iterated is changed into Gray code ordering so that only one bit changes in the index of the point at a time, then the coordinate of the next point can be obtained by simply applying the XOR-operation between the previous value of that coordinate and the particular column from the generator matrix corresponding to the single bit change in the Gray code.

An interlaced polynomial lattice rule in $s$ dimensions with interlacing factor $\alpha$ is now obtained by taking a polynomial lattice rule in $\alpha s$ dimensions and then interlacing the bits from every successive $\alpha$ dimensions to yield one dimension. More explicitly, for $\alpha = 3$, given three coordinates $x = (0, x_1 x_2 \ldots x_m)_2, y = (0, y_1 y_2 \ldots y_m)_2$ and $z = (0, z_1 z_2 \ldots z_m)_2$ we interlace their bits to obtain $w = (0, x_1 y_1 z_1 x_2 y_2 z_2 \ldots x_m y_m z_m)_2$. Note that interlacing can be applied to any existing point set, but the construction of the polynomial lattice rules in this section actually takes the interlacing into account.

The interlacing operation can also be done directly by constructing new generating matrices which are obtained by interlacing the rows of $\alpha$ successive generating matrices. That is, if we have generating matrices $C_1, \ldots, C_s \in \mathbb{Z}_2^{m \times m}$, then we obtain new generating matrices $B_1, \ldots, B_s \in \mathbb{Z}_2^{\alpha m \times m}$ and the points of the interlaced polynomial lattice rule are given by

$$t_i \simeq (B_1 [\vec{i}]_m, \ldots, B_s [\vec{i}]_m)^\top, \quad i = 0, \ldots, 2^m - 1. \quad (5.8)$$

We remark that for efficiency reasons the points are normally iterated in Gray code ordering such that each coordinate can be obtained by a single XOR operation.

A function space setting for smooth integrands defined over the unit cube was introduced in [23]. Let $\alpha, s \in \mathbb{N}$ and $1 \leq q, r \leq \infty$, and let $\mathbf{\gamma} = (\gamma_u)_{u \subset \mathbb{N}}$ be a collection of weights as in the previous subsections. The norm in this setting for $1 \leq q, r < \infty$ is given by

$$\|f\|_{L^q L^r} = \left( \int_{\mathbb{R}^s} \left( \int_{\mathbb{R}^s} |f(x)|^q \, dx \right)^{r/q} \, dx \right)^{1/r}.$$
\[ \| F \|_{s, \alpha, \gamma, q, r} := \left[ \sum_{u \subseteq [1:s]} \left( \frac{1}{\gamma_u^q} \sum_{v \subseteq u} \sum_{\tau_{u \setminus v} \in [1:|u|]} \left( \int_{[0,1]^{|u|}} \int_{[0,1]^{s-|v|}} \left( \partial^{(\alpha, \tau_{u \setminus v}, 0)} F \right)(y) \, dy \right) \right) \right]^{r/q} \]

with the obvious modifications if \( q \) or \( r \) is infinite; see (5.10). Here \((\alpha, \tau_{u \setminus v}, 0)\) denotes a vector \( \nu \) with \( \nu_j = \alpha \) for \( j \in v \), \( \nu_j = \tau \) for \( j \in u \setminus v \), and \( \nu_j = 0 \) for \( j \notin u \). We denote the \( \nu \)th partial derivative of \( F \) by \( \partial^{\nu} F = (\partial^{\nu_1} F / \partial y_1^{\nu_1} \partial y_2^{\nu_2} \cdots \partial y_s^{\nu_s}) \). We remark that the norm is stated incorrectly in [23, Equation (3.7)].

A new form of weights arose from the analysis of interlaced polynomial lattice rules in this setting in the context of PDE applications; see [23]. They are called “smoothness-driven product and order-dependent weights” or “SPOD weights” for short, and take the form

\[ \gamma_u = \sum_{\nu_u \in [1:|u|]} \Gamma_{|u|} \prod_{j \in u} \gamma_j(v_j). \] (5.9)

For PDE applications, it was shown in [23] that good theoretical results can be obtained by taking \( r = \infty \) while \( q \) can be arbitrary. Therefore, for simplicity, here we take \( q = r = \infty \) and denote the corresponding norm by

\[ \| F \|_{s, \alpha, \gamma, \infty, \infty} := \sup_{u \subseteq [1:s]} \sup_{y \in [0,1]^{|u|}} \left( \int_{[0,1]^{s-|u|}} \left( \partial^{(\alpha, \tau_{u \setminus v}, 0)} F \right)(y) \, dy \right) \]

(5.10)

The following theorem is adjusted from [23, Theorem 3.10] for \( b = 2 \). We made use of a new result in [107] that a constant \( C_{\alpha, b} \) which usually appears is exactly 1 when \( b = 2 \). Furthermore, we followed the proof of [84, Theorem 5.1] to obtain the factor \( 2/n \) instead of \( 2/(n - 1) \). We remark that the interlacing factor \( \alpha \) needs to be at least 2 for the theorem to hold.

**Theorem 5.4** Let \( F \) belong to the weighted space of smooth functions over \([0,1]^s\) with \( \alpha \geq 2 \) and weights \( \gamma \). An interlaced polynomial lattice rule with interlacing factor \( \alpha \), with irreducible modulus polynomial of degree \( m \), and with \( n = 2^m \) points in \( s \) dimensions, can be constructed by a CBC algorithm such that, for all \( \lambda \in (1/\alpha, 1) \),

\[ |I(F) - Q(F)| \leq \left( \frac{2}{n} \sum_{\emptyset \neq u \subseteq [1:s]} \gamma_u^{\lambda} [\rho_{\alpha}(\lambda)]^{|u|} \right)^{1/\lambda} \| F \|_{s, \alpha, \gamma}, \]
where

$$\rho_\alpha(\lambda) = 2^{\alpha\lambda}(\alpha-1)/2 \left( \left( 1 + \frac{1}{2^{\alpha\lambda}} - 2 \right)^\alpha - 1 \right).$$

If the weights \( \gamma \) are SPOD weights, then the CBC algorithm has cost \( O(\alpha s n \log n + \alpha^2 s^2 n) \) operations. If the weights \( \gamma \) are product weights, then the CBC algorithm has cost \( O(\alpha s n \log n) \) operations.

6 Error Analysis

In this section we summarize the error analysis for various algorithms based on QMC methods in the uniform and lognormal cases. Not surprisingly, the error is a combination of dimension truncation error, FE discretization error and QMC quadrature error. The errors are additive in the case of single-level algorithms, while in the case of multi-level algorithms the overall error includes some multiplicative effects between FE and QMC errors. Recall that bounds on truncation and FE errors are summarized in Sect. 3, while bounds on QMC errors for different spaces are summarized in Sect. 5.

The convergence of the QMC method (deterministic or randomized) can be independent of the truncation dimension both in the rate (i.e., the exponent of \( 1/n \) in the error estimate) and in the asymptotic constant. This is achieved by working in weighted spaces with strategically chosen weights \( \gamma_u \). To bound the QMC error we will use Theorems 5.1, 5.2 or 5.4 depending on the setting. From these theorems we see that the key step of our analysis is to obtain bounds on the particular weighted norm of the integrand, which depends on the PDE solution \( u_h^s(x, y) \). Specifically, this means that we need to obtain bounds on the mixed partial derivatives of \( u_h^s(x, y) \) with respect to \( y \). Once we obtain estimates on the norm of the integrand, we then choose suitable weights \( \gamma_u \) to reduce the error bound, optimizing on the theoretical QMC convergence rate while ensuring (under some conditions) that the implied constant is independent of the truncation dimension. The chosen weights \( \gamma_u \) then enter the fast CBC construction to produce tailored QMC methods for the given setting.

In the next subsections we outline the error analysis for different algorithms under different settings. First-order results are based on randomly shifted lattice rules and we obtain probabilistic error bounds. Higher-order results are based on interlaced polynomial lattice rules and we obtain deterministic error bounds. In the latter case we also replace the \( \ell^2 \) norm in the typical definition of the function space norm by the \( \ell^\infty \) norm, and this enables us to gain an extra factor of \( n^{-1/2} \) in the QMC convergence rate in the context of PDE applications. However, this analysis only applies in the uniform case.

To obtain multiplicative error bounds in the case of multi-level algorithms, we need to assume a stronger regularity of \( u_h^s(x, y) \) in \( x \), and we need to establish regularity results simultaneously in \( x \) and \( y \). This makes the analysis more challenging. The resulting weights \( \gamma_u \) are also different from those in the single-level algorithms.

The error versus cost analysis depends crucially on the cost model assumption. In the single-level algorithms we choose \( n, s, h \) to balance the three sources of errors.
In the multi-level algorithms we choose \( n_\ell, s_\ell, h_\ell \), for each level, to minimize the error for a fixed cost using Lagrange multiplier arguments, and we choose \( L \) such that the combined error meets the required threshold. We assume that \( r \) and \( r_\ell \) are fixed constants.

Before we proceed we introduce some notation. For a multi-index \( \nu = (\nu_j)_{j \geq 1} \) with \( \nu_j \in \{0, 1, 2, \ldots\} \), we write its “order” as \( |\nu| := \sum_{j \geq 1} \nu_j \) and its “support” as \( \text{supp}(\nu) := \{j \geq 1 : \nu_j \geq 1\} \). Furthermore, we write \( \nu! := \prod_{j \geq 1} \nu_j! \), which is different from \( |\nu|! := (\sum_{j \geq 1} \nu_j)! \). We denote by \( F \) the (countable) set of all “finitely supported” multi-indices:

\[
F := \{\nu \in \mathbb{N}_0^\infty : \text{supp}(\nu) < \infty\}.
\]

For \( \nu \in F \), we denote the \( \nu \)th partial derivative with respect to the parametric variables \( y \) by

\[
\partial^\nu = \frac{\partial^{|\nu|}}{\partial y_1^{\nu_1} \partial y_2^{\nu_2} \ldots}.
\]

For any sequence of real numbers \( b = (b_j)_{j \geq 1} \), we write \( b^\nu := \prod_{j \geq 1} b_j^{\nu_j} \). By \( m \leq \nu \) we mean that the multi-index \( m \) satisfies \( m_j \leq \nu_j \) for all \( j \). Moreover, \( \nu - m \) denotes a multi-index with the elements \( \nu_j - m_j \), and \( \binom{\nu}{m} := \prod_{j \geq 1} \binom{\nu_j}{m_j} \). We denote by \( e_j \) the multi-index whose \( j \)th component is 1 and all other components are 0.

We remind the reader that throughout this article, the gradient \( \nabla \) and the Laplacian \( \Delta \) are to be taken with respect to the spatial variables \( x \), while the partial derivatives \( \partial^\nu \) are to be taken with respect to the parametric variables \( y \).

Note that in Sect. 5 we have used two different notations for the mixed derivatives with respect to \( y \). In the norms \((5.2)\) and \((5.4)\) we restrict only to mixed first derivatives, and the subsets \( u \subseteq \{1 : s\} \) are used to identify the indices of the variables with respect to which we differentiate. For example, if \( u = \{1, 2, 5\} \) then

\[
\frac{\partial^{|u|} F}{\partial y_u} = \frac{\partial^3 F}{\partial y_1 \partial y_2 \partial y_5} = \partial^\nu F,
\]

with \( \nu = (1, 1, 0, 0, 1, 0, 0, \ldots) \) in the multi-index notation.

### 6.1 First-Order, Single-Level, Uniform

The mean-square error for our single-level algorithm with randomly shifted lattice rules can be expressed as

\[
\mathbb{E} \left[ |(I - A_{\text{ran}}^{\text{SL}}(G(u)))|^2 \right] = |I(G(u - u_h^s))|^2 + \mathbb{E} \left[ |(I - Q_{\text{ran}})(G(u_h^s))|^2 \right], \quad (6.1)
\]

where we used the linearity of \( G \) and the unbiased property of randomly shifted QMC rules, i.e., \( \mathbb{E} \left[ Q_{\text{ran}}(F) \right] = I(F) \). The first term on the right-hand side of \((6.1)\) can be split trivially using linearity of \( I \) and \( G \) into
\[ I (G (u - u_h^s)) = I(G(u - u^s)) + I (G (u^s - u_h^s)) . \] (6.2)

In the uniform case, we estimate the truncation error using (3.16)–(3.17), and estimate the FE error using (3.11) but adapt it to the truncated solutions (this is valid since (3.10) holds for all \( y \in U \), including those with \( y_j = 0 \) for \( j > s \)). For the QMC error we use Theorem 5.1 which requires a bound on the norm \( \|G(u_h^s)\|_{s,y} \), and we see from the definition (5.2) that we need to bound the mixed first partial derivatives of \( G(u_h^s(\cdot, y)) \). Due to linearity and boundedness of \( G \), we have

\[ \left| \frac{\partial^{|u|}}{\partial y_u} G(u_h^s(\cdot, y)) \right| = \left| G\left( \frac{\partial^{|u|}}{\partial y_u} u_h^s(\cdot, y) \right) \right| \leq \|G\|_{V^*} \left\| \frac{\partial^{|u|}}{\partial y_u} u_h^s(\cdot, y) \right\|_{V} . \] (6.3)

Hence, we need a regularity result on the PDE solution \( u_h^s(x, y) \) with respect to the parameters \( y \). In Lemma 6.1 we state such a result for general mixed derivatives using the multi-index notation.

**Lemma 6.1** In the uniform case under Assumptions (U1) and (U2), for every \( f \in V^* \), every \( y \in U \), and every \( \nu \in F \), we have

\[ \left\| \partial^\nu u(\cdot, y) \right\|_V = \left\| \nabla (\partial^\nu u(\cdot, y)) \right\|_{L^2} \leq \left| \nu \right|! \frac{b^\nu \|f\|_{V^*}}{a_{\min}} , \] where the sequence \( b \) is defined in (2.3). The same estimate holds when the exact solution \( u \) is replaced by \( u_h^s \).

This result was proved in [15]. The proof is by induction on \( |\nu| \). We take the mixed partial derivative \( \partial^\nu \) on both sides of the weak formulation (3.5) and then substitute the test function \( v = \partial^\nu u(\cdot, y) \). Rearranging and estimating the terms then yield the required bound. Since the same proof strategy is used repeatedly in subsequent more complicated proofs, we include this relatively simple proof in Sect. 9 as a first illustration.

For bounds on mixed first derivatives in the norm (5.2), we restrict Lemma 6.1 to multi-indices \( \nu \) with \( \nu_j \leq 1 \) for all \( j \). Using also (6.3), we obtain the estimate

\[ \|G(u_h^s)\|_{s,y} \leq \frac{\|f\|_{V^*} \|G\|_{V^*}}{a_{\min}} \left( \sum_{\nu \subseteq \{1:s\}} (|\nu|!)^2 \prod_{j \in \nu} b_j \right)^{1/2} . \] (6.4)

Combining this with Theorem 5.1 gives

\[ \mathbb{E}\left[ \left| (I - Q_{\text{ran}})(G(u_h^s)) \right|^2 \right] \leq \frac{1}{r} \frac{C^2_f(\lambda)}{n^{1/\kappa}} \frac{\|f\|^2_{V^*} \|G\|^2_{V^*}}{a^2_{\min}} , \]
where
\[
C_2^\gamma (\lambda) = \left(2 \sum_{u \subseteq \{1 \ldots s\}} \gamma_u^\lambda [\rho(\lambda)]^{|u|} \right)^{1/\lambda} \left( \sum_{u \subseteq \{1 \ldots s\}} \frac{(|u|!)^2 \prod_{j \in u} b_j^2}{\gamma_u} \right). 
\]

Elementary calculus (see [68, Lemma 6.2]) yields the result that for any given \(\lambda\), \(C_2^\gamma (\lambda)\) is minimized as a function of \(\gamma_u\) by taking
\[
\gamma_u = \left(\frac{|u|! \prod_{j \in u} b_j}{\sqrt[2/(1+\lambda)]{\rho(\lambda)}}\right), 
\]
which are of the POD form (5.3). Under Assumption (U5), it is proved in [68, Theorem 6.4] that if we take
\[
\lambda = \begin{cases} 
\frac{1}{2 - 2\delta} & \text{for some } \delta \in (0, 1/2) \text{ when } p_0 \in (0, 2/3), \\
\frac{p_0}{2 - p_0} & \text{when } p_0 \in (2/3, 1),
\end{cases} 
\]
then the constant \(C_2^\gamma (\lambda)\) can be bounded independently of \(s\). The case \(p_0 = 1\) can also be analyzed but it requires an additional assumption which can be quite restrictive; we omit this case here.

We summarize the final result in the theorem below.

**Theorem 6.1** In the uniform case under Assumptions (U1)--(U5), for every \(f \in L^2(D)\) and every \(G \in L^2(D)\), the single-level algorithm \(A_{SL}^G(u)\) using a randomly shifted lattice rule with \(n = 2^m\) points constructed from a CBC algorithm with POD weights (6.5)--(6.6), at the pre-computation cost of \(O(s n \log n + s^2 n)\) operations, achieves the mean-square error bound
\[
\mathbb{E} \left[ \left| (I - A_{SL}^G(u)) G(u) \right|^2 \right] \leq s^{-4(1/p_0-1)} + h^4 + r^{-1} n^{-2 \min(1/p_0-1/2, 1-\delta)}, \quad \delta \in (0, \frac{1}{2}),
\]
where the implied constant is independent of \(s, h, r\) and \(n\).

If we treat \(r\) as a fixed constant and choose \(s, h, n\) to balance the terms so that the mean-square error is \(O(\varepsilon^2)\), then the cost of the algorithm \(A_{SL}^G(u)\) is \(O(s n h^{-d}) = O(\varepsilon^{-\tau})\) with
\[
\tau = \frac{p_0}{2 - 2p_0} + \frac{d}{2} + \max \left( \frac{2p_0}{2 - p_0}, \frac{1}{1 - \delta} \right), \quad \delta \in (0, \frac{1}{2}).
\]
6.2 First-Order, Multi-Level, Uniform

For our multi-level algorithm with randomly shifted lattice rules, we can write the mean-square error as

\[
\mathbb{E} \left[ |(I - A_{\text{ran}}^{\text{ML}})(G(u))|^2 \right]
\]

\[= |I(G(u - u_{h_L}^{S_L}))|^2 + \sum_{\ell=0}^{L} \mathbb{E} \left[ |(I - Q_{\text{ran}}^\ell)(G(u_{h_L}^{S_{\ell-1}} - u_{h_{\ell-1}}^{S_{\ell-1}}))|^2 \right], \quad (6.7)
\]

where \( Q_{\text{ran}}^\ell \) denotes a randomly shifted lattice rule in \( s_\ell \) dimensions with \( n_\ell \) points and \( r_\ell \) independent shifts. The first term on the right-hand side of (6.7) can be estimated in exactly the same way as for the single-level algorithm in the previous subsection. Each term in the sum over \( \ell \) in (6.7) can be estimated using Theorem 5.1. For simplicity we restrict our discussion here to the case where \( s_\ell = s \) for all \( \ell \). The case \( s_\ell \neq s_{\ell-1} \) was analyzed in [69, Theorem 8]. Thus, we need to estimate the norm

\[
\| G(u_{h_\ell}^s - u_{h_{\ell-1}}^s) \|_{s,\gamma} \leq \| G(u^s - u_{h_\ell}^s) \|_{s,\gamma} + \| G(u^s - u_{h_{\ell-1}}^s) \|_{s,\gamma}.
\]

For this we need to estimate the mixed first derivatives of \( G(u - u_h) \) with respect to \( y \). We state a result for general mixed derivatives in Lemma 6.4.

To establish Lemma 6.4 we need some intermediate results in Lemmas 6.2 and 6.3. These three lemmas together correspond to [69, Theorems 6 and 7], but in addition to a generalization of [69, Theorem 7] from first derivatives to general derivatives, the results are different because the sequence \( \mathfrak{B} \) defined here is simpler compared to [69], and we take a different (arguably more direct) route with the proofs. However, we get bigger factorials here. For simplicity we restrict our discussion to the case \( f, G \in L^2(D) \). The results can be generalized to cover \( f \in H^{-1+t}(D) \) and \( G \in H^{-1+t'}(D) \) for \( t, t' \in [0, 1] \) as in [69].

**Lemma 6.2** In the uniform case under Assumptions (U1)–(U3), for every \( f \in L^2(D) \), every \( y \in U \) and every \( v \in \mathfrak{F} \), we have

\[
\| \Delta(\partial^v u(\cdot, y)) \|_{L^2} \lesssim (|v| + 1)! \mathfrak{B}^v \| f \|_{L^2},
\]

where the sequence \( \mathfrak{B} \) is defined in (2.4).

**Lemma 6.3** In the uniform case under Assumptions (U1)–(U3), for every \( f \in L^2(D) \), every \( y \in U \), every \( v \in \mathfrak{F} \) and every \( h > 0 \), we have

\[
\| \partial^v (u - u_h)(\cdot, y) \|_V = \| \nabla(\partial^v (u - u_h)(\cdot, y)) \|_{L^2} \lesssim h (|v| + 2)! \mathfrak{B}^v \| f \|_{L^2},
\]

where the sequence \( \mathfrak{B} \) is defined in (2.4).
Lemma 6.4 In the uniform case under Assumptions (U1)–(U3), for every \( f \in L^2(D) \), every \( G \in L^2(D) \), every \( y \in U \), every \( \nu \in F \) and every \( h > 0 \), we have

\[
|\partial^\nu G((u - u_h)(\cdot, y))| \lesssim h^2 (|\nu| + 5)! \bar{b}^\nu \| f \|_{L^2} \| G \|_{L^2},
\]

where the sequence \( \bar{b} \) is defined in (2.4).

The proofs of these three lemmas are given in Sect. 9. Lemma 6.2 is proved by induction on \(|\nu|\), similarly to the proof of Lemma 6.1, but this time by differentiating the strong form (2.1) of the PDE and obtaining estimates involving the Laplacian of the mixed derivatives of \( u \); the proof makes use of Lemma 6.1. Lemma 6.3 is also proved by induction on \(|\nu|\), but by differentiating the equation representing Galerkin orthogonality; the proof makes use of Lemma 6.2. Lemma 6.4 is proved using a duality argument and it makes use of Lemma 6.3.

For \( \ell \geq 1 \) we obtain from Lemma 6.4 the norm estimate

\[
\| G(u^\ell_{h\ell} - u^{\ell-1}_{h\ell-1}) \|_{s, q} \lesssim h_{\ell-1}^2 \| f \|_{L^2} \| G \|_{L^2} \left( \sum_{u \subseteq \{1, s\}} [(|u| + 5)!] \prod_{j \in u} j^2 \gamma_u \right)^{1/2},
\]

which also holds for the case \( \ell = 0 \); see (6.4), if we define \( h_{-1} := 1 \). Combining these with Theorem 5.1 gives

\[
\sum_{\ell=0}^L \mathbb{E} \left[ |(I - Q^\ell_{ran})(G(u^\ell_{h\ell} - u^{\ell-1}_{h\ell-1}))|^2 \right] \lesssim C^2_\nu(\lambda) \| f \|_{L^2}^2 \| G \|_{L^2}^2 \sum_{\ell=0}^L r_{\ell}^{-1/n_{\ell}^{-1/\lambda}} h_{\ell-1}^4,
\]

where

\[
C^2_\nu(\lambda) = \left( 2 \sum_{u \subseteq \{1, s\}} \gamma_u^\nu [\rho(\lambda)]^{|u|} \right)^{1/\lambda} \left( \sum_{u \subseteq \{1, s\}} [(|u| + 5)!] \prod_{j \in u} j^2 \gamma_u \right). 
\]

Here \( C^2_\nu(\lambda) \) is minimized by taking a different set of POD weights

\[
\gamma_u = \left( \frac{(|u| + 5)!}{120} \prod_{j \in u} \frac{\bar{b}_j}{\sqrt{\rho(\lambda)}} \right)^{2/(1+\lambda)}.
\]

Under Assumption (U6), we can prove that if we take

\[
\lambda = \begin{cases} 
\frac{1}{2 - 2\delta} & \text{for some } \delta \in (0, 1/2) \text{ when } p_1 \in (0, 2/3), \\
\frac{p_1}{2 - p_1} & \text{when } p_1 \in (2/3, 1),
\end{cases}
\]

(6.9)
then the constant $C_y^2(\lambda_\ell)$ can be bounded independently of $s$. We point out again that the weights (6.8) have a larger factorial factor than those in [69, Theorem 10] due to differences in the definition of $\bar{b}_j$ and the proof argument.

We summarize the final result in the theorem below.

**Theorem 6.2** In the uniform case under Assumptions (U1)–(U6), for every $f \in L^2(D)$ and every $G \in L^2(D)$, the multi-level algorithm $A_{\text{ran}}^{\text{ML}}(G(u))$ using randomly shifted lattice rules with $s_\ell = s$ for all $\ell$, and with $n_\ell = 2^{m_\ell}$ points constructed from a CBC algorithm with POD weights (6.8)–(6.9), at the pre-computation cost of $\mathcal{O}(s n_\ell \log n_\ell + s^2 n_\ell)$ operations, achieves the mean-square error bound

$$\mathbb{E} \left[ \left( \left( I - A_{\text{ran}}^{\text{ML}}(G(u)) \right)^2 \right) \right] \lesssim s^{-4(1/p_0 - 1)} + h_L^{4} + \sum_{\ell=0}^{L} r_\ell^{-1} n_\ell^{-2 \min(1/p_1 - 1/2, 1 - \delta)} h_{\ell - 1}^{4}, \quad \delta \in (0, \frac{1}{2}),$$

where the implied constant is independent of $s$, $h_\ell$, $r_\ell$ and $n_\ell$.

In [69] it is assumed more generally that $f \in H^{-1+t}(D)$ and $G \in H^{-1+t'}$ for $t, t' \in [0, 1]$. Moreover, the analysis allowed for different $s_\ell$ at different levels to arrive at the mean-square error bound

$$s_L^{-4(1/p_0 - 1)} + h_L^{4(2t+t')} + \sum_{\ell=0}^{L} r_\ell^{-1} n_\ell^{-2 \min(1/p_1 - 1/2, 1 - \delta)} (\theta_{\ell - 1} s_{\ell - 1}^{-2(1/p_0 - 1/p_1)} + h_{\ell - 1}^{2(t+t')}),$$

for $\delta \in (0, 1/2)$, where $\theta_{\ell - 1}$ is 0 if $s_\ell = s_{\ell - 1}$ and is 1 otherwise. For this analysis a modified sequence $\bar{b}_j \coloneqq \max(\bar{b}_j, b_j^{p_0/p_1})$ is needed in place of $\bar{b}_j$ in the choice of weights $\gamma_{i_\ell}$; see [69, Theorem 11].

Since the $s_\ell$ are potentially different for different $\ell$, the strategy in [69] is to first choose $s_\ell$ in relation to $h_\ell \propto 2^{-\ell}$ to balance the truncation error and FE error on each level. Observe from the error bound that the truncation error between the levels decays significantly more slowly than the truncation error at the highest level, i.e., comparing the exponent $2(1/p_0 - 1/p_1)$ with $4(1/p_0 - 1)$. For this reason one ends up with a sequence of $s_\ell$ that is strictly increasing initially but then the remaining $s_\ell$ become constant and equal to $s_L$.

Then, with $r_\ell = r$ assumed to be a fixed constant, and with the assumption that the cost of the algorithm $A_{\text{ran}}^{\text{ML}}(G(u))$ is $\mathcal{O}(\sum_{\ell=0}^{L} s_\ell n_\ell h_\ell^{-d})$ operations, a Lagrange multiplier argument is used in [69] to choose $n_\ell$ in relation to $h_\ell$ to minimize the mean-square error subject to a fixed cost. Finally the value of $L$ is chosen so that the combined error meets the required threshold. If the mean-square error is $\mathcal{O}(\varepsilon^2)$, then the cost of the multi-level algorithm can be expressed as $\mathcal{O}(\varepsilon^{-\tau} (\log \varepsilon^{-1}))$, with $\tau$ much smaller than the corresponding exponent in the single-level algorithm in most cases; see [69, Theorem 12].
6.3 First-Order, Single-Level, Lognormal

In the lognormal case, the mean-square error for our single-level algorithm with randomly shifted lattice rules can be expressed as in (6.1), but instead of using (6.2), here we split the first term into

$$I(G(u - u_h^s)) = I(G(u - u_h)) + I(G(u_h - u_h^s)),$$

where we estimate the FE error using an analogous result to (3.11) and we estimate the truncation error using (3.18). We use Theorem 5.2 for the QMC error, namely the second term in (6.1). As in the uniform case, we need an estimate of $\|G(u_h^s)\|_{s, \gamma}$ but now with the norm defined by (5.4). Using again (6.3), we conclude that we need a regularity result analogous to Lemma 6.1.

Lemma 6.5 In the lognormal case under Assumptions (L1) and (L2), with the sequence $\beta$ defined in (2.8), for every $f \in V^*$, every $y \in U_\beta$ and every $v \in \mathcal{F}$, we have

$$\|\partial^v u(\cdot, y)\|_V = \|\nabla(\partial^v u(\cdot, y))\|_{L^2} \leq \frac{|v|!}{(\ln 2)^{|v|}} \beta^v \|f\|_{V^*, \text{min}(y)}.$$ 

The same estimate holds when the exact solution $u$ is replaced by $u_h^s$.

The proof is given in Sect. 9; see also [46, Theorem 14]. It is proved by obtaining an estimate on $\|a^{1/2}(\cdot, y) \nabla(\partial^v u(\cdot, y))\|_{L^2}$ using induction on $|v|$.

Comparing with the corresponding result for the uniform case, the critical difference in the lognormal case is that $a_{\text{min}}(y)$ depends on $y$, thus making the estimation of $\|G(u_h^s)\|_{s, \gamma}$ more complicated. In the following, we argue along the lines of the proofs of [46, Theorems 16 and 20 and Corollary 21]; however, the estimates are slightly different since there was an omission of a factor in [46] which meant that the formula for the weights $\gamma_u$ is different and the implied constant in Theorem 6.3 is smaller here compared with [46].

We can use (6.3) and Lemma 6.5, together with the bound

$$\frac{1}{a_{\text{min}}(y)} \leq \frac{1}{\min_{x \in D} a_0(x)} \prod_{j \geq 1} \exp(\beta_j |y_j|),$$

as well as

$$\int_{-\infty}^{\infty} \exp(\beta_j |y_j|) \phi(y_j) \ dy_j = 2 \exp \left( \frac{\beta_j^2}{2} \right) \Phi(\beta_j) \leq \exp \left( \frac{\beta_j^2}{2} + 2\beta_j / \sqrt{2\pi} \right),$$

(6.10)
to obtain an estimate of the norm (5.4)

$$
\| G(u_h) \|_{s, \gamma}^2 \leq \left( \frac{\| f \|_{V^*} \| G \|_{V^*}}{\min_{x \in D} a_0(x)} \right)^2 \prod_{j=1}^{s} \left[ 2 \exp(\beta_j^2/2) \Phi(\beta_j) \right]^2 
\times \sum_{u \subseteq \{1:s\}} (|u|!)^2 \gamma_u (\ln 2)^2|u| \prod_{j \in u} \frac{\beta_j^2}{2 \exp(\beta_j^2/2) \Phi(\beta_j)} \int_{-\infty}^{\infty} \exp(2\beta_j |y_j|) \sigma_j^2(y_j) \, dy_j.
$$

(6.11)

This leads us to choose the weight functions $\sigma_j$ to be exponential functions given by (5.5), with $\alpha_j > \beta_j$, so that

$$
\int_{-\infty}^{\infty} \exp(2\beta_j |y_j|) \sigma_j^2(y_j) \, dy_j = \frac{1}{\alpha_j - \beta_j},
$$

(6.12)

and thus

$$
\| G(u_h) \|_{s, \gamma}^2 \lesssim \sum_{u \subseteq \{1:s\}} (|u|!)^2 \gamma_u (\ln 2)^2|u| \prod_{j \in u} \frac{\beta_j^2}{2 \exp(\beta_j^2/2) \Phi(\beta_j)} \left( \frac{\beta_j}{\sqrt{(\alpha_j - \beta_j) \rho_j(\lambda)}} \right)^{2/(1+\lambda)},
$$

(6.13)

where the implied constant is independent of $s$ under Assumption (L1).

Combining this with Theorem 5.2 and following the strategy for choosing weights $\gamma_u$ in the uniform case, we obtain

$$
\gamma_u = \left( |u|! \prod_{j \in u} \frac{\beta_j}{2(\ln 2) \exp(\beta_j^2/2) \Phi(\beta_j) \sqrt{(\alpha_j - \beta_j) \rho_j(\lambda)}} \right)^{2/(1+\lambda)},
$$

(6.14)

with $\lambda$ chosen as in (6.6) but with $p_0$ as in Assumption (L5), where $\rho_j(\lambda)$ is as given in Theorem 5.2. We substitute this choice of weights into Theorem 5.2 and (6.13), and then minimize the resulting bound with respect to the parameters $\alpha_j$. As argued in [46, Corollary 21], this corresponds to minimizing $[\rho_j(\lambda)]^{1/\lambda}/(\alpha_j - \beta_j)$ with respect to $\alpha_j$, and yields

$$
\alpha_j = \frac{1}{2} \left( \beta_j + \sqrt{\beta_j^2 + 1 - \frac{1}{2\lambda}} \right).
$$

(6.15)

Note that $2 \exp(\beta_j^2/2) \Phi(\beta_j) \to 1$ as $\beta_j \to 0$, and $\rho_j(\lambda)$ is also bounded away from zero as $\beta_j \to 0$. Moreover, $\alpha_j - \beta_j > 0$ is minimized by the largest $\beta_j$. Thus, the summability of the product factors in (6.14) is determined by the summability of the numerator $\beta_j$. Arguing as in the proof of [46, Theorem 20], we conclude that

$$
\mathbb{E} \left[ \| (I - Q_{\text{ran}}) (G(u_h)) \|^2 \right] \lesssim r^{-1} n^{-1/\lambda},
$$

where the implied constant is independent of $s$ under Assumption (L5).

We summarize the final result in the theorem below.
Theorem 6.3 In the lognormal case under Assumptions (L1)–(L5), for every \( f \in L^2(D) \) and every \( G \in L^2(D) \), the single-level algorithm \( A_{\text{ran}}^{\text{SL}}(G(u)) \) using a randomly shifted lattice rule with \( n = 2^m \) points constructed from a CBC algorithm with POD weights (6.14)–(6.15) together with (6.6), at the pre-computation cost of \( O(s n \log n + s^2 n) \) operations, achieves the mean-square error bound

\[
\mathbb{E} \left[ |(I - A_{\text{ran}}^{\text{SL}})(G(u))|^2 \right] \lesssim s^{-2\chi} + h^4 + r^{-1} n^{-2 \min(1/p_0 - 1/2, 1 - \delta)}, \quad \delta \in \left( 0, \frac{1}{2} \right),
\]

with \( \chi \) as in (3.18), where the implied constant is independent of \( s, h, r \) and \( n \).

Similarly to the uniform case, we can treat \( r \) as a fixed constant and choose \( s, h \) and \( n \) to achieve \( O(\varepsilon^2) \) mean-square error.

We remark that an alternative analysis can be carried out following Theorem 5.3 instead of Theorem 5.2, by choosing different weight functions \( \sigma_j \) in (6.11). This would yield a different formula (6.12), a different bound for the norm (6.13) and a different choice of weights (6.14). This is work in progress and could potentially lead to better constants in the bounds and therefore better numerical results.

6.4 First-Order, Multi-Level, Lognormal

For our multi-level algorithm in the lognormal case, the mean-square error with randomly shifted lattice rules can again be expressed as (6.7). We now need to estimate \( \|G(u_h - u_{h-1})\|_{L^2} \) with the norm defined by (5.4). We need regularity results similar to Lemmas 6.2, 6.3 and 6.4 in the uniform case.

Lemma 6.6 In the lognormal case under Assumptions (L1)–(L3), with the sequence \( \mathbf{B} \) defined in (2.9), for every \( f \in L^2(D) \), every \( y \in U_{\mathbf{B}^{-}} \) and every \( v \in \mathbb{F} \), we have

\[
\|\Delta(\partial^v u(\cdot, y))\|_{L^2} \lesssim T(y) (|v| + 1)! 2^{|v|} \|\mathbf{B}^v\| \|f\|_{L^2},
\]

where \( T(y) \) is defined in (3.13).

Lemma 6.7 In the lognormal case under Assumptions (L1)–(L3), with the sequence \( \mathbf{B} \) defined in (2.9), for every \( f \in L^2(D) \), every \( y \in U_{\mathbf{B}^{-}} \), every \( v \in \mathbb{F} \), for every \( h > 0 \), we have

\[
\|a^{1/2}(\cdot, y) \nabla(\partial^v (u - u_h)(\cdot, y))\|_{L^2} \lesssim h T(y) a_{\max}^{1/2}(y) \frac{(|v| + 2)!}{2} 2^{|v|} \|\mathbf{B}^v\| \|f\|_{L^2},
\]

where \( T(y) \) is defined in (3.13).

Lemma 6.8 In the lognormal case under Assumptions (L1)–(L3), with the sequence \( \mathbf{B} \) defined in (2.9), for every \( f \in L^2(D) \), every \( G \in L^2(D) \), every \( y \in U_{\mathbf{B}^{-}} \), every \( v \in \mathbb{F} \), for every \( h > 0 \), we have

\[
|\partial^v G((u - u_h)(\cdot, y))| \lesssim h^2 T^2(y) a_{\max}(y) (|v| + 5)! 2^{|v|} \|\mathbf{B}^v\| \|f\|_{L^2} \|G\|_{L^2},
\]

where \( T(y) \) is defined in (3.13).
The proof of these three lemmas are given in Sect. 9; see also [66]. These proofs follow the same general steps as in the proofs for Lemmas 6.2, 6.3 and 6.4 in the uniform case. The main challenge is in identifying which power of $a(\cdot, y)$ to be included in the $L^2$ norm estimate so that the recursion works. Lemma 6.6 is proved by obtaining an estimate on $\|a^{-1/2}(\cdot, y)\nabla \cdot (a(\cdot, y) \nabla (\partial^\nu u(\cdot, y)))\|_{L^2}$ using induction on $|\nu|$; the proof makes use of an estimate on $\|a^{1/2}(\cdot, y) \nabla (\partial^\nu u(\cdot, y))\|_{L^2}$ which was established in the proof of Lemma 6.5. Lemma 6.7 is also obtained by induction on $|\nu|$; the proof makes use of the estimate established in proof of Lemma 6.6. The proof of Lemma 6.8 makes use of Lemma 6.7.

As in [66] we can show that

$$T^2(y) a_{\text{max}}(y) \leq \left(1 + \frac{\|\nabla a_0\|_{L^\infty}}{\min_{x \in T} a_0(x)}\right)^2 \frac{\|a_0\|_{L^\infty}^3}{(\min_{x \in T} a_0(x))^4} \prod_{j \geq 1} \exp(9\beta_j |y_j|).$$

Thus, with the weight functions $\varpi_j$ in the norm (5.4) given by (5.5) where $\alpha_j > 9\beta_j$, and using (6.10) and (6.12) with $\beta_j$ replaced by $9\beta_j$, we obtain from Lemma 6.8 the estimate for $\ell \geq 1$

$$\|G(u_{h_{\ell}}^s - u_{h_{\ell-1}}^s)\|_{S,Y}^2 \lesssim h_{\ell-1}^4 \sum_{u \subseteq \{1:s\}} [(|u| + 5)!]^2 2^{2|u|} \prod_{j \in u} \frac{\tilde{\beta}_j^2}{2 \exp(81\beta_j^2/2)\Phi(9\beta_j)]^2 (\alpha_j - 9\beta_j)},$$

where the implied constant is independent of $s$ under Assumption (L1). This also holds for $\ell = 0$ with $h_{-1} := 1$; see (6.13). Combining this with Theorem 5.2 and following the same strategy for choosing weights as in the previous subsections, we conclude that

$$\sum_{\ell = 0}^L \mathbb{E} \left[ |(I - Q_{\text{ran}}^\ell)(G(u_{h_{\ell}}^s - u_{h_{\ell-1}}^s))|^2 \right] \lesssim \sum_{\ell = 0}^L r_{\ell}^{-1} n_{\ell}^{-1/\lambda} h_{\ell-1}^4,$$

where the implied constant is independent of $s$ under Assumption (L6), if we take $\lambda$ as in (6.9) and if we choose the weights

$$\gamma_u = \left(\frac{(|u| + 5)!}{120} \prod_{j \in u} \frac{\tilde{\beta}_j}{\exp(81\beta_j^2/2)\Phi(9\beta_j)} (\alpha_j - 9\beta_j) \rho_j(\lambda)\right)^{2/(1+\lambda)}.$$  

A generalization of [46, Corollary 21] yields the choice

$$\alpha_j = \frac{1}{2} \left(9\beta_j + \sqrt{81\beta_j^2 + 1 - \frac{1}{2\lambda}}\right).$$  

We summarize the final result in the theorem below.
Theorem 6.4 In the lognormal case under Assumptions (L1)–(L6), for every \( f \in L^2(D) \) and every \( G \in L^2(D) \), the multi-level algorithm \( A_{\text{ran}}^{\text{ML}}(G(u)) \) using randomly shifted lattice rules with \( s_\ell = s \) for all \( \ell \), and with \( n_\ell = 2^{m_\ell} \) points constructed from a CBC algorithm with POD weights (6.16)–(6.17) together with (6.9), at the pre-computation cost of \( \mathcal{O}(s n_\ell \log n_\ell + s^2 n_\ell) \) operations, achieves the mean-square error bound

\[
\mathbb{E} \left[ |(I - A_{\text{ran}}^{\text{SL}})(G(u))|^2 \right] \lesssim s^{-4(1/p_0 - 1)} + h^4 + r^{-1} n^{-2 \min(1/p_0 - 1/2, 1 - \delta)}, \quad \delta \in (0, \frac{1}{2}),
\]

where the implied constant is independent of \( s, h, \) and \( n_\ell \).

We can carry out a cost versus error analysis as in the uniform case to obtain the optimal choice of values \( s, h, \) and \( n_\ell \). We could also adjust the levels adaptively in practice, as described in [66]; see also [42].

As in the single-level algorithm, we could choose the weight functions \( \sigma_j \) differently by following Theorem 5.3 and this would yield a different choice of weights (6.16).

6.5 Higher-Order, Single-Level, Uniform

The error for our single-level algorithm with interlaced polynomial lattice rules can be expressed as

\[
(I - A_{\text{ran}}^{\text{SL}})(G(u)) = I \left( G(u - u^s) \right) + \left( G(u^s - u^s_h) \right) + \left( I - Q \right) \left( G(u^s_h) \right).
\]

The truncation error can be estimated using (3.16) as before. The FE error of higher order can be estimated using (3.15). For the QMC error we use Theorem 5.4, and therefore, we need an estimate on the norm \( \|G(u^s_h)\|_{s, \alpha, \gamma} \) defined by (5.10). Due to linearity and boundedness of \( G \) we have \( |\partial^\nu G(u^s_h(\cdot, y))| \leq \|G\|_{V^*} \|\partial^\nu u^s_h(\cdot, y)\|_V \), where the last norm can be estimated as in Lemma 6.1. Thus, we have

\[
\|G(u^s_h)\|_{s, \alpha, \gamma} \leq \frac{\|f\|_{V^*} \|G\|_{V^*}}{a_{\min}} \sup_{u \in \{1\}^s} \frac{1}{\gamma_u} \sum_{u \subseteq \tau, \tau \in \{1\}^s \setminus \{0\}} \sum_{u \in \{1\}^s \setminus \{0\}} |(\alpha_0, \tau \setminus \{0\})! b^{(\alpha_0, \tau \setminus \{0\})} | v_u^! \prod_j (2^{\delta(v_j, \alpha)} b_j^{v_j^*}),
\]

where \( \delta(v_j, \alpha) = 1 \) if \( v_j = \alpha \) and is 0 otherwise.

We choose SPOD weights (5.9)

\[
\gamma_u = \sum_{\nu_u \in \{1\}^s \setminus \{0\}} |v_u^! \prod_j (2^{\delta(v_j, \alpha)} b_j^{v_j})
\]

so that \( \|G(u^s_h)\|_{s, \alpha, \gamma} \leq \|f\|_{V^*} \|G\|_{V^*}/a_{\min} \). Substituting these weights into Theorem 5.4 and following the arguments in [23, pp. 2694–2695], we take \( \lambda = p_0 \) and

\[
\Theta \text{ Springer}
\]
\[ \alpha = \lfloor 1/p_0 \rfloor + 1, \text{ and conclude that} \]
\[ |(I - Q)(G(u_h))| \lesssim n^{-1/p_0}, \]

with the implied constant independent of \( s \).

We summarize the final result in the theorem below.

**Theorem 6.5** In the uniform case under Assumptions \((U1)-(U5)\), for every \( f \in \mathcal{F}_t^* \) and every \( G \in \mathcal{F}_t^* \) with \( t, t' \geq 0 \), the single-level algorithm \( A_{\text{det}}^\text{SL}(G(u)) \) using an interlaced polynomial lattice rule with interlacing factor \( \alpha = \lfloor 1/p_0 \rfloor + 1 \geq 2 \) and with \( n = 2^m \) points constructed from a CBC algorithm with SPOD weights \((6.18)\), at the pre-computation cost of \( \mathcal{O}(\alpha s n \log n + \alpha^2 s^2 n) \) operations, achieves the error bound

\[ |(I - A_{\text{det}}^\text{SL})(G(u))| \lesssim s^{-2(1/p_0 - 1)} + h^{t + t'} + n^{-1/p_0}, \]

where the implied constant is independent of \( s, h \) and \( n \).

Again we can choose \( s, h, n \) to achieve \( \mathcal{O}(\varepsilon) \) error.

### 6.6 Higher-Order, Multi-Level, Uniform

The error for our multi-level algorithm with interlaced polynomial lattice rules can be expressed as

\[ (I - A_{\text{det}}^\text{ML})(G(u)) \]
\[ = I(G(u - u_s)) + I(G(u_s - u_{s-1})) + \sum_{\ell=0}^L (I - Q^\ell)(G(u_{s_\ell} - u_{s_{\ell-1}})), \]

where \( Q^\ell \) denotes an interlaced polynomial lattice rule in \( s_\ell \) dimensions with \( n_\ell \) points. Again for simplicity we restrict our discussion to the case \( s_\ell = s \) for all \( \ell \). So we need an estimate on the norm \( \|G(u_{h_\ell} - u_{h_{\ell-1}})\|_{s,\alpha,\gamma} \). From Lemma 6.4 we conclude that for \( \ell \geq 1 \) we have

\[ \|G(u_{h_\ell} - u_{h_{\ell-1}})\|_{s,\alpha,\gamma} \]
\[ \lesssim h_{\ell-1}^2 \|f\|_{L^2} \|G\|_{L^2} \sup_{u \subseteq \{1:s\}} \frac{1}{Y_u} \sum_{u \subseteq u_r \subseteq \{1:|u|\}} \sum_{\alpha} (\|\alpha \| \|u_r\| \|0\| + 5)! B^{(\alpha,0)} \]
\[ = h_{\ell-1}^2 \|f\|_{L^2} \|G\|_{L^2} \sup_{u \subseteq \{1:s\}} \frac{1}{Y_u} \sum_{u \subseteq \{1:|u|\}} (|v_u| + 5)! \prod_{j \in u} (2^{5(v_j,0)} B_{v_j}). \]

We therefore choose SPOD weights

\[ \gamma_u = \sum_{v_u \subseteq \{1:|u|\}} \frac{(|v_u| + 5)!}{120} \prod_{j \in u} (2^{5(v_j,0)} B_{v_j}). \quad (6.19) \]
Substituting these weights into Theorem 5.4 and following similar arguments as those before, we take now $\lambda = p_1$ and $\alpha = \lceil 1/p_1 \rceil + 1$, and conclude that the convergence rate is $n^{-1/p_1}$, with the implied constant independent of $s$.

**Theorem 6.6** In the uniform case under Assumptions (U1)–(U6), for every $f \in L^2(D)$ and every $G \in L^2(D)$, the multi-level algorithm $A_{\text{det}}^{\text{ML}}(G(u))$ using interlaced polynomial lattice rules with interlacing factor $\alpha = \lceil 1/p_1 \rceil + 1 \geq 2$, with $s_\ell = s$ for all $\ell$, and with $n_\ell = 2^{m_\ell}$ points constructed from a CBC algorithm with SPOD weights (6.19), at the pre-computation cost of $O(\alpha s n_\ell \log n_\ell + \alpha^2 s^2 n_\ell)$ operations, achieves the error bound

$$\left| (I - A_{\text{det}}^{\text{ML}})(G(u)) \right| \lesssim s^{-2(1/p_0-1)} + h_L^2 + \sum_{\ell=0}^L n_\ell^{-1/p_1} h_\ell^{-2},$$

where the implied constant is independent of $s$, $h_\ell$ and $n_\ell$. We could consider higher-order FE methods here, but then we would need stronger regularity on $f$ and $G$, as well as analogous results for Lemmas 6.2, 6.3 and 6.4. This is where Assumption (U7) would be needed. We could also allow different $s_\ell$ at different levels. These generalizations are considered in [25] where a comprehensive error versus cost analysis of the multi-level algorithm is provided for the general setting of affine parametric operator equations.

Now we summarize and compare the results from [23,25,68,69] for the uniform case:

First-order single-level [68]

$$s^{-2(1/p_0-1)} + h^{t+t'} + n^{-\min(1/p_0-1/2,1-\delta)} \quad \text{(rms)}.$$

First-order multi-level [69]

$$s_L^{-2(1/p_0-1)} + h_L^{t+t'} + \sum_{\ell=0}^L n_\ell^{-\min(1/p_1-1/2,1-\delta)} \left( \theta_{\ell-1} s_{\ell-1}^{-1/p_0-1/p_1} + h_{\ell-1}^{t+t'} \right) \quad \text{(rms)}.$$

Higher-order single-level [23]

$$s^{-2(1/p_0-1)} + h^{t+t'} + n^{-1/p_0}.$$

Higher-order multi-level [25]

$$s_L^{-2(1/p_0-1)} + h_L^{t+t'} + \sum_{\ell=0}^L n_\ell^{-1/p_0} \left( \theta_{\ell-1} s_{\ell-1}^{-1/p_0-1/p_1} + h_{\ell-1}^{t+t'} \right) .$$
For the first-order results, “rms” indicates that the error is in the root-mean-square sense since we use a randomized QMC method. The higher-order results are deterministic. The results include general parameters $t, t'$ for the regularity of $f$ and $G$: in the first-order results we have $f \in H^{-1+t}(D)$ and $G \in H^{-1+t'}(D)$ for $t, t' \in [0, 1]$, while in the higher-order results we have $f \in X_t^*$ and $G \in X_t'^*$ for integers $t, t' \geq 0$. For the multi-level results we include the analysis for potentially taking different $s_\ell$ at each level. Recall that $\delta \in (0, 1/2)$, and $\theta_{\ell-1}$ is 0 if $s_\ell = s_{\ell-1}$ and is 1 otherwise.

Note that in many applications $p_\ell$ in Assumption (U7) satisfies

$$p_\ell = \frac{p_0}{1 - tp_0/d},$$

which means it can be much bigger than $p_0$. So the higher-order multi-level algorithm does not necessarily lead to improved error bounds.

7 A Practical Guide to the Software for Constructing QMC Points

In this section we explain how to use the code which accompanies this article to construct QMC rules for the different settings which have been discussed in this article. The construction algorithms are all fast component-by-component constructions, using results from [17, 23, 84–86]. For lattice rules the construction algorithm will output the generating vector and for the interlaced polynomial lattice rules the algorithm will output the generating matrices. These generating vectors and matrices can then be used in the provided sample point generators.

For randomly shifted lattice rules we have to construct a good generating vector $z$ for the lattice rule (5.1). The results in Sects. 5 and 6 were stated for $n = 2^m$ points, but as noted before they hold for any prime power (including $n$ being a prime). The construction script expects the prime $p$ (which defaults to 2) and the power $m$ to be given such that the (maximum) number of points is $n = p^m$. The natural thing to do is to construct such rules to be good for any intermediate power of $p$ as an embedded sequence of lattice rules. Such lattice sequences can then be used as a sequence of QMC rule approximations. A construction of such lattice sequences was proposed in [17] and this is the approach followed in the lattice rule construction code.

For interlaced polynomial lattice rules (5.8) we have to construct the associated generating matrices. We fix the base of the finite field to be 2 for practicalities in the construction, and in the generation of the points. The number of points is $n = 2^m$, but, in contrast to the lattice rules, we currently do not provide these as embedded sequences. The construction script will automatically choose a default irreducible polynomial of degree $m$ as the modulus polynomial (which can be overridden by the user). The specific choice of modulus polynomial does not influence the error bound in Theorem 5.4. The output of the script will be both the generating matrices $C_1, \ldots, C_\alpha \in \mathbb{Z}_2^{m \times m}$ of the polynomial lattice rule (5.7) and the generating matrices $B_1, \ldots, B_\beta \in \mathbb{Z}_2^{am \times m}$ of the interlaced polynomial lattice rule (5.8).

Once the generating vector or the generating matrices have been constructed, they are used as input to the corresponding point generator. These point generators are relatively straightforward to program, and their computational cost to generate a point...
is really minor and so can be neglected for practical purposes; in fact they are comparable to the fastest random number generators. To generate lattice points, one only requires an integer multiplication, a modulus operation, and a fixed float multiplication/division per dimension; see (5.1). This is comparable to the cost of a simple LCG (linear congruential generator) per dimension. To generate (interlaced) polynomial lattice points in Gray code ordering, one only requires an xor instruction and a fixed float multiplication/division per dimension; see (5.7) and (5.8). Additionally, a CTZ (count trailing zeros) algorithm is used to determine the column number of the generating matrices to perform each xor instruction, which is available on most CPUs as a machine instruction or can be implemented with a simple algorithm having a fixed low arithmetic cost. This is comparable to the cost of a LFSR (linear feedback shift register) generator per dimension. In the case of randomly shifted lattice rules the points still have to be randomly shifted before being used as quadrature points. As QMC points are naturally enumerated, it is straightforward to parallelize the solving of the different PDE problems and we therefore equip the point generators with an option to start at any offset in the enumeration of the points.

As the theory in Sect. 6 is often quite involved, we extract the essential properties of the analysis here and allow them to be applied to any general integrand which shares similar characteristics. In essence, the analysis in Sect. 6 made use of bounds on the mixed derivatives, \( \partial^\nu \) for \( \nu \in \{0, \ldots, \alpha\}^s \), of the integrands to derive suitable weights \( \gamma_u \). In the uniform case these bounds can be stated as follows.

- Uniform, single-level, with \( F(y) = G(u_h^s(\cdot, y)) \):

\[
|\partial^\nu F(y)| \lesssim |\nu|! \prod_{j=1}^s b_{\nu_j}^{v_j}.
\]

- Uniform, multi-level, with \( F_\ell(y) = G((u_h^s - u_{h_{\ell-1}}^s)(\cdot, y)) \):

\[
|\partial^\nu F_\ell(y)| \lesssim (|\nu| + 5)! \prod_{j=1}^s B_{\nu_j}^{v_j}.
\]

For first-order methods the multi-index \( \nu \) satisfies \( \nu_j \leq 1 \) for all \( j \). In the lognormal case the bounds are not uniformly bounded in \( y \) and can be stated as follows.

- Lognormal, single-level, with \( F(y) = G(u_h^s(\cdot, y)) \):

\[
|\partial^\nu F(y)| \lesssim |\nu|! \prod_{j=1}^s (\beta_j / \ln 2)^{\nu_j} \exp(\beta_j |y_j|).
\]

- Lognormal, multi-level, with \( F_\ell(y) = G((u_h^s - u_{h_{\ell-1}}^s)(\cdot, y)) \):

\[
|\partial^\nu F_\ell(y)| \lesssim (|\nu| + 5)! \prod_{j=1}^s (2\beta_j)^{\nu_j} \exp(9\beta_j |y_j|).
\]
We note that the results in Sect. 6 for the lognormal case only hold for first-order methods with all $\nu_j \leq 1$.

We provide two Python scripts, `lat-cbc.py` and `polylat-cbc.py` (as interfaces to the construction script `spod-cbc.py`), to construct lattice rules and interlaced polynomial lattice rules, respectively, in which the following generalized bound on the mixed derivatives is assumed: for all $\nu \in \{0, \ldots, \alpha\}^s$,

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

for some integers $\alpha \geq 1$ and $a_1 \geq 0$, real numbers $a_2 > 0$, $a_3 \geq 0$ and $d_1 \geq 0$, and a sequence of positive numbers $B_j$, corresponding to the values of $b_j, b_j, \beta_j$ or $\beta_j$, appropriate for the setting; see (2.3), (2.4), (2.8) and (2.9). An overview of all parameters and their description for the `lat-cbc.py` and the `polylat-cbc.py` scripts is given in Table 2.

The summary of bounds from the analysis of Sect. 6 corresponds to taking $d_1 = 1$. For randomly shifted lattice rules the order of convergence is limited to 1 and thus $\alpha = 1$. For interlaced polynomial lattice rules we need $\alpha \geq 2$. The uniform case corresponds to taking $a_3 = 0$. The lognormal case corresponds to taking $a_3 > 0$, with $a_3 = 1$ and $a_3 = 9$ for the single-level and multi-level algorithms, respectively. Our analysis lead to $a_1 = 0$ and $a_1 = 5$ for the single-level and multi-level algorithms, respectively. (Following the proof arguments in [25,69] we could set $a_1 = 3$ for the multi-level algorithms in the uniform case, but the sequence $B_j$ is defined differently.)

We have $a_2 = 1$ in the uniform case, while in the lognormal case we have $a_2 = 1/\ln 2$ and $a_2 = 2$ for the single-level and multi-level algorithms, respectively. To cater for other potential integrands which satisfy the generalized bound (7.1), our scripts can take general values of $a_1, a_2, a_3$ and $d_1$ as input.

To specify the sequence $B_j$ the user has two main options. Either the user provides a Python expression (with access to variables $j$ and $\nu$, to stand for the values of $j$ and $\nu_j$) as the argument to command line option `b`, or the user provides the name of an input file containing numerical values for each of the $B_j$ by means of the `b_file` option. (Other possibilities are available, including a configuration file, but are not discussed here for conciseness.)

We remark that the analysis in Sect. 6 takes into account the truncation from infinite dimensions to $s$ dimensions. Therefore, it is essential to have an idea of the summability of the infinite sequence $B_j$. As before we are interested in the value of $p_* \in (0, 1)$ for which $\sum_{j=1}^{\infty} B_j^{p_*} < \infty$ and we would like $p_*$ to be as small as possible. Here it is more convenient to work with the reciprocal value, denoted by $d_2 > 1$, and we call this the “decay” of the sequence $B_j$.

The theoretical QMC convergence rate in the context of PDE problems, with the implied constant independent of $s$, is roughly of order $n^{-\min(1,d_2^{-1/2})}$ for randomly shifted lattice rules, and $n^{-\min(\alpha,d_2)}$ for interlaced polynomial lattice rules with interlacing factor $\alpha \geq 2$.

The analysis in Sect. 6 can be extended to handle the bound (7.1) with a general exponent $d_1$ on the factorial factor, provided that $d_2 > d_1$ (to ensure that the implied
### Table 2 Options for the Python scripts `lat-cbc.py` and `polylat-cbc.py`

| Option   | Description                                                                                                                                 |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| `s`      | Number of dimensions                                                                                                                        |
| `m`      | Number of points given by $2^m$ (or $p^m$ in case of optional argument `p` for lattice rules)                                                |
| `p`      | Optional<br>• For lattice rules $n = p^m$, prime, defaults to $p = 2$
• For polynomial lattice rules this is the primitive modulus polynomial of degree $m$ (the code uses a table of default polynomials) |
| `alpha`  | Optional<br>• No effect for lattice rules, $\alpha = 1$
• Integer interlacing factor for polynomial lattice rules, $\alpha \geq 2$ (defaults to $\alpha = 2$) |
| `a1`     | Optional, defaults to 0<br>e.g., in our analysis, $a_1 = 0$ for single-level and $a_1 = 5$ for multi-level |
| `a2`     | Optional, defaults to 1<br>e.g., in our analysis, $a_2 = 1$ for uniform and $a_2 = 1/\ln 2$ for single-level lognormal and $a_2 = 2$ for multi-level lognormal |
| `a3`     | Optional, defaults to 0<br>e.g., in our analysis, $a_3 = 1$ for single-level lognormal and $a_3 = 9$ for multi-level lognormal |
| `d1`     | Optional, defaults to 1<br>Extra power on the factorial factor ($d_1 = 0$ implies product weights)                                             |
| `d2`     | Optional, defaults to 2<br>Decay of the $B_j$ sequence, $d_2 > 1$                                                                            |
| `b`      | Optional, defaults to $c_j^{-d_2}$<br>The $B_j$ sequence as a Python expression, see text (alternatively as numerical values through a file with `b_file`) |
| `c`      | Optional, defaults to 1<br>In case no `b` and `b_file` are given, $B_j$ is set to $c_j^{-d_2}$                                               |
| `b_file` | Optional<br>File name containing numerical values for the sequence $B_j$                                                                   |
| `out`    | Optional<br>Output directory to write results to                                                                                           |

A constant in the error estimate is bounded independently of $s$. The case $d_1 = 0$ will lead to product weights $\gamma_u$, in which case the CBC construction has a lower cost.

### 7.1 Constructing Randomly Shifted Lattice Rules

A synopsis of how to call the Python script to construct a randomly shifted lattice rule with $n = 2^m$ points is

```bash
./lat-cbc.py --s={s} --m={m} [--a1={a1}] [--a2={a2}] [--a3={a3}] --d2={d2} \
[--b="{bound-function}" | --b_file={file_name} | --c={c}]
```
In particular, the default value of $a_3 = 0$ specifies the uniform case, while any value of $a_3 > 0$ specifies the lognormal case. These two cases correspond to the different function space settings in Sects. 5.1 and 5.2.

The construction script will automatically choose the parameter $\lambda \in (1/2, 1]$ from Theorems 5.1 and 5.2 as (slightly different from (6.6) and (6.9))

$$\frac{1}{2\lambda} = \begin{cases} 1 - \delta & \text{if } d_2 \geq 3/2 - \delta, \\ d_2 - \frac{1}{2} & \text{if } d_2 \leq 3/2 - \delta, \end{cases}$$

(7.2)

with $\delta = 0.125$, which yields the theoretical convergence rate of order $n^{-(1-\delta)}$ for $d_2 \geq 3/2 - \delta$ and $n^{-(d_2-1/2)}$ for $d_2 \leq 3/2 - \delta$. Therefore, a correct value of $d_2$ should be provided. (The value of $\delta$ can also be changed by a command line argument.)

For the uniform case the script will use the weights (see (6.5) and (6.8))

$$\gamma_u = \left( \left( \frac{(|u| + a_1)!}{a_1!} \right)^{d_1} \prod_{j \in u} \frac{a_2 B_j}{\sqrt{\rho(\lambda)}} \right)^{2/(1+\lambda)}.$$

For the lognormal case the script will set the parameters $\alpha_j$ in the weight functions (5.5) to be (see (6.15) and (6.17))

$$\alpha_j = \frac{1}{2} \left( a_3 B_j + \sqrt{(a_3 B_j)^2 + 1 - \frac{1}{2\lambda}} \right),$$

and use the weights (see (6.14) and (6.16))

$$\gamma_u = \left( \left( \frac{(|u| + a_1)!}{a_1!} \right)^{d_1} \prod_{j \in u} \frac{a_2 B_j}{2 \exp((a_3 B_j)^2/2) \Phi(a_3 B_j) \sqrt{(\alpha_j - a_3 B_j) \rho_j(\lambda)}} \right)^{2/(1+\lambda)}.$$

We give some examples on how to call the script:

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

```bash
# as above, but multi-level and with bounds from file:
./lat-cbc.py --s=100 --m=10 --a1=5 --d2=3 --b_file=bounds.txt
```

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

```bash
# as above, but multi-level and with bounds from file:
./lat-cbc.py --s=100 --m=10 --a1=5 --a2=2 --a3=9 --d2=3 --b_file=bounds.txt
```

This will produce several files in the output directory. The most important one is $z.txt$ which contains the generating vector. These points then need to be randomly shifted for the theory to apply. In the lognormal case, the randomly shifted points
should be mapped to \( \mathbb{R}^s \) by applying the inverse of the cumulative normal distribution function componentwise.

Codes are available in Python, MATLAB/Octave and C++ to generate lattice points. An example usage in MATLAB is given below:

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

With respect to the multi-level algorithm there are two important features of these lattice rules: They are lattice sequences in terms of the number of points, and they are constructed by a component-by-component algorithm which allows a rule constructed for \( s \) dimensions to be used for a lower number of dimensions. This means the construction only has to be done once for the maximum number of points \( \max_{0 \leq \ell \leq L} n_{\ell} \) and the maximum number of dimensions \( s_L \), since the parameters in (7.1) are the same for all levels.

As we have already mentioned, for the lognormal case there is work in progress on the analysis with a different choice of weight functions \( \varpi_j \), see Theorem 5.3, and this would yield a different choice of weights \( \gamma_u \). We may provide codes for this alternative setting at a later time.

### 7.2 Constructing Interlaced Polynomial Lattice Rules

A synopsis of how to call the Python script to construct an interlaced polynomial lattice rule with \( n = 2^m \) and interlacing factor \( \alpha \geq 2 \) is

```bash
./polylat-cbc.py --s={s} --m={m} --alpha={alpha} [--a1={a1}] [--a2={a2}] \\
[--b="{bound-function}" | --b_file={file_name} | --d2={d2} --c={c}]
```

The construction script will use the weights (see (6.18) and (6.19))

\[
\gamma_u = \sum_{\nu \in \{1, \ldots, \alpha \}^{|u|}} \left( \frac{(|\nu| + a_1)!}{a_1!} \right)^{d_1} \prod_{j \in u} \left( 2^{\delta(v_j, \alpha)} (a_2 B_j)^{v_j} \right).
\]

We expect the theoretical convergence rate to be of order close to \( n^{-\min(\alpha, d_2)} \).

We give some examples on how to call the script:

```bash
## 100-dimensional rule, 2^10 points, interlacing 3 and with specified bounds b:
./polylat-cbc.py --s=100 --m=10 --alpha=3 --b="0.1 * j**-3 / log(j+1)"

## as above, but multi-level and with bounds from file:
./polylat-cbc.py --s=100 --m=10 --alpha=3 --a1=5 --b_file=bounds.txt
```

Several files will be saved into the output directory. The most important one is \( B_s.col \) which contains the generating matrices of the interlaced polynomial lattice.
rule. (Also the generating matrices of the non-interlaced polynomial lattice rule are available in the file Cs.col.)

Codes are available in Python, MATLAB/Octave and C++ to generate these points. For interlacing to work correctly, the product $\alpha m$ should be no more than the number of available bits. We note that the IEEE double precision type only has 53 bits available and MATLAB uses this type to do its calculations. As a compromise (which comes with no guarantee) we load the generating matrices truncated to 53 bits precision, which are available in the file Bs53.col. Similarly, to cater for the extended long double precision in C++ we provide a file Bs64.col. For instance, with interlacing factor 4 we can have up to $2^{13}$ points in MATLAB and up to $2^{16}$ points in C++ using long double. The Python point generator is implemented such that it can use arbitrary precision. One can also change the C++ generator to use arbitrary precision.

Below we give an example to illustrate how to feed the output from the construction script into the point generator. At the same time we experiment on the effect of truncating the generating matrices. First construct the generating matrices using the Python script and then save these points (in long double precision) to the file points.txt using the C++ example program:

```
./polylat-cbc.py --s=10 --m=15 --alpha=4 --b="0.1 * j**-4" --out=.
./digitalseq_b2g <Bs64.col >points.txt
```

In this example of $2^{15}$ points with interlacing factor 4, we need $4 \times 15 = 60$ bits of precision, which can be realized in full in C++ using long double. Now we use the generating matrices (truncated to 53 bits) in MATLAB, make a plot, and compare these points to the full precision points we just created on the command line:

```
load Bs53.col
s = 10; m = 15;
digitalseq_b2g('init0', Bs53) % initialize the procedural generator
Pa = digitalseq_b2g(s, pow2(m-1)); % first half of the points
Pb = digitalseq_b2g(s, pow2(m-1)); % second half of the points
s1 = 2; s2 = 10;
plot(Pa(s1,:), Pa(s2,:), 'b.', Pb(s1,:), Pb(s2,:), 'r.'
axis([0 1 0 1]); axis square
load points.txt % compare with the C++ generated points in long double
points = points';
Pc = points(:,1:pow2(m-1));
Pd = points(:,pow2(m-1)+(1:pow2(m-1)));
norm(Pc - Pa) % this should be in the order of 1e-14
norm(Pd - Pb) % and this as well...
```

Here the effect of truncating the generating matrices appears to be empirically insignificant, but the higher-order QMC convergence theory no longer applies and there is no guarantee how well they would perform in practice.
7.3 Generating Interlaced Sobol’ Sequences

We note that since the point generators operate using generating matrices, they can be used to generate any other digital sequence, interlaced or not. On the Web site we provide the generating matrices for an implementation of the Sobol’ sequence from [63] with 21201 dimensions (as the file sobol_cs.col), as well as the generating matrices for interlaced Sobol’ sequences with interlacing factor \( \alpha = 2, 3, 4, 5 \) (e.g., sobol_alpha3_Bs53.col). An example usage in MATLAB to generate the points is given below:

```matlab
load sobol_alpha3 Bs53.col % load generating matrices
digitalseq_b2g('init0', sobol_alpha3 Bs53) % initialize the procedural generator
Pa = digitalseq_b2g(10, 1024); % first 1024 10-dimensional points
Pb = digitalseq_b2g(10, 1024); % next 1024 10-dimensional points
```

8 Concluding Remarks

In this article we gave a survey of the results from \([23, 25, 46, 66, 68, 69]\) on the application of QMC methods to PDEs with random coefficients, in a unified view. We outlined three weighted function space settings for analyzing randomly shifted lattice rules (first-order) in both the uniform and lognormal cases, and interlaced polynomial lattice rules (higher-order) in the uniform case. At present there is no QMC theory that can give higher-order convergence for the lognormal case.

We summarized the error analysis for single-level and multi-level algorithms based on these QMC methods, in conjunction with FE methods and dimension truncation. The key step of the analysis is to obtain bounds on the appropriate weighted norm of the integrand, i.e., \( G(uh^\beta) \) for the single-level algorithm and \( G(uh^\beta - uh^{\beta-1}) \) for the multi-level algorithm. We discussed the strategy to obtain suitable weights \( \gamma_u \) for the function space setting and arrived at weights of POD or SPOD form. These weights are to be fed into the CBC construction of QMC points tailored to the PDE problems. This survey is augmented with code to construct such tailored QMC points and we explained how to do this in Sect. 7.

The combination of a particular family of QMC methods with a specific function space setting, and the careful designing of POD or SPOD weights \( \gamma_u \), means that we obtain QMC error bounds that are independent of the truncation dimension, while optimizing on the theoretical convergence rates under minimal assumptions on the PDE problems. Indeed, we could consider other classes of QMC methods, or use the same QMC methods but construct them with weights that are not as prescribed here; however, we might not achieve the same theoretical error bounds. For example, we recall from Sect. 6.1 that in the uniform case with randomly shifted lattice rules we can achieve nearly first-order convergence if \( p_0 \leq 2/3 \), with \( p_0 \) from Assumption (U5). As pointed out in [68], we could consider randomly shifted lattice rules constructed with product weights (instead of POD weights), or the deterministic lattice rules constructed following [62], or Niederreiter and Sobol’ sequences following the analysis in [105], but then to achieve nearly first-order convergence we would require, respectively, \( p_0 \leq 1/2, p_0 \leq 1/2 \) and \( p_0 \leq 1/3 \), meaning that a stronger assumption on the PDE
problem is required to achieve the same convergence rate. More strikingly, we recall from Sect. 6.5 that in the uniform case we can construct interlaced polynomial lattice rules with interlacing factor 2 to achieve first-order convergence when \( p_0 < 1 \), which is a much weaker condition on the PDE problem.

We already pointed out that the uniform framework can be extended to general affine parametric operator equations; see [23,25,93]. Thus, the QMC strategy in this article applies to a wide range of PDE problems including, e.g., stationary and time-dependent diffusion in random media [15], wave propagation [59], parametric nonlinear PDEs [13], and optimal control problems for uncertain systems [64].

There may be additional properties of the functions \( \psi_j \) in (2.2) or \( \xi_j \) in (2.6) that could be exploited to improve the results. For example, in [69] a special orthogonality property for multi-resolution function systems was used to obtain a better dimension truncation estimate for the multi-level error analysis. In a different direction, [23, Theorem 3.2] pointed out that if we were to replace the \( |\nu| = (\sum_{j \geq 1} \nu_j)! \) factor in Lemma 6.1 by \( \prod_{j \geq 1} \nu_j! \), then our analysis would yield product weights \( \gamma_u \) rather than POD or SPOD weights, and this would significantly reduce the CBC construction cost.

We have assumed in this article that the cost for our single-level algorithm is

\[
\mathcal{O}(n s h^{-d}) \quad (8.1)
\]

operations, based on \( n \) instances of FE discretizations where the cost for assembling the stiffness matrix is \( \mathcal{O}(s h^{-d}) \) operations, with the \( \mathcal{O}(s) \) factor coming from the number of KL terms in the coefficient \( a(\mathbf{x}, \mathbf{y}) \). An analogous cost model is assumed for the multi-level algorithm, and we argued that the values of \( n_\ell, s_\ell, h_\ell \) should be chosen by minimizing the error for a fixed cost. Understandably this latter optimization depends crucially on the cost model. Below we discuss two related strategies to reduce cost.

We mentioned the “circulant embedding” strategy for the lognormal case, see [47, 48], but did not go into any details in this article. Roughly speaking, the idea is to sample the random field only at a discrete set of grid points with respect to the covariance matrix, where the number of grid points is of the same order as the number of FE nodes. Then the problem of generating samples turns into a matrix factorization problem which can be done in \( \mathcal{O}(h^{-d} (\log h^{-d})) \) operations using FFT, by embedding the covariance matrix in a larger but circulant matrix (some further padding may be required to ensure positive definiteness). With this strategy the cost becomes

\[
\mathcal{O}(n h^{-d} (\log h^{-d}))
\]

operations, where we effectively replaced the \( \mathcal{O}(s) \) factor in (8.1) by \( \mathcal{O}(\log h^{-d}) \).

The other strategy to reduce cost is the “fast QMC matrix–vector multiplication”; see [24], which exploits a certain structure in the QMC point set. By choosing suitable QMC point sets, and by formulating the QMC quadrature computation as a matrix–vector multiplication with a circulant matrix obtained by indexing the QMC points in a certain way, the cost becomes

\[
\mathcal{O}(n (\log n) h^{-d})
\]
operations by using FFT, which essentially means that the $O(s)$ factor in (8.1) is replaced by $O(\log n)$. Unfortunately, this strategy does not work with randomly shifted lattice rules or interlaced polynomial lattice rules considered in this article, because both randomization and interlacing destroy the required structure in the QMC point sets to yield a circulant matrix. However, this strategy is compatible with “tent transformed” lattice rules or polynomial lattice rules, which are deterministic QMC rules that can achieve nearly first- or second-order convergence; see [29,45]. The error analysis of applying tent transformed lattice rules or polynomial lattice rules for the PDE problems is work in progress.

The above three strategies for cost reduction (namely multi-level algorithms, circulant embedding and fast QMC matrix–vector multiplication) are not mutually exclusive and they could potentially be combined to have a compounding effect in reducing cost. There is also a generalization of the multi-level concept called “multi-index” [50], which is in some sense related to “sparse grid techniques” [8]. Note that each strategy has its prerequisite: multi-level algorithm requires stronger regularity assumptions on the PDE problem, circulant embedding requires stationary covariance functions, while fast QMC matrix–vector multiplication requires a certain structure in the QMC point set. It would be interesting to see which strategy or which combination of strategies yields the most effective reduction in cost under different scenarios.

Now we make some remarks on theory versus practice. Although the careful tuning of weights $\gamma_u$ played a significant role in our analysis and affected the theoretical QMC convergence rates, the outcomes from numerical experiments so far have been inconclusive. We have seen that some “off the shelf” lattice rules (i.e., lattice rules constructed with product weights chosen to have some generic algebraic or geometric decay) perform just as well as those lattice rules which are tailored to the PDE problems. On the other hand, we have also seen that some badly tuned lattice rules (e.g., when the weights $\gamma_u$ are badly scaled) can perform poorly.

We also noted that the theoretical convergence rates are not always reflected in the computations. In the lognormal computations in [46] we see that the convergence rates are not so much influenced by the smoothness parameter $\nu$ of the Matérn covariance function as the theory predicted. Rather, it is the variance and the correlation length that affect the empirical convergence rates.

The numerical experiments for the lognormal case in [47] were obtained using randomly digitally shifted Sobol’ points with no theory, yet the results were very encouraging. One could also try tent transformed Sobol’ points or interlaced Sobol’ points. We suspect that they may work reasonably well in practice, even though at present we are lacking a strong supporting theory. A brief explanation on how to generate interlaced Sobol’ sequences from the code can be found in Sect. 7.

Finally, if the story in this survey article sounds incomplete, we hope the reader will understand that we are trying to tell a story that is changing underneath us, even as we write. We live in interesting times!

Acknowledgements We graciously acknowledge many insightful discussions and valuable comments from our collaborators Josef Dick, Mahadevan Ganesh, Thong Le Gia, Alexander Gilbert, Ivan Graham, Yoshihito Kazashi, James Nichols, Pieterjan Robbe, Robert Scheichl, Christoph Schwab and Ian Sloan. We especially thank Mahadevan Ganesh for suggesting an alternative proof strategy to improve some
existing estimates. We are also grateful for the financial supports from the Australian Research Council (FT130100655 and DP150101770) and the KU Leuven research fund (OT:3E130287 and C3:3E150478).

9 Appendix: Selected Proofs

In this section we provide the proofs for Lemmas 6.1–6.8. For simplicity of presentation, in the proofs we will often omit the arguments \(x\) and \(y\) in our notation. We start by collecting some identities and estimates that we need for the proofs.

We will make repeated use of the Leibniz product rule

\[
\partial^\nu(AB) = \sum_{m \leq \nu} \binom{\nu}{m} (\partial^m A) (\partial^{\nu-m} B),
\]

(9.1)

and the identity

\[
\nabla \cdot (A \nabla B) = A \Delta B + \nabla A \cdot \nabla B.
\]

(9.2)

We also need the combinatorial identity

\[
\sum_{m \leq \nu} \binom{\nu}{m} = \binom{|\nu|}{i},
\]

(9.3)

which follows from a simple counting argument (i.e., consider the number of ways to select \(i\) distinct balls from some baskets containing a total number of \(|\nu|\) distinct balls). The identity (9.3) is used to establish the following identities

\[
\sum_{m \leq \nu} \binom{\nu}{m} |m|! |\nu - m|! = (|\nu| + 1)!,
\]

(9.4)

\[
\sum_{m \leq \nu} \binom{\nu}{m} |m|! (|\nu - m| + 1)! = \frac{(|\nu| + 2)!}{2},
\]

(9.5)

\[
\frac{1}{2} \sum_{m \leq \nu} \binom{\nu}{m} (|m| + 2)! (|\nu - m| + 2)! = \frac{(|\nu| + 5)!}{120}.
\]

(9.6)

Additionally, we need the recursive estimates in the next two lemmas. The proofs can be found in [25] and [66], respectively.

Lemma 9.1 Given a sequence of non-negative numbers \(b = (b_j)_{j \in \mathbb{N}}\), let \((A_\nu)_{\nu \in \mathbb{F}}\) and \((B_\nu)_{\nu \in \mathbb{F}}\) be non-negative numbers satisfying the inequality

\[
A_\nu \leq \sum_{j \in \text{supp} (\nu)} \nu_j b_j A_{\nu-e_j} + B_\nu \text{ for any } \nu \in \mathbb{F} \text{ (including } \nu = 0).\]

\(\square\) Springer
Then
\[
A_{\nu} \leq \sum_{m \leq \nu} \binom{\nu}{m} |m| b^m B_{\nu-m} \text{ for all } \nu \in \mathfrak{F}.
\]

The result holds also when both inequalities are replaced by equalities.

**Lemma 9.2** Given a sequence of non-negative numbers \( \beta = (\beta_j)_{j \in \mathbb{N}} \), let \( (A_{\nu})_{\nu \in \mathfrak{F}} \) and \( (B_{\nu})_{\nu \in \mathfrak{F}} \) be non-negative numbers satisfying the inequality
\[
A_{\nu} \leq \sum_{m \leq \nu} \binom{\nu}{m} |m| \beta^m B_{\nu-m} \text{ for any } \nu \in \mathfrak{F} \text{ (including } \nu = 0).\]

Then
\[
A_{\nu} \leq \sum_{k \leq \nu} \binom{\nu}{k} \Lambda_k \beta^k B_{\nu-k} \text{ for all } \nu \in \mathfrak{F},
\]

where the sequence \( (\Lambda_n)_{n \geq 0} \) is defined recursively by
\[
\Lambda_0 := 1 \text{ and } \Lambda_n := \sum_{i=0}^{n-1} \binom{n}{i} \Lambda_i \text{ for all } n \geq 1. \quad (9.7)
\]

The result holds also when both inequalities are replaced by equalities. Moreover, we have
\[
\Lambda_n \leq \frac{n!}{(\ln 2)^n} \text{ for all } n \geq 0. \quad (9.8)
\]

**Proof of Lemma 6.1**

This result was proved in [15]. Since the same proof strategy is used repeatedly in subsequent more complicated proofs, we include this relatively simple proof as a first illustration.

Let \( f \in V^* \) and \( y \in U \). We prove this result by induction on \( |\nu| \). For \( \nu = 0 \), we take \( v = u(\cdot, y) \) in (3.5) to obtain
\[
\int_D a(x, y) |\nabla u(x, y)|^2 \, dx = \int_D f(x) u(x, y) \, dx,
\]
which leads to
\[
a_{\min} \|u(\cdot, y)\|_V^2 \leq \|f\|_{V^*} \|u(\cdot, y)\|_V \quad \implies \quad \|u(\cdot, y)\|_V \leq \frac{\|f\|_{V^*}}{a_{\min}},
\]
as required (see also (3.7)).
Given any multi-index \( \nu \) with \( |\nu| \geq 1 \), suppose that the result holds for any multi-index of order \( \leq |\nu| - 1 \). Applying the mixed derivative operators \( \partial^\nu \) to the variational formulation (3.5), recalling that \( f \) is independent of \( y \), and using the Leibniz product rule (9.1), we obtain the identity (suppressing \( x \) and \( y \) in our notation)

\[
\int_D \left( \sum_{m \leq \nu} \binom{\nu}{m} (\partial^m a) \nabla (\partial^{\nu-m} u) \cdot \nabla z \right) \, dx = 0 \quad \text{for all } z \in V.
\]

Observe that due to the linear dependence of \( a(x, y) \) on the parameters \( y \), the partial derivative \( \partial^m \) of \( a \) with respect to \( y \) satisfies

\[
\partial^m a(x, y) = \begin{cases} a(x, y) & \text{if } m = 0, \\ \psi_j(x) & \text{if } m = e_j, \\ 0 & \text{otherwise}. \end{cases} \quad (9.9)
\]

Taking \( z = \partial^\nu u(\cdot, y) \) and separating out the \( m = 0 \) term, we obtain

\[
\int_D a \left| \nabla (\partial^\nu u) \right|^2 \, dx = - \sum_{j \in \text{supp}(\nu)} v_j \int_D \psi_j \left( \nabla (\partial^{\nu-e_j} u) \cdot \nabla (\partial^\nu u) \right) \, dx,
\]

which yields

\[
a_{\min} \left\| \nabla (\partial^\nu u) \right\|_{L^2}^2 \leq \sum_{j \geq 1} v_j \left\| \psi_j \right\|_{L^\infty} \left\| \nabla (\partial^{\nu-e_j} u) \right\|_{L^2} \left\| \nabla (\partial^\nu u) \right\|_{L^2},
\]

and therefore

\[
\left\| \nabla (\partial^\nu u) \right\|_{L^2} \leq \sum_{j \geq 1} v_j b_j \left\| \nabla (\partial^{\nu-e_j} u) \right\|_{L^2},
\]

where we used the definition of \( b_j \) in (2.3). The induction hypothesis then gives

\[
\left\| \nabla (\partial^\nu u) \right\|_{L^2} \leq \sum_{j \geq 1} v_j b_j \left| \nu - e_j \right|! \, b^{\nu-e_j} \left\| f \right\|_{V^*} \frac{a_{\min}}{a_{\min}} = \left| \nu \right|! \, b^\nu \left\| f \right\|_{V^*}.\]

This completes the proof. \( \square \)

**Proof of Lemma 6.2**

This result corresponds to [69, Theorem 6]. Here we take a more direct route with the proof and the bound depends on the sequence \( \vec{b} \) which is simpler than the sequence in [69] (there the sequence depends on an additional parameter \( \kappa \in (0, 1) \) and other constants), at the expense of increasing the factorial factor from \( |\nu|! \) to \( (|\nu| + 1)! \). For
simplicity we consider here the case \( f \in L^2(D) \), but the proof can be generalized to the case \( f \in H^{-1+t}(D) \) for \( t \in [0, 1] \) as in [69].

Let \( f \in L^2(D) \) and \( y \in U \). For \( \mathbf{v} = \mathbf{0} \), we apply the identity (9.2) to the strong formulation (2.1) to obtain (formally, at this stage, since we do not yet know that \( \Delta u(\cdot, y) \in L^2(D) \))

\[
-a(\mathbf{x}, y) \Delta u(\mathbf{x}, y) = \nabla a(\mathbf{x}, y) \cdot \nabla u(\mathbf{x}, y) + f(\mathbf{x}),
\]

which leads to

\[
a_{\min} \| \Delta u(\cdot, y) \|_{L^2} \leq \| \nabla a(\cdot, y) \|_{L^\infty} \| \nabla u(\cdot, y) \|_{L^2} + \| f \|_{L^2}.
\]

Combining this with (3.7) gives

\[
\| \Delta u(\cdot, y) \|_{L^2} \leq \frac{\sup_{\mathbf{z} \in U} \| \nabla a(\cdot, \mathbf{z}) \|_{L^\infty}}{a_{\min}} \| f \|_{V^*} \frac{a_{\min}}{a_{\min}} + \| f \|_{L^2} \leq C \| f \|_{L^2},
\]

where we could take

\[
C := C_{\text{emb}} \left( \frac{\sup_{\mathbf{z} \in U} \| \nabla a(\cdot, \mathbf{z}) \|_{L^\infty}}{a_{\min}^2} + \frac{1}{a_{\min}} \right), \quad \text{with} \quad C_{\text{emb}} := \sup_{f \in L^2(D)} \frac{\| f \|_{V^*}}{\| f \|_{L^2}}.
\]

Thus, the result holds for \( \mathbf{v} = \mathbf{0} \) (see also (3.8)).

For \( \mathbf{v} \neq \mathbf{0} \), we apply \( \partial^\mathbf{v} \) to the strong formulation (2.1) and use the Leibniz product rule (9.1) to obtain (suppressing \( \mathbf{x} \) and \( \mathbf{y} \))

\[
\nabla \cdot \left( \sum_{m \leq \mathbf{v}} \binom{\mathbf{v}}{m} \partial^m a \nabla (\partial^\mathbf{v-m} u) \right) = 0.
\]

Using again (9.9) and separating out the \( m = \mathbf{0} \) term yield the following identity

\[
\nabla \cdot (a \nabla (\partial^\mathbf{v} u)) = -\nabla \cdot \left( \sum_{j \in \text{supp}(\mathbf{v})} v_j \psi_j(\mathbf{x}) \nabla (\partial^{\mathbf{v}-e_j} u) \right).
\]

Applying the identity (9.2) to both sides yields (formally)

\[
a \Delta (\partial^\mathbf{v} u) + \nabla a \cdot \nabla (\partial^\mathbf{v} u) = - \sum_{j \in \text{supp}(\mathbf{v})} v_j \left( \psi_j \Delta (\partial^{\mathbf{v}-e_j} u) + \nabla \psi_j \cdot \nabla (\partial^{\mathbf{v}-e_j} u) \right).
\]

In turn, we obtain

\[
a_{\min} \| \Delta (\partial^\mathbf{v} u) \|_{L^2} \leq \sum_{j \in \text{supp}(\mathbf{v})} v_j \left( \| \psi_j \|_{L^\infty} \| \Delta (\partial^{\mathbf{v}-e_j} u) \|_{L^2} + \| \nabla \psi_j \|_{L^\infty} \| \nabla (\partial^{\mathbf{v}-e_j} u) \|_{L^2} \right)
\]

\[
+ \| \nabla a \|_{L^\infty} \| \nabla (\partial^\mathbf{v} u) \|_{L^2},
\]
which leads to
\[ \| \Delta (\partial^{v} u) \|_{L^2} \leq \sum_{j \in \text{supp}(v)} v_{j} b_{j} \| \Delta (\partial^{v-e_j} u) \|_{L^2} + B_{v}, \]
where we used the definition of $b_{j}$ in (2.3), and

\[ B_{v} := \sum_{j \in \text{supp}(v)} v_{j} \frac{\| \nabla \psi_{j} \|_{L^\infty}}{a_{\min}} \| \nabla (\partial^{v-e_j} u) \|_{L^2} + \frac{\| \nabla a \|_{L^\infty}}{a_{\min}} \| \nabla (\partial^{v} u) \|_{L^2}. \]

Note that this formulation of $B_{v}$ cannot be used as $B_{\nu}$ in Lemma 9.1 because the base step $A_{0} \leq B_{0}$ does not hold. From Lemma 6.1 we can estimate

\[ B_{v} \leq \sum_{j \in \text{supp}(v)} v_{j} \frac{\| \nabla \psi_{j} \|_{L^\infty}}{a_{\min}} |v - e_{j}|! \| b^{v-e_j} \|_{V\ast} + \sup_{z \in U} \frac{\| \nabla a (\cdot, z) \|_{L^\infty}}{a_{\min}} |v|! \| b^{v} \|_{V\ast} \| f \|_{V\ast} \]

\[ \leq C |v|! \| \tilde{b}^{v} \|_{L^\infty(D)} =: \|B_{v}\|_{L^\infty(D)}. \]

where we used the definition of $\tilde{b}_{j} \geq b_{j}$ in (2.4). This definition of $B_{v}$ ensures that the base step $A_{0} \leq B_{0}$ does hold; see (9.10). Now we apply Lemma 9.1 to conclude that

\[ \| \Delta (\partial^{v} u) \|_{L^2} \leq \sum_{m \leq v} (\begin{pmatrix} v \\ m \end{pmatrix}) \| m \! | \tilde{b}^{v-m} \|_{L^2} \leq C (|v| + 1)! \| \tilde{b}^{v} \|_{L^\infty(D)}, \]

where we used the identity (9.4). This completes the proof. □

Proof of Lemma 6.3

This result appeared as a technical step in the proof of [69, Theorem 7], but only first derivatives were considered there, i.e., $v_{j} \leq 1$ for all $j$. Here we consider general derivatives, and we make use of Lemma 6.2 instead of [69, Theorem 6] so that the sequence $\tilde{b}$ is different, the factorial factor is larger, and we restrict to $f \in L^{2}(D)$.

Let $f \in L^{2}(D), y \in U$ and $v \in \mathbb{F}$. Galerkin orthogonality for the FE method yields

\[ A(y; u(\cdot, y) - u_{h}(\cdot, y), z_{h}) = 0 \quad \text{for all } z_{h} \in V_{h}, \quad (9.11) \]

Let $I : V \to V$ denote the identity operator and let $\mathcal{P}_{h} = \mathcal{P}_{h}(y) : V \to V_{h}$ denote the parametric FE projection onto $V_{h}$ which is defined, for arbitrary $w \in V$, by

\[ A(y; \mathcal{P}_{h}(y) w - w, z_{h}) = 0 \quad \text{for all } z_{h} \in V_{h}. \quad (9.12) \]

In particular, we have $u_{h} = \mathcal{P}_{h} u \in V_{h}$ and

\[ \mathcal{P}_{h}^{2}(y) \equiv \mathcal{P}_{h}(y) \quad \text{on } V_{h}. \quad (9.13) \]
Moreover, since $\partial^v u_h \in V_h$ for every $v \in \mathcal{F}$, it follows from (9.13) that

$$\mathcal{I} - \mathcal{P}_h(y)(\partial^v u_h(\cdot, y)) \equiv 0. \quad \text{(9.14)}$$

Thus,

$$\| \nabla \partial^v(u - u_h) \|_{L^2} = \| \nabla \mathcal{P}_h \partial^v(u - u_h) + \nabla (\mathcal{I} - \mathcal{P}_h) \partial^v(u - u_h) \|_{L^2} \leq \| \nabla \mathcal{P}_h \partial^v(u - u_h) \|_{L^2} + \| \nabla (\mathcal{I} - \mathcal{P}_h) \partial^v u \|_{L^2}. \quad \text{(9.15)}$$

We stress here that, since the parametric FE projection $\mathcal{P}_h(y)$ depends on $y$, in general we have $\partial^v(u(\cdot, y) - u_h(\cdot, y)) \neq (\mathcal{I} - \mathcal{P}_h(y))(\partial^v u(\cdot, y))$; this is why we need the estimate (9.15).

Now, applying $\partial^v$ to (9.11) and recalling (9.9), we get for all $z_h \in V_h$,

$$\int_D a \nabla \partial^v(u - u_h) \cdot \nabla z_h \, dx = - \sum_{j \in \text{supp}(v)} v_j \int_D \psi_j \nabla \partial^{v - e_j}(u - u_h) \cdot \nabla z_h \, dx. \quad \text{(9.16)}$$

Choosing $z_h = \mathcal{P}_h \partial^v(u - u_h)$ and using the definition (9.12) of $\mathcal{P}_h$, the left-hand side of (9.16) is equal to $\int_D a |\nabla \mathcal{P}_h \partial^v(u - u_h)|^2 \, dx$. Using the Cauchy–Schwarz inequality, we then obtain

$$a_{\text{min}} \| \nabla \mathcal{P}_h \partial^v(u - u_h) \|_{L^2}^2 \leq \sum_{j \in \text{supp}(v)} v_j \| \psi_j \|_{L^\infty} \| \nabla \partial^{v - e_j}(u - u_h) \|_{L^2} \| \nabla \mathcal{P}_h \partial^v(u - u_h) \|_{L^2}. \quad \text{(9.17)}$$

Canceling one common factor from both sides, we arrive at

$$\| \nabla \mathcal{P}_h \partial^v(u - u_h) \|_{L^2} \leq \sum_{j \in \text{supp}(v)} v_j b_j \| \nabla \partial^{v - e_j}(u - u_h) \|_{L^2}, \quad \text{(9.17)}$$

where we used the definition of $b_j$ in (2.3). Substituting (9.17) into (9.15), we then obtain

$$\| \nabla \partial^v(u - u_h) \|_{L^2} \leq \sum_{j \in \text{supp}(v)} v_j b_j \| \nabla \partial^{v - e_j}(u - u_h) \|_{L^2} + \| \nabla (\mathcal{I} - \mathcal{P}_h) \partial^v u \|_{L^2}. \quad \text{(9.15)}$$

Noting that $A_0 = B_0$, we now apply Lemma 9.1 to obtain

$$\| \nabla \partial^v(u - u_h) \|_{L^2} \leq \sum_{m \leq v} \binom{v}{m} |v|! b^v \| \nabla (\mathcal{I} - \mathcal{P}_h) \partial^{v - m} u \|_{L^2}. \quad \text{(9.17)}$$
Next we use the FE estimate (3.9) that for all \( y \in U \) and \( w \in H^2(D) \), we have 
\[
\| \nabla (\mathcal{A} - \mathcal{P}_h) w \|_{L^2} \lesssim h \| \Delta w \|_{L^2}.
\]
Hence, from Lemma 6.2 we obtain 
\[
\| \nabla \partial^v (u - u_h) \|_{L^2} \lesssim \sum_{m \leq v} \binom{v}{m} |\mathbf{v}|! \mathbf{b}^v h \| \Delta (\partial^{v-m} u) \|_{L^2} \\
\lesssim \sum_{m \leq v} \binom{v}{m} |\mathbf{v}|! \mathbf{b}^v h (|\mathbf{v} - m| + 1)! \mathbf{b}^{v-m} \| f \|_{L^2} \\
\lesssim h \frac{(|\mathbf{v}| + 2)!}{2} \mathbf{b}^v \| f \|_{L^2},
\]
where we used the identity (9.5). This completes the proof.

**Proof of Lemma 6.4**

This result generalizes [69, Theorem 7] from first derivatives to general derivatives. The proof is based on a duality argument since \( G \) is a bounded linear functional. It makes use of Lemma 6.3, and therefore, the sequence \( \mathbf{b} \) is different, the factorial factor is larger, and we restrict to \( f, G \in L^2(D) \) here.

Let \( f, G \in L^2(D) \) and \( y \in U \). We define \( v^G(\cdot, y) \in V \) and \( v_h^G(\cdot, y) \in V_h \) via the adjoint problems

\[
\mathcal{A}(y; w, v^G(\cdot, y)) = G(w) \quad \text{for all} \quad w \in V, \quad (9.18)
\]

\[
\mathcal{A}(y; w_h, v_h^G(\cdot, y)) = G(w_h) \quad \text{for all} \quad w_h \in V_h. \quad (9.19)
\]

Due to Galerkin orthogonality (9.11) for the original problem, by choosing the test function \( w = u(\cdot, y) - u_h(\cdot, y) \) in (9.18), we obtain

\[
G(u(\cdot, y) - u_h(\cdot, y)) = \mathcal{A}(y; u(\cdot, y) - u_h(\cdot, y), v^G(\cdot, y) - v_h^G(\cdot, y)). \quad (9.20)
\]

From the Leibniz product rule (9.1) and (9.9) we have for \( v \in \mathcal{A} \)

\[
\partial^v G(u - u_h) = \int_D \partial^v \left( a \nabla (u - u_h) \cdot \nabla (v^G - v_h^G) \right) \, dx \\
= \int_D \sum_{m \leq v} \binom{v}{m} (\partial^m a) \partial^{v-m} \left( \nabla (u - u_h) \cdot \nabla (v^G - v_h^G) \right) \, dx \\
= \int_D a \partial^v \left( \nabla (u - u_h) \cdot \nabla (v^G - v_h^G) \right) \, dx \\
+ \sum_{j \in \text{supp}(v)} v_j \int_D \psi_j \partial^{v-e_j} \left( \nabla (u - u_h) \cdot \nabla (v^G - v_h^G) \right) \, dx \\
= \int_D a \sum_{k \leq v} \binom{v}{k} \nabla \partial^k (u - u_h) \cdot \nabla \partial^{v-k} (v^G - v_h^G) \, dx \\
+ \sum_{j \in \text{supp}(v)} v_j \int_D \psi_j \sum_{k \leq v-e_j} \binom{v-e_j}{k} \nabla \partial^k (u - u_h) \cdot \nabla \partial^{v-e_j-k} (v^G - v_h^G) \, dx.
\]
The Cauchy–Schwarz inequality then yields

\[
|\partial^\nu G(u - u_h)| \leq a_{\text{max}} \sum_{k \leq \nu} \binom{\nu}{k} \|\nabla \partial^k (u - u_h)\|_{L^2} \|\nabla \partial^{\nu-k} (v^G - v^G_h)\|_{L^2} \\
+ \sum_{j \in \text{supp}(\nu)} v_j \|\psi_j\|_{L^\infty} \sum_{k \leq \nu - e_j} \binom{\nu - e_j}{k} \|\nabla \partial^k (u - u_h)\|_{L^2} \|\nabla \partial^{\nu-e_j-k} (v^G - v^G_h)\|_{L^2}.
\]  
(9.21)

We see from (the proof of) Lemma 6.3 that

\[
\|\nabla \partial^k (u - u_h)\|_{L^2} \lesssim h \frac{(|k| + 2)!}{2} \bar{b}^k \|f\|_{L^2}.
\]  
(9.22)

Since the bilinear form \(\mathcal{A}(y, \cdot, \cdot)\) is symmetric and since the representer \(g\) for the linear functional \(G\) is in \(L^2(D)\), all the results hold verbatim also for the adjoint problem (9.18) and for its FE discretisation (9.19). Hence, as in (9.22), we obtain

\[
\|\nabla \partial^{\nu-k} (v^G - v^G_h)\|_{L^2} \lesssim h \frac{(|\nu - k| + 2)!}{2} \bar{b}^{\nu-k} \|G\|_{L^2}.
\]  
(9.23)

Substituting (9.22) and (9.23) into (9.21), and using \(\|\psi_j\|_{L^\infty} = a_{\text{min}} b_j \leq a_{\text{max}} \bar{b}_j\), yields

\[
|\partial^\nu G(u - u_h)| \\
\lesssim a_{\text{max}} \|f\|_{L^2} \|G\|_{L^2} h^2 \left( \sum_{k \leq \nu} \binom{\nu}{k} \frac{(|k| + 2)!}{2} \bar{b}^k \frac{(|\nu - k| + 2)!}{2} \bar{b}^{\nu-k} \right) \\
+ \sum_{j \in \text{supp}(\nu)} v_j \bar{b}_j \sum_{k \leq \nu - e_j} \binom{\nu - e_j}{k} \frac{(|k| + 2)!}{2} \bar{b}^k \frac{(|\nu - e_j - k| + 2)!}{2} \bar{b}^{\nu-e_j-k} \\
\lesssim a_{\text{max}} \|f\|_{L^2} \|G\|_{L^2} h^2 \left( \frac{(|\nu| + 5)!}{120} + \sum_{j \in \text{supp}(\nu)} v_j \frac{(|\nu - e_j| + 5)!}{120} \right) \bar{b}^\nu \\
\lesssim a_{\text{max}} \|f\|_{L^2} \|G\|_{L^2} h^2 \frac{(|\nu| + 5)!}{120} \bar{b}^\nu,
\]

where we used the identity (9.6). This completes the proof. \(\square\)

**Proof of Lemma 6.5**

This result is [46, Theorem 14]. We include the proof here since, unlike in the uniform case where we do induction directly for the quantity \(\|\nabla (\partial^\nu u(\cdot, y))\|_{L^2}\), here we need to work with \(\|a^{1/2}(\cdot, y) \nabla (\partial^\nu u(\cdot, y))\|_{L^2}\), and this technical step is needed for the subsequent proof.
Let \( f \in V^* \) and \( y \in U_b \). We first prove by induction on \(|\nu|\) that

\[
\|a^{\nu} \nabla (\partial^\nu u(\cdot, y))\|_{L^2} \leq \Lambda_{|\nu|} \beta^\nu \frac{\|f\|_{V^*}}{\sqrt{a_{\min}(y)}},
\]

(9.24)

where the sequence \((\Lambda_n)_{n \geq 0}\) is defined recursively by (9.7) and satisfies (9.8).

We take \( v = u(\cdot, y) \) in the weak form (3.5) to obtain

\[
\int_D a |\nabla v|^2 \, dx \leq \|f\|_{V^*} \|u(\cdot, y)\|_{V} \leq \|f\|_{V^*} \left( \int_D a |\nabla u|^2 \, dx \right)^{1/2},
\]

and then cancel the common factor from both sides to obtain (9.24) for the case \( \nu = 0 \).

Given any multi-index \( \nu \) with \(|\nu| = n \geq 1\), we apply \( \partial^\nu \) to (3.5) to obtain

\[
\int_D \left( \sum_{m \leq \nu} (\nu_m) \nabla (\partial^m a) \cdot \nabla (\partial^\nu u) \cdot z \right) \, dx = 0 \quad \text{for all} \quad z \in V.
\]

Taking \( z = \partial^\nu u(\cdot, y) \), separating out the \( m = \nu \) term, dividing and multiplying by \( a \), and using the Cauchy–Schwarz inequality, we obtain

\[
\int_D a |\nabla (\partial^\nu u)|^2 \, dx = - \sum_{m \leq \nu} \sum_{m \neq \nu} (\nu_m) \int_D (\partial^m a) \nabla (\partial^m u) \cdot \nabla (\partial^\nu u) \, dx
\]

\[
\leq \sum_{m \leq \nu} (\nu_m) \left\| \frac{\partial^{\nu-m} a(\cdot, y)}{a(\cdot, y)} \right\|_{L^\infty} \left( \int_D a |\nabla (\partial^m u)|^2 \, dx \right)^{1/2} \left( \int_D a |\nabla (\partial^\nu u)|^2 \, dx \right)^{1/2}.
\]

We observe from (2.6) that

\[
\partial^{\nu-m} a = a \prod_{j \geq 1} (\sqrt{\mu_j} \xi_j)^{\nu_j-m_j} \quad \text{for all} \quad \nu \neq m,
\]

(9.25)

and therefore

\[
\left\| \frac{\partial^{\nu-m} a(\cdot, y)}{a(\cdot, y)} \right\|_{L^\infty} = \left\| \prod_{j \geq 1} (\sqrt{\mu_j} \xi_j)^{\nu_j-m_j} \right\|_{L^\infty} \leq \beta^{\nu-m}.
\]

(9.26)

Thus, we arrive at

\[
\left( \int_D a |\nabla (\partial^\nu u)|^2 \, dx \right)^{1/2} \leq \sum_{m \leq \nu} (\nu_m) \beta^{\nu-m} \left( \int_D a |\nabla (\partial^m u)|^2 \, dx \right)^{1/2}.
\]
We now use the inductive hypothesis that (9.24) holds when $|\nu| \leq n - 1$ in each of the terms on the right-hand side to obtain

$$
\left( \int_D a |\nabla (\partial^\nu u)|^2 \, dx \right)^{1/2} \leq \sum_{i=0}^{n-1} \sum_{m \leq \nu, |m|=i} (\nu_m) \beta^{\nu-m} \Lambda_i \beta^m \frac{\|f\|_V^*}{\sqrt{a_{\min}(y)}}
$$

$$
= \sum_{i=0}^{n-1} \binom{n}{i} \Lambda_i \beta^\nu \frac{\|f\|_V^*}{\sqrt{a_{\min}(y)}} = \Lambda_n \beta^\nu \frac{\|f\|_V^*}{\sqrt{a_{\min}(y)}},
$$

where we used the identity (9.3). This completes the induction proof of (9.24).

The desired bound in the lemma is obtained by applying (9.8) on the right-hand side of (9.24), and by noting that the left-hand side of (9.24) can be bounded from below by $\sqrt{a_{\min}(y)} \|\partial^\nu u(\cdot, y)\|_V$. The case $\nu = 0$ corresponds to (3.12). This completes the proof. \qed

**Proof of Lemma 6.6**

This result was proved in [66] based on an argument similar to the proof of Lemma 6.2 in the uniform case. The tricky point of the proof is in recognizing that for the recursion to work in the lognormal case we need to multiply the expression by $a^{-1/2}(\cdot, y)$, which is not intuitive.

Let $f \in L^2(D)$ and $y \in U_{\beta^\nu}$. For any multi-index $\nu \neq 0$, we apply $\partial^\nu$ to (2.1) to obtain (formally, at this stage)

$$
\nabla \cdot \partial^\nu (a \nabla u) = \nabla \cdot \left( \sum_{m \leq \nu} (\nu_m) \partial^{\nu-m} a \nabla (\partial^m u) \right) = 0.
$$

Separating out the $m = \nu$ term yields the following identity

$$
g_\nu := \nabla \cdot (a \nabla (\partial^\nu u)) = -\nabla \cdot \left( \sum_{m \leq \nu, m \neq \nu} (\nu_m) \partial^{\nu-m} a \nabla (\partial^m u) \right)
$$

$$
= -\sum_{m \leq \nu, m \neq \nu} (\nu_m) \nabla \cdot \left( \frac{\partial^{\nu-m} a}{a} \nabla (\partial^m u) \right)
$$

$$
= -\sum_{m \leq \nu, m \neq \nu} (\nu_m) \left( \frac{\partial^{\nu-m} a}{a} \cdot \delta_m + \nabla \left( \frac{\partial^{\nu-m} a}{a} \right) \cdot (a \nabla (\partial^m u)) \right),
$$
where we used the identity (9.2). Due to Assumption (L2) we may multiply $g_\nu$ by $a^{-1/2}$ and obtain, for any $|\nu| > 0$, the recursive bound

$$
\|a^{-1/2}g_\nu\|_{L^2} \leq \sum_{m \leq \nu \atop m \neq \nu} (\nu m) \left( \|\frac{\partial^{\nu-m}a}{a}\|_{L^\infty} \|a^{-1/2}g_m\|_{L^2} \right)
+ \|\nabla\left(\frac{\partial^{\nu-m}a}{a}\right)\|_{L^\infty} \|a^{1/2}\nabla(a^m u)\|_{L^2}.
$$

(9.27)

By assumption, $-g_0 = f \in L^2(D)$, so that we obtain (by induction with respect to $|\nu|$) from (9.27) that $a^{-1/2}(\cdot, y) g_\nu(\cdot, y) \in L^2(D)$, and hence from Assumption (L2) that $g_\nu(\cdot, y) \in L^2(D)$ for every $\nu \in \mathfrak{F}$. The above formal identities therefore hold in $L^2(D)$.

To complete the proof, it remains to bound the above $L^2$ norm. Applying the product rule to (9.25) we obtain

$$
\nabla\left(\frac{\partial^{\nu-m}a}{a}\right) = \sum_{k \geq 1} (\nu k - m_k)(\sqrt{\mu_k} \xi_k)^{\nu_k-m_k-1}(\sqrt{\mu_k} \nabla \xi_k) \prod_{j \geq 1 \atop j \neq k} (\sqrt{\mu_j} \xi_j)^{\nu_j-m_j}.
$$

Due to the definition of $\overline{\beta}_j$ in (2.9), this implies, in a similar manner to (9.26), that

$$
\|\nabla\left(\frac{\partial^{\nu-m}a}{a}\right)\|_{L^\infty} \leq |\nu - m| \overline{\beta}^{\nu-m}.
$$

(9.28)

Substituting (9.26) and (9.28) into (9.27), we conclude that

$$
\|a^{-1/2}g_\nu\|_{L^2} \leq \sum_{m \leq \nu \atop m \neq \nu} (\nu m) \overline{\beta}^{\nu-m} \|a^{-1/2}g_m\|_{L^2} + B_\nu,
$$

where

$$
B_\nu := \sum_{m \leq \nu \atop m \neq \nu} (\nu m) |\nu - m| \overline{\beta}^{\nu-m} \|a^{1/2}\nabla(a^m u)\|_{L^2}
$$

$$
\leq \sum_{m \leq \nu \atop m \neq \nu} (\nu m) |\nu - m| \overline{\beta}^{\nu-m} \lambda_{|\nu|} \beta^m \frac{\|f\|_{V^*}}{\sqrt{a_{\min}(y)}} \leq \Lambda_{|\nu|} \overline{\beta} \frac{\|f\|_{V^*}}{\sqrt{a_{\min}(y)}},
$$

where we used (9.24) and again the identity (9.3) to write, with $n = |\nu|$,

$$
\sum_{m \leq \nu \atop m \neq \nu} (\nu m) |\nu - m| \lambda_{|\nu|} = \sum_{i=0}^{n-1} \sum_{m \leq \nu \atop |m| = i} (\nu m) (n-i) \Lambda_i = \sum_{i=0}^{n-1} \binom{n}{i} (n-i) \Lambda_i =: \overline{\Lambda}_n.
$$
Since \( A_0 = \|a^{-1/2}f\|_{L^2} \leq \|f\|_{L^2}/\sqrt{a_{\min}(y)} \), we now define
\[
\mathbb{B}_v := C_{emb} \Lambda_{|v|} \beta^v \|f\|_{L^2}^{\sqrt{a_{\min}(y)}},
\]
so that \( A_0 \leq \mathbb{B}_0 \) and \( B_v \leq \mathbb{B}_v \) for all \( v \). We may now apply Lemma 9.2 to obtain
\[
\|a^{-1/2}g_v\|_{L^2} \leq \sum_{k \leq v} \binom{v}{k} \Lambda_{|k|} \beta^k C_{emb} \Lambda_{|v-k|} \beta^{v-k} \|f\|_{L^2}^{\sqrt{a_{\min}(y)}}. \tag{9.29}
\]
Note the extra factor \( n-i \) in the definition of \( \Lambda_n \) compared to \( \Lambda_n \) in (9.7) so that \( \Lambda_n \leq \overline{\Lambda}_n \). Using the bound in (9.8) with \( \alpha \leq 1 \), we have
\[
\Lambda_n \leq \sum_{i=0}^{n-1} \binom{n}{i} (n-i) \frac{i!}{\alpha^i} = \frac{n!}{\alpha^n} \sum_{i=0}^{n-1} \frac{\alpha^{n-i-1}}{(n-i-1)!} = \frac{n!}{\alpha^n} \sum_{k=0}^{n-1} \frac{\alpha^k}{k!} \leq \frac{n!}{\alpha^n} \alpha e^\alpha \leq \frac{n!}{\alpha^n},
\]
where the final step is valid provided that \( \alpha e^\alpha \leq 1 \). Thus, it suffices to choose \( \alpha \leq 0.567 \ldots \). For convenience we take \( \alpha = 0.5 \) to bound (9.29). This together with the identity (9.4) gives
\[
\|a^{-1/2}g_v\|_{L^2} \leq C_{emb} (|v| + 1)! 2^{|v|} \beta^v \|f\|_{L^2}^{\sqrt{a_{\min}(y)}}. \tag{9.30}
\]
Since \( a^{-1/2}g_v = a^{-1/2}\nabla \cdot (a\nabla (\partial^v u)) = a^{1/2} \Delta(\partial^v u) + a^{-1/2} (\nabla a \cdot \nabla (\partial^v u)) \) by applying (9.2), we have
\[
\|a^{1/2} \Delta(\partial^v u)\|_{L^2} \leq \|a^{-1/2}g_v\|_{L^2} + \|a^{-1/2} (\nabla a \cdot \nabla (\partial^v u))\|_{L^2},
\]
which yields
\[
\sqrt{a_{\min}(y)} \|\Delta(\partial^v u)\|_{L^2} \leq \|a^{-1/2}g_v\|_{L^2} + \frac{\|\nabla a(\cdot, y)\|_{L^\infty}}{a_{\min}(y)} \|a^{1/2} \nabla (\partial^v u)\|_{L^2},
\]
and in turn
\[
\|\Delta(\partial^v u)\|_{L^2} \leq \frac{\|a^{-1/2}g_v\|_{L^2}}{\sqrt{a_{\min}(y)}} + \frac{\|\nabla a(\cdot, y)\|_{L^\infty}}{a_{\min}(y)} \frac{\|a^{1/2} \nabla (\partial^v u)\|_{L^2}}{\sqrt{a_{\min}(y)}}. \tag{9.31}
\]
Substituting (9.30) and (9.24) into (9.31), and using \( \Lambda_{|v|} \leq 2^{|v|} |v|! \) and \( \beta^v \leq \overline{\beta}^v \), we conclude that
\[
\|\Delta(\partial^v u)\|_{L^2} \leq C_{emb} \left( \frac{1}{a_{\min}(y)} + \frac{\|\nabla a(\cdot, y)\|_{L^\infty}}{a_{\min}^2(y)} \right) (|v| + 1)! 2^{|v|} \overline{\beta}^v \|f\|_{L^2}.
\]
This completes the proof.
Proof of Lemma 6.7

This result was proved in [66]. We include the proof here to provide a complete unified view of the proof techniques discussed in this survey.

Let \( f \in L^2(D) \) and \( \nu \in U_{\beta} \). Following (9.12)–(9.15) in the uniform case, we can write in the lognormal case

\[
\| a^{1/2} \nabla \partial^\nu (u - u_h) \|_{L^2} \leq \| a^{1/2} \nabla \mathcal{P}_h \partial^\nu (u - u_h) \|_{L^2} + \| a^{1/2} \nabla (\mathcal{I} - \mathcal{P}_h) \partial^\nu u \|_{L^2}.
\]

(9.32)

Now, applying \( \partial^\nu \) to (9.11) and separating out the \( \nu = m \) term, we get for all \( z_h \in V_h \) in the lognormal case that

\[
\int_D a \nabla \partial^\nu (u - u_h) \cdot \nabla z_h \, dx = - \sum_{m \leq \nu \atop m \neq \nu} \nu_m \int_D (\partial^{\nu-m} a) \nabla^m (u - u_h) \cdot \nabla z_h \, dx.
\]

(9.33)

Choosing \( z_h = \mathcal{P}_h \partial^\nu (u - u_h) \) and using the definition (9.12) of \( \mathcal{P}_h \), the left-hand side of (9.33) is equal to \( \int_D a |\nabla \mathcal{P}_h \partial^\nu (u - u_h)|^2 \, dx \). Dividing and multiplying the right-hand side of (9.33) by \( a \), and using the Cauchy–Schwarz inequality, we then obtain

\[
\int_D a |\nabla \mathcal{P}_h \partial^\nu (u - u_h)|^2 \, dx \\
\leq \sum_{m \leq \nu \atop m \neq \nu} \nu_m \left( \int_D a |\nabla \partial^m (u - u_h)|^2 \, dx \right)^{1/2} \left( \int_D a |\nabla \mathcal{P}_h \partial^\nu (u - u_h)|^2 \, dx \right)^{1/2}.
\]

Canceling one common factor from both sides and using (9.26), we arrive at

\[
\| a^{1/2} \nabla \mathcal{P}_h \partial^\nu (u - u_h) \|_{L^2} \leq \sum_{m \leq \nu \atop m \neq \nu} \nu_m \| a^{1/2} \nabla \partial^m (u - u_h) \|_{L^2}.
\]

(9.34)

Substituting (9.34) into (9.32), we then obtain

\[
\| a^{1/2} \nabla \partial^\nu (u - u_h) \|_{L^2} \\
\leq \sum_{m \leq \nu \atop m \neq \nu} \nu_m \| a^{1/2} \nabla \partial^m (u - u_h) \|_{L^2} + \| a^{1/2} \nabla (\mathcal{I} - \mathcal{P}_h) \partial^\nu u \|_{L^2}.
\]
Note that we have \( A_0 = B_0 \). Now applying Lemma 9.2 with \( \alpha = 0.5 \), together with (3.14), Lemma 6.6 and (9.5), we conclude that

\[
\| a^{1/2} \nabla \partial^\nu (u - u_h) \|_{L^2} \leq \sum_{m \leq \nu} (\nu_m^k)^{m} \beta^m \| a^{1/2} \nabla (\mathcal{F} - \mathcal{P}_h) \partial^\nu - m \|_{L^2} \\
\lesssim h a^{1/2} (y) \sum_{m \leq \nu} (\nu_m^k)^{m} \beta^m \| \Delta (\partial^{\nu - m} u) \|_{L^2} \\
\lesssim h T(y) a^{1/2} (y) \sum_{m \leq \nu} (\nu_m^k)^{m} \beta^m (|\nu - m| + 1)! \\
\times 2^{\nu - m} \beta^{-\nu - m} \| f \|_{L^2} \\
\lesssim h T(y) a^{1/2} (y) (|\nu| + 2)! 2^{\nu - \nu} \beta^{-\nu} \| f \|_{L^2},
\]

where \( T(y) \) is defined in (3.13). This completes the proof. \( \square \)

**Proof of Lemma 6.8**

This result was proved in [66]. Again we include the proof here to provide a complete unified view of the proof techniques discussed in this survey.

Let \( f, G \in L^2(D) \) and \( y \in U \). Following (9.18)–(9.20) in the uniform case, and using the Leibniz product rule (9.1), we have for the lognormal case that

\[
\partial^\nu G(u - u_h) = \int_D \partial^\nu \left( a \nabla (u - u_h) \cdot \nabla \left( v^G - v^G_h \right) \right) \, dx \\
= \int_D \sum_{m \leq \nu} (\nu_m^k)^{m} \beta^m \partial^{\nu - m} a \partial^m \left( \nabla (u - u_h) \cdot \nabla \left( v^G - v^G_h \right) \right) \, dx \\
= \int_D \sum_{m \leq \nu} (\nu_m^k)^{m} \partial^{\nu - m} a \sum_{k \leq m} (m_k^{m}) \nabla \partial^k (u - u_h) \cdot \nabla \partial^{m - k} \left( v^G - v^G_h \right) \, dx \\
= \int_D \sum_{m \leq \nu} (\nu_m^k)^{m} \partial^{\nu - m} a \sum_{k \leq m} (m_k^{m}) \left( a^{1/2} \nabla \partial^k (u - u_h) \right) \cdot \left( a^{1/2} \nabla \partial^{m - k} \left( v^G - v^G_h \right) \right) \, dx.
\]

Using the Cauchy–Schwarz inequality and (9.26), we obtain

\[
|\partial^\nu G(u - u_h)| \\
\leq \sum_{m \leq \nu} (\nu_m^k)^{m} \beta^{-\nu - m} \sum_{k \leq m} (m_k^{m}) \| a^{1/2} \nabla \partial^k (u - u_h) \|_{L^2} \| a^{1/2} \nabla \partial^{m - k} \left( v^G - v^G_h \right) \|_{L^2}. 
\]

(9.35)
We have from Lemma 6.7 that
\[
\| a^{1/2} \nabla^k (u - u_h) \|_{L^2} \lesssim h T(y) a^{1/2}_{\text{max}}(y) \frac{(|k| + 2)!}{2} 2^{k|\beta|} \| f \|_{L^2}.
\tag{9.36}
\]

Since the bilinear form $\mathcal{A}(\cdot; \cdot, \cdot)$ is symmetric and since the representer $g$ for the linear functional $G$ is in $L^2$, all the results hold verbatim also for the adjoint problem (9.18) and for its FE discretisation (9.19). Hence, as in (9.36), we obtain
\[
\| a^{1/2} \nabla^{m-k} (v^G - v^G_h) \|_{L^2} \lesssim h T(y) a^{1/2}_{\text{max}}(y) \frac{(|m-k| + 2)!}{2} 2^{m-k|\beta|} \| G \|_{L^2}.
\tag{9.37}
\]

Substituting (9.36) and (9.37) into (9.35), and using the identity (9.6), we obtain
\[
| \partial^\nu G(u - u_h) | \lesssim h^2 T^2(y) a_{\text{max}}(y) \sum_{m \leq \nu} \binom{\nu}{m} \frac{(|m| + 5)!}{120} 2^{m|\beta|^\nu} \| f \|_{L^2} \| G \|_{L^2}.
\]

Using again (9.3), with $n = |\nu|$ we have
\[
\sum_{m \leq \nu} \binom{\nu}{m} 2^{|m| \frac{(|m| + 5)!}{120}} = \sum_{i=0}^{n} \binom{n}{i} 2^{(i+5)!} \frac{n!}{120} \sum_{i=0}^{n} (i+1)(i+2)(i+3)(i+4)(i+5)2^i \leq \frac{(n+5)!}{120} \frac{2^n n!}{120}.
\]

This yields the required bound in the lemma. This completes the proof. \qed

References

1. I. Babuška, F. Nobile, and R. Tempone, A stochastic collocation method for elliptic partial differential equations with random input data, *SIAM J. Numer. Anal.* 45 (2007), 1005–1034.
2. I. Babuška, R. Tempone, and G.E. Zouraris, Galerkin finite element approximations of stochastic elliptic partial differential equations, *SIAM J. Numer. Anal.* 42 (2004), 800–825.
3. J. Baldeaux and J. Dick, QMC rules of arbitrary high order: reproducing kernel Hilbert space approach, * Constr. Approx.* 30 (2009), 495–527.
4. J. Baldeaux, J. Dick, J. Greslehner, and F. Pillichshammer, Construction algorithms for higher order polynomial lattice rules, *J. Complexity* 27 (2011), 281–299.
5. J. Baldeaux, J. Dick, G. Leobacher, D. Nuyens, and F. Pillichshammer, Efficient calculation of the worst-case error and (fast) component-by-component construction of higher order polynomial lattice rules, *Numer. Algorithms* 59 (2012), 403–431.
6. A. Barth, Ch. Schwab, and N. Zollinger, Multi-level Monte Carlo finite element method for elliptic PDEs with stochastic coefficients, *Numer. Math.* 119 (2011), 123–161.
7. V. I. Bogachev, *Gaussian Measures*, AMS Monographs Vol. 62, American Mathematical Society, R.I., USA, 1998.
8. H. Bungartz and M. Griebel, Sparse grids, *Acta Numer.* 13 (2004), 147–269.
9. R. E. Caflisch, W. Morokoff, and A. Owen, Valuation of mortgage backed securities using Brownian bridges to reduce effective dimension, *J. Comput. Finance* 1 (1997), 27–46.
10. J. Charron, Strong and weak error estimates for elliptic partial differential equations with random coefficients, *SIAM J. Numer. Anal.* 50 (2012), 216–246.
11. J. Charrier, R. Scheichl, and A. L. Teckentrup, Finite element error analysis of elliptic PDEs with random coefficients and its application to multilevel Monte Carlo methods, *SIAM J. Numer. Anal.* **51** (2013), 322–352.
12. K. A. Cliffe, M. B. Giles, R. Scheichl, and A. L. Teckentrup, Multilevel Monte Carlo methods and applications to elliptic PDEs with random coefficients, *Computing and Visualization in Science* **14** (2011), 3–15.
13. A. Cohen, A. Chkifa, and Ch. Schwab, Breaking the curse of dimensionality in sparse polynomial approximation of parametric PDEs, *Journ. Math. Pures et Appliquees* **103** (2015), 400–428.
14. A. Cohen and R. DeVore, Approximation of high-dimensional parametric PDEs, *Acta Numer.* **24** (2015), 1–159.
15. A. Cohen, R. DeVore, and Ch. Schwab, Convergence rates of best $N$-term Galerkin approximations for a class of elliptic sPDEs, *Found. Comp. Math.* **10** (2010), 615–646.
16. N. Collier, A.-L. Haji-Ali, F. Nobile, E. von Schwerin, and R. Tempone, A continuation multilevel Monte Carlo algorithm, *BIT*, **55** (2015), 399–432.
17. R. Cools, F. Y. Kuo, and D. Nuyens, Constructing embedded lattice rules for multivariate integration, *SIAM J. Sci. Comput.* **28** (2006), 2162–2188.
18. G. Dagan, Solute transport in heterogeneous porous formations, *J. Fluid Mech.* **145** (1984), 151–177.
19. J. Dick, On the convergence rate of the component-by-component construction of good lattice rules, *J. Complexity* **20** (2004), 493–522.
20. J. Dick, Explicit constructions of Quasi-Monte Carlo rules for the numerical integration of high-dimensional periodic functions, *SIAM J. Numer. Anal.* **45** (2007), 2141–2176.
21. J. Dick, Walsh spaces containing smooth functions and Quasi-Monte Carlo rules of arbitrary high order, *SIAM J. Numer. Anal.* **46** (2008), 1519–1553.
22. J. Dick, The decay of the Walsh coefficients of smooth functions, *Bull. Aust. Math. Soc.* **80** (2009), 430–453.
23. J. Dick, F. Y. Kuo, Q. T. Le Gia, D. Nuyens, and Ch. Schwab, Higher order QMC Galerkin discretization for parametric operator equations, *SIAM J. Numer. Anal.* **52** (2014), 2676–2702.
24. J. Dick, F. Y. Kuo, Q. T. Le Gia, and Ch. Schwab, Fast QMC matrix-vector multiplication, *SIAM J. Sci. Comput.* **37** (2015), A1436–A1450.
25. J. Dick, F. Y. Kuo, Q. T. Le Gia, and Ch. Schwab, Multi-level higher order QMC Galerkin discretization for affine parametric operator equations, *SIAM J. Numer. Anal.* **54** (2016), 2541–2568.
26. J. Dick, F. Y. Kuo, F. Pillichshammer, and I. H. Sloan, Construction algorithms for polynomial lattice rules for multivariate integration, *Math. Comp.* **74** (2005), 1895–1921.
27. J. Dick, F. Y. Kuo, and I. H. Sloan, High-dimensional integration: the Quasi-Monte Carlo way, *Acta Numer.* **22** (2013), 133–288.
28. J. Dick, Q. T. Le Gia, and Ch. Schwab, Higher order Quasi-Monte Carlo integration for holomorphic, parametric operator equations, in review.
29. J. Dick, D. Nuyens, and F. Pillichshammer, Lattice rules for nonperiodic smooth integrands, *Numer. Math.* **126** (2014), 259–291.
30. J. Dick and F. Pillichshammer, Multivariate integration in weighted Hilbert spaces based on Walsh functions and weighted Sobolev spaces, *J. Complexity* **21** (2005), 149–195.
31. J. Dick and F. Pillichshammer, Digital Nets and Sequences, Cambridge University Press, 2010.
32. J. Dick, F. Pillichshammer, and B. J. Waterhouse, The construction of good extensible rank-1 lattices, *Math. Comp.* **77** (2008), 2345–2374.
33. J. Dick, I. H. Sloan, X. Wang, and H. Woźniakowski, Liberating the weights, *J. Complexity* **20** (2004), 593–623.
34. C. R. Dietrich and G. H. Newsam, Fast and exact simulation of stationary Gaussian processes through circulant embedding of the covariance matrix, *SIAM J. Sci. Comput.* **18** (1997), 1088–1107.
35. R. Freeze, A stochastic conceptual analysis of one-dimensional groundwater flow in nonuniform homogeneous media, *Water Resour. Res.* **11** (1975), 725–741.
36. M. Ganesh and S. C. Hawkins, A high performance computing and sensitivity analysis algorithm for stochastic many-particle wave scattering, *SIAM J. Sci. Comput.* **37** (2015), A1475–A1503.
37. R. N. Gantner and Ch. Schwab, Computational Higher-Order QMC integration, Research Report 2014-24, Seminar for Applied Mathematics, ETH Zürich.
38. R. G. Ghanem and P. D. Spanos, *Stochastic Finite Elements*, Dover, 1991.
39. M. B. Giles, Improved multilevel Monte Carlo convergence using the Milstein scheme, Monte Carlo and Quasi-Monte Carlo methods 2006, Springer, 2007, pp 343–358.
40. M. B. Giles, Multilevel Monte Carlo path simulation, *Oper. Res.* **256** (2008), 981–986.
41. M. B. Giles, Multilevel Monte Carlo methods, *Acta Numer.* **24** (2015), 259–328.
42. M. B. Giles and B. J. Waterhouse, Multilevel quasi-Monte Carlo path simulation. *Radon Series Comp. Appl. Math.* **8** (2009), 1–18.
43. T. Goda, Good interlaced polynomial lattice rules for numerical integration in weighted Walsh spaces, *J. Comput. Appl. Math.* **285** (2015), 279–294.
44. T. Goda and J. Dick, Construction of interlaced scrambled polynomial lattice rules of arbitrary high order, *Found. Comput. Math.* **15** (2015), 1245–1278.
45. T. Goda, K. Suzuki, and T. Yoshi, Digital nets with infinite digit expansions and construction of folded digital nets for quasi-Monte Carlo integration, *J. Complexity* **33** (2016), 30–54.
46. I. G. Graham, F. Y. Kuo, J. Nichols, R. Scheichl, Ch. Schwab, and I. H. Sloan, QMC FE methods for PDEs with log-normal random coefficients, *Numer. Math.* **131** (2015), 329–368.
47. I. G. Graham, F. Y. Kuo, D. Nuyens, R. Scheichl, and I. H. Sloan, Quasi-Monte Carlo methods for elliptic PDEs with random coefficients and applications, *J. Comput. Phys.* **230** (2011), 3668–3694.
48. I. G. Graham, F. Y. Kuo, D. Nuyens, R. Scheichl, and I. H. Sloan, Analysis of QMC methods with circulant embedding for elliptic PDEs with lognormal coefficients, in preparation.
49. M. Gunzburger, C. Webster, and G. Zhang, Stochastic finite element methods for partial differential equations with random input data, *Acta Numer.* **23** (2014), 521–650.
50. A. L. Haji-Ali, F. Nobile, and R. Tempone, Multi-index Monte Carlo: when sparsity meets sampling, *Numer. Math.* **132** (2016), 767–806.
51. H. Harbrecht, M. Peters, and M. Siebenmorgen, On multilevel quadrature for elliptic stochastic partial differential equations, in Sparse Grids and Applications, Lecture Notes in Computational Science and Engineering, Volume 88, 2013, pp.161–179.
52. H. Harbrecht, M. Peters, and M. Siebenmorgen, Multilevel accelerated quadrature for PDEs with log-normal distributed random coefficient. Preprint 2013-18, Math. Institut, Universität Basel, 2013.
53. H. Harbrecht, M. Peters, and M. Siebenmorgen, On the quasi-Monte Carlo method with Halton points for elliptic PDEs with log-normal diffusion, *Math. Comp.*, appeared online.
54. S. Heinrich, Monte Carlo complexity of global solution of integral equations, *J. Complexity* **14** (1998), 151–175.
55. S. Heinrich, Multilevel Monte Carlo methods, Lecture notes in Compu. Sci. Vol. 2179, Springer, 2001, pp. 3624–3651.
56. F. J. Hickernell, Obtaining $O(N^{-2+\epsilon})$ convergence for lattice quadrature rules, in Monte Carlo and Quasi-Monte Carlo Methods 2000 (K. T. Fang, F. J. Hickernell, and H. Niederreiter, eds.), Springer, Berlin, 2002, pp. 274–289.
57. F. J. Hickernell, P. Kritzer, F. Y. Kuo, and D. Nuyens, Weighted compound integration rules with higher order convergence for all $N$, *Numer. Algorithms* **59** (2012), 161–183.
58. V. H. Hoang and Ch. Schwab, Regularity and generalized polynomial chaos approximation of parametric and random second-order hyperbolic partial differential equations, *Analysis and Applications (Singapore)* **10** (2012), 295–326.
59. V. H. Hoang and Ch. Schwab, Analytic regularity and polynomial approximation of stochastic, parametric elliptic multiscale PDEs, *Analysis and Applications (Singapore)* **11** (2013), 1350001, 50.
60. V. Hoang, Ch. Schwab, and A. Stuart, Sparse MCMC gpc finite element methods for Bayesian inverse problems, *Inverse Problems* **29** (2013), 085010.
61. R. J. Hoeksema and P. K. Kitanidis, Analysis of the spatial structure of properties of selected aquifers, *Water Resour. Res.* **21** (1985), 536–572.
62. S. Joe, Construction of good rank-1 lattice rules based on the weighted star discrepancy, in Monte Carlo and Quasi-Monte Carlo Methods 2004 (H. Niederreiter and D. Talay, eds.), Springer Verlag, pp. 181–196, 2006.
63. S. Joe and F. Y. Kuo, Constructing Sobol’ sequences with better two-dimensional projections, *SIAM J. Sci. Comput.* **30** (2008), 2635–2654.
64. A. Kunoth and Ch. Schwab, Analytic regularity and GPC approximation for stochastic control problems constrained by linear parametric elliptic and parabolic PDEs, *SIAM J. Control and Optimization* **51** (2013), 2442–2471.
65. F. Y. Kuo, Component-by-component constructions achieve the optimal rate of convergence for multivariate integration in weighted Korobov and Sobolev spaces, *J. Complexity* **19** (2003), 301–320.
66. F. Y. Kuo, R. Scheichl, Ch. Schwab, I. H. Sloan, and E. Ullmann, Multilevel quasi-Monte Carlo methods for lognormal diffusion problems. *Math. Comp.*, to appear.
67. F. Y. Kuo, Ch. Schwab, and I. H. Sloan, Quasi-Monte Carlo methods for high dimensional integration: the standard weighted-space setting and beyond, *ANZIAM J.* 53 (2011), 1–37.

68. F. Y. Kuo, Ch. Schwab, and I. H. Sloan, Quasi-Monte Carlo finite element methods for a class of elliptic partial differential equations with random coefficient, *SIAM J. Numer. Anal.* 50 (2012), 3351–3374.

69. F. Y. Kuo, Ch. Schwab, and I. H. Sloan, Multi-level quasi-Monte Carlo finite element methods for a class of elliptic partial differential equations with random coefficient, *Found. Comput. Math.* 15 (2015), 411–449.

70. F. Y. Kuo, I. H. Sloan, G. W. Wasilkowski, and B. J. Waterhouse, Randomly shifted lattice rules with the optimal rate of convergence for unbounded integrands, *J. Complexity* 26 (2010), 135–160.

71. Q. T. Le Gia, A QMC-spectral method for elliptic PDEs with random coefficients on the unit sphere, in Monte Carlo and Quasi-Monte Carlo Methods 2012 (J. Dick, F. Y. Kuo, G. W. Peters and I. H. Sloan, eds.), Springer Verlag, Heidelberg, 2013, pp. 491–508.

72. M. Loève, Probability Theory, Volume II, Springer-Verlag, New York, 4th edition, 1978.

73. Z. Lu and D. Zhang, A comparative study on quantifying uncertainty of flow in randomly heterogeneous media using Monte Carlo simulations, the conventional and KL-based moment-equation approaches, *SIAM J. Sci. Comput.* 26 (2004), 558–577.

74. R. L. Naff, D. F. Haley, and E. A. Sudicky, High-resolution Monte Carlo simulation of flow and conservative transport in heterogeneous porous media 1. Methodology and flow results, *Water Resour. Res.*, 34 (1998), 663–677.

75. R. L. Naff, D. F. Haley, and E. A. Sudicky, High-resolution Monte Carlo simulation of flow and conservative transport in heterogeneous porous media 2. Transport Results, *Water Resour. Res.*, 34 (1998), 679–697.

76. J. A. Nichols and F. Y. Kuo, Fast CBC construction of randomly shifted lattice rules achieving $O(N^{-\alpha+\delta})$ in weighted spaces with POD weights, *J. Complexity* 30 (2014), 444–468.

77. H. Niederreiter, Random Number Generation and Quasi-Monte Carlo Methods, SIAM, Philadelphia, 1992.

78. V. Nistor and C. Schwab, High order Galerkin approximations for parametric second order elliptic partial differential equations, *Math. Mod. Meth. Appl. Sci.* 23 (2013), 1729–1760.

79. F. Nobile, R. Tempone, and C. G. Webster, A sparse grid stochastic collocation method for partial differential equations with random input data, *SIAM J. Numer. Anal.* 46 (2008), 2309–2345.

80. F. Nobile, R. Tempone, and C. G. Webster, An anisotropic sparse grid stochastic collocation method for partial differential equations with random input data, *SIAM J. Numer. Anal.* 46 (2008), 2411–2442.

81. E. Novak and H. Woźniakowski, Tractability of Multivariate Problems, Volume I: Linear Information, European Mathematical Society, Zürich, 2008.

82. E. Novak and H. Woźniakowski, Tractability of Multivariate Problems, II: Standard Information for Functionals, European Mathematical Society, Zürich, 2010.

83. E. Novak and H. Woźniakowski, Tractability of Multivariate Problems, Volume III: Standard Information for Operators, European Mathematical Society, Zürich, 2012.

84. D. Nuyens, The construction of good lattice rules and polynomial lattice rules, In: Uniform Distribution and Quasi-Monte Carlo Methods (P. Kritzer, H. Niederreiter, F. Pillichshammer, A. Winterhof, eds.), Radon Series on Computational and Applied Mathematics Vol. 15, De Gruyter, 2014, pp. 223–256.

85. D. Nuyens and R. Cools, Fast algorithms for component-by-component construction of rank-1 lattice rules in shift-invariant reproducing kernel Hilbert spaces, *Math. Comp.* 75 (2006), 903–920.

86. D. Nuyens and R. Cools, Fast component-by-component construction of rank-1 lattice rules with a non-prime number of points, *J. Complexity* 22 (2006), 4–28.

87. D. Nuyens and R. Cools, Fast component-by-component construction, a reprise for different kernels, in Monte Carlo and quasi-Monte Carlo methods 2004 (H. Niederreiter and D. Talay, eds.), Radon Series on Computational and Applied Mathematics Vol. 15, De Gruyter, 2014, pp. 223–256.

88. S. H. Paskov and J. F. Traub, Faster valuation of financial derivatives, *J. Portfolio Management* 22 (1995), 113–120.

89. P. Robbe, D. Nuyens, and S. Vandewalle, A practical multilevel quasi-Monte Carlo method for elliptic PDEs with random coefficients. In Master Thesis “Een parallelle multilevel Monte-Carlo-methode voor de simulatie van stochastische partiële differentiaalvergelijkingen” by P. Robbe, June 2015.

90. Y. Rubin, Applied Stochastic Hydrogeology, Oxford University Press, New York, 2003.

91. C. Schillings and Ch. Schwab, Sparse, adaptive smolyak algorithms for Bayesian inverse problems, *Inverse Problems* 29 (2013), 065011.
92. C. Schillings and Ch. Schwab, Sparsity in Bayesian inversion of parametric operator equations, *Inverse Problems* 30 (2014), 065007.

93. Ch. Schwab, QMC Galerkin discretizations of parametric operator equations, in Monte Carlo and Quasi-Monte Carlo Methods 2012 (J. Dick, F. Y. Kuo, G. W. Peters and I. H. Sloan, eds.), Springer Verlag, Heidelberg, 2013, pp. 613–629.

94. Ch. Schwab and C. J. Gittelson, Sparse tensor discretizations of high-dimensional parametric and stochastic PDEs, *Acta Numer.* 20 (2011), 291–467.

95. Ch. Schwab and R. A. Todor, Karhunen-Loève approximation of random fields by generalized fast multipole methods, *J. Comput. Phy.* 217 (2006), 100–122.

96. Ch. Schwab and R. A. Todor, Convergence rates for sparse chaos approximations of elliptic problems with stochastic coefficients, *IMA J. Numer. Anal.* 27 (2007), 232–261.

97. I. H. Sloan, What’s New in high-dimensional integration? – designing quasi-Monte Carlo for applications, In: Proceedings of the ICIAM, Beijing, China (L. Guo and Z. Ma eds), Higher Education Press, Beijing, 2015, pp. 365–386.

98. I. H. Sloan and S. Joe, Lattice Methods for Multiple Integration, Oxford University Press, Oxford, 1994.

99. I. H. Sloan, F. Y. Kuo, and S. Joe, Constructing randomly shifted lattice rules in weighted Sobolev spaces, *SIAM J. Numer. Anal.* 40 (2002), 1650–1665.

100. I. H. Sloan and A. V. Reztsov, Component-by-component construction of good lattice rules, *Math. Comp.* 71 (2002), 263–273.

101. I. H. Sloan, X. Wang, and H. Woźniakowski, Finite-order weights imply tractability of multivariate integration, *J. Complexity* 20 (2004), 46–74.

102. I. H. Sloan and H. Woźniakowski, When are quasi-Monte Carlo algorithms efficient for high-dimensional integrals?, *J. Complexity* 14 (1998), 1–33.

103. A. L. Teckentrup, P. Jantsch, C. G. Webster, and M. Gunzburger, A multilevel stochastic collocation method for partial differential equations with random input data, Preprint arXiv:1404.2647 (2014).

104. A. L. Teckentrup, R. Scheichl, M. B. Giles, and E. Ullmann, Further analysis of multilevel Monte Carlo methods for elliptic PDEs with random coefficient, *Numer. Math.* 125 (2013), 569–600.

105. X. Wang, Strong tractability of multivariate integration using quasi-Monte Carlo algorithms, *Math. Comp.* 72 (2002), 823–838.

106. D. Xiu and G. E. Karniadakis, Modeling uncertainty in flow simulations via generalized polynomial chaos, *J. Comput. Phys.* 187 (2003), 137–167.

107. T. Yoshiki, Bounds on Walsh coefficients by dyadic difference and a new Koksma-Hlawka type inequality for Quasi-Monte Carlo integration, arXiv:1504.03175 (2015).

108. D. Zhang, Stochastic Methods for Flow in Porous Media: Coping with Uncertainties, Academic Press, San Diego, 2002.