algorithms

MDPI

Article

p-Refined Multilevel Quasi-Monte Carlo for Galerkin Finite Element Methods with Applications in Civil Engineering

Philippe Blondeel [1,*], Pieterjan Robbe [1], Cédric Van hoorickx [2], Stijn François [2], Geert Lombaert [2] and Stefan Vandewalle [1]

[1] Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium; pieterjan.robbe@kuleuven.be (P.R.); stefan.vandewalle@kuleuven.be (S.V.)

[2] Department of Civil Engineering, KU Leuven, Kasteelpark Arenberg 40, 3001 Leuven, Belgium; cedric.vanhoorickx@kuleuven.be (C.V.h.); stijn.francois@kuleuven.be (S.F.); geert.lombaert@kuleuven.be (G.L.)

* Correspondence: philippe.blondeel@kuleuven.be

Received: 20 March 2020; Accepted: 24 April 2020; Published: 28 April 2020

Abstract: Civil engineering applications are often characterized by a large uncertainty on the material parameters. Discretization of the underlying equations is typically done by means of the Galerkin Finite Element method. The uncertain material parameter can be expressed as a random field represented by, for example, a Karhunen–Loève expansion. Computation of the stochastic responses, i.e., the expected value and variance of a chosen quantity of interest, remains very costly, even when state-of-the-art Multilevel Monte Carlo (MLMC) is used. A significant cost reduction can be achieved by using a recently developed multilevel method: p-refined Multilevel Quasi-Monte Carlo (p-MLQMC). This method is based on the idea of variance reduction by employing a hierarchical discretization of the problem based on a p-refinement scheme. It is combined with a rank-1 Quasi-Monte Carlo (QMC) lattice rule, which yields faster convergence compared to the use of random Monte Carlo points. In this work, we developed algorithms for the p-MLQMC method for two dimensional problems. The p-MLQMC method is first benchmarked on an academic beam problem. Finally, we use our algorithm for the assessment of the stability of slopes, a problem that arises in geotechnical engineering, and typically suffers from large parameter uncertainty. For both considered problems, we observe a very significant reduction in the amount of computational work with respect to MLMC.

Keywords: Multilevel Monte Carlo; Multilevel Quasi-Monte Carlo; h- and p-refinement; uncertainty quantification; structural engineering; geotechnical engineering

1. Introduction

There is an increasing need to accurately simulate and compute solutions to engineering problems, whilst accounting for model uncertainties. Methods for uncertainty quantification and propagation in engineering disciplines can be categorized into two groups: non-sampling and sampling methods.

Examples of non-sampling methods are the perturbation method and the Stochastic Galerkin Finite Element method. The perturbation method is based on a Taylor series expansion approximating the mean and variance of the solution [1]. While quite effective, its use is restricted to models with a limited number of relatively small uncertainties. The Stochastic Galerkin method, proposed by Ghanem and Spanos [2], is based on a spectral representation in the stochastic space. It transforms the uncertain coefficient partial differential equation (PDE) problem by means of a Galerkin projection technique into a large coupled system of deterministic PDEs. This method allows for somewhat larger numbers

of uncertainties and is quite accurate. However, it is highly intrusive and memory demanding, making its implementation cumbersome, and restricting its use to still rather low stochastic dimensions.

Sampling methods, on the other hand, are typically non-intrusive. Each sample corresponds to a deterministic solve, with a particular choice for the parameter values. Two popular examples are the Stochastic Collocation method, see [3], and the Monte Carlo (MC) method, see [4]. The former samples a stochastic PDE at a carefully selected multidimensional set of collocation points. After this sampling, a Lagrange interpolation is performed, leading to a polynomial response surface. From this response surface, the relevant stochastic characteristics can easily be computed in a post-processing step. Like Stochastic Galerkin, the Stochastic Collocation method suffers from the curse of dimensionality: the computational cost grows exponentially with the number of random variables considered in the problem. The MC method on the other hand, selects its samples randomly and does not suffer from the curse of dimensionality. However, a drawback is its slow convergence as a function of the number of samples taken.

The convergence of Monte Carlo can be accelerated in a variety of ways. For example, an alternative non-random selection of sampling points can be used, as in Quasi-Monte Carlo [5,6] and Latin Hypercube [7] sampling methods. Other techniques, such as Multilevel Monte Carlo (MLMC) [8], Multilevel Quasi-Monte Carlo (MLQMC) [9] and its generalizations, see, e.g., [10,11], can speed up the method. These improved Monte Carlo methods are based on a hierarchy of meshes with increasing resolution, where samples on coarser meshes are computationally less expensive than samples on finer meshes.

For completeness, we mention that there also exist hybrid variants which exhibit both a sampling and non-sampling character. This type of methods combine, for example, the Stochastic Finite Element methodology with Monte Carlo sampling or a multi-dimensional cubature method, see, e.g., [12,13].

In previous work, we presented a comparison between Monte Carlo and Multilevel Monte Carlo [14], and between Multilevel Monte Carlo and Multilevel Quasi-Monte Carlo [15]. In this work, we will present a novel multilevel method: p-refined Multilevel Quasi-Monte Carlo (p-MLQMC). This method reduces the cost of the simulation by taking samples on a hierarchy of nested p-refined meshes, and by selecting the sample points in a non-random way.

In [16], the authors introduced a multi-order Monte Carlo algorithm (MOMC) for computing the statistics of quantities of interest described by linear hyperbolic equations with stochastic parameters. The MOMC method relies on a hierarchy of p-refined meshes, similar to the setup in this paper. A numerical analysis of MOMC, including numerical results for dispersion in long-distance propagation of waves subjected to uncertainty can be found in [16]. Our p-MLQMC method is also based on a hierarchy of p-refined meshes with the addition of a deterministic sampling rule. Our focus lies on the implementation and on the practicalities of p-MLQMC. We discuss how the uncertainty represented as a random field needs to be taken into account in the discretized versions of our civil/geotechnical model problem so as to ensure a minimal total computational cost. The MOMC method has a computational cost increase in function of a user requested tolerance ϵ, proportional to ϵ^{-2}, while for p-MLQMC this is close to ϵ^{-1}, because of the use of a deterministic sampling rule.

First we will introduce the multilevel methods and detail the technicalities pertaining to p-MLQMC. Here, we also discuss how the uncertainty is modeled and incorporated in our models. Second, we will present the two model problems on which we will test and benchmark our method. The model problems consist of an academic beam bending problem and a more complicated slope stability problem. Last, we present our results obtained with p-MLQMC.

2. p-Refined Multilevel Quasi-Monte Carlo

The Multilevel Monte Carlo method and Multilevel Quasi-Monte Carlo method are extensions of the standard Monte Carlo method, see, e.g., [8,9,17]. These methods rely on a clever combination of many computationally cheap low resolution samples and a relatively small number of higher resolution, but computationally more expensive samples. MLMC and MLQMC require a predefined hierarchy of meshes in order to work properly. Classically, the hierarchy of the meshes is based on

a successive spatial refinement of the mesh per level, known as h-refinement. Hence, we will refer to classical MLMC and MLQMC as h-ML(Q)MC. Our p-MLQMC method, based on the MLQMC method, uses a spatial mesh refinement based on increasing the element's polynomial order, i.e., p-refinement. In this section we will first give an overview of the main characteristics of the MLMC method, whereafter we give an overview of the MLQMC method, and highlight the differences with respect to the MLMC method. We then give an account of the mesh hierarchies before introducing the p-MLQMC algorithm. We conclude with a general complexity theorem for ML(Q)MC sampling.

2.1. The Multilevel and Multilevel Quasi Monte Carlo Methods

Let $\mathbb{E}[P_L(\mathbf{x}^{(n)})]$, or $\mathbb{E}[P_L]$ for short, be the expected value of a particular quantity of interest P depending on a set of random variables $\mathbf{x}_L^{(n)}$, discretized on mesh L. The standard MC estimator for $\mathbb{E}[P_L]$ using N_L samples on mesh L, denoted as Q_L^{MC}, can be written as

$$Q_L^{\mathrm{MC}} := \frac{1}{N_L} \sum_{n=1}^{N_L} P_L(\mathbf{x}_L^{(n)}) \,. \tag{1}$$

The MLMC method, on the other hand, starts from a reformulation of $\mathbb{E}[P_L]$ as a telescoping sum. The expected value of the quantity of interest on the finest mesh L is expressed as the expected value of the quantity of interest on the coarsest mesh, plus a series of correction terms (or *differences*), i.e.,

$$\mathbb{E}[P_L] = \mathbb{E}[P_0] + \sum_{\ell=1}^{L} \mathbb{E}[P_\ell - P_{\ell-1}] = \sum_{\ell=0}^{L} \mathbb{E}\left[\Delta P_\ell\right], \tag{2}$$

where we defined $\Delta P_\ell := P_\ell(\mathbf{x}_\ell) - P_{\ell-1}(\mathbf{x}_\ell)$ with $P_{-1} := 0$.

Each term on the right-hand side is then estimated separately by a standard MC estimator with N_ℓ samples, i.e.,

$$Q_L^{\mathrm{MLMC}} := \frac{1}{N_0} \sum_{n=1}^{N_0} P_0(\mathbf{x}_0^{(n)}) + \sum_{\ell=1}^{L} \left\{ \frac{1}{N_\ell} \sum_{n=1}^{N_\ell} \left(P_\ell(\mathbf{x}_\ell^{(n)}) - P_{\ell-1}(\mathbf{x}_\ell^{(n)}) \right) \right\}, \tag{3}$$

where Q_L^{MLMC} is the MLMC estimator for the expected value $\mathbb{E}[P_L]$, which is a discrete approximation for the expected value of the quantity of interest, $\mathbb{E}[P]$. The MLMC estimator in Equation (3) can be written as a sum of $L+1$ estimators for the expected value of the difference on each level, i.e.,

$$Q_L^{\mathrm{MLMC}} = \sum_{\ell=0}^{L} \frac{1}{N_\ell} \sum_{n=1}^{N_\ell} \Delta P_\ell^{(n)}, \tag{4}$$

where we defined $\Delta P_\ell^{(n)} := P_\ell(\mathbf{x}_\ell^{(n)}) - P_{\ell-1}(\mathbf{x}_\ell^{(n)})$ with $P_{-1}^{(n)} := 0$.

Because of the telescoping sum property, i.e., Equation (2), the MLMC estimator is an unbiased estimator for the quantity of interest on the finest mesh, i.e.,

$$\mathbb{E}[Q_L^{\mathrm{MLMC}}] = \mathbb{E}[P_L]. \tag{5}$$

The error is controlled by imposing a tolerance ϵ^2 on the mean square error (MSE) of the estimator, defined as

$$\begin{aligned} \mathrm{MSE}(Q_L^{\mathrm{MLMC}}) &:= \mathbb{E}\left[\left(Q_L^{\mathrm{MLMC}} - \mathbb{E}\left[P\right] \right)^2 \right] \\ &= \mathbb{V}\left[Q_L^{\mathrm{MLMC}} \right] + \left(\mathbb{E}\left[Q_L^{\mathrm{MLMC}} \right] - \mathbb{E}\left[P\right] \right)^2 \\ &= \mathbb{V}\left[Q_L^{\mathrm{MLMC}} \right] + \left(\mathbb{E}\left[P_L - P\right] \right)^2 . \end{aligned} \tag{6}$$

The right-hand side of Equation (6) consists of two parts: the variance of the estimator, $\mathbb{V}\left[Q_L^{\text{MLMC}}\right]$, and the squared bias, $\left(\mathbb{E}\left[P_L - P\right]\right)^2$. The stopping criterion for multilevel schemes is typically based on the requirements that both terms are less than $\epsilon^2/2$. The root mean square error (RMSE) is defined as, $\text{RMSE}(Q_L^{\text{MLMC}}) = \sqrt{\text{MSE}(Q_L^{\text{MLMC}})}$.

The variance of the estimator satisfies

$$\mathbb{V}[Q_L^{\text{MLMC}}] = \sum_{\ell=0}^{L} \frac{V_\ell}{N_\ell}, \tag{7}$$

with $V_\ell := \mathbb{V}\left[\Delta P_\ell\right]$ the variance of the difference, which is used to determine the number of samples N_ℓ needed on each level ℓ. This is done by following the classic argument by Giles in [8], where the total cost of the MLMC estimator,

$$\text{cost}(Q_L^{\text{MLMC}}) = \sum_{\ell=0}^{L} N_\ell C_\ell, \tag{8}$$

is minimized, with C_ℓ denoting the cost to compute a single realization of the difference $\Delta P_\ell^{(n)}$, subjected to the constraint

$$\sum_{\ell=0}^{L} \frac{V_\ell}{N_\ell} \leq \frac{\epsilon^2}{2}. \tag{9}$$

When treating the N_ℓ as continuous variables, one finds

$$N_\ell = \frac{2}{\epsilon^2}\sqrt{\frac{V_\ell}{C_\ell}} \sum_{\ell=0}^{L} \sqrt{V_\ell C_\ell}. \tag{10}$$

The second term in Equation (6) is used to determine the maximum number of levels L. A typical MLMC implementation is level-adaptive, i.e., starting from a coarse finite element mesh, finer meshes are only added if required to reach a certain accuracy. Assuming that the convergence $\mathbb{E}[P_\ell] \to \mathbb{E}[P]$ is bounded as $|\,\mathbb{E}[P_\ell - P]| = \mathcal{O}(2^{-\alpha\ell})$, then we can use the heuristic

$$|\mathbb{E}[P_L - P]| = \left| \sum_{\ell=L+1}^{\infty} \mathbb{E}[\Delta P_\ell] \right| \approx \frac{|\mathbb{E}[\Delta P_L]|}{2^\alpha - 1} \tag{11}$$

and check for convergence using $|\,\mathbb{E}[P_L - P_{L-1}]|/(2^\alpha - 1) \leq \epsilon/\sqrt{2}$, see [8] for details.

The MLQMC estimator has a similar formulation as MLMC. It differs in that it uses an average over a number of shifts R_ℓ on each level, and a set of deterministic sample points $\mathbf{x}_\ell^{(r,n)}$, i.e.,

$$Q_L^{\text{MLQMC}} = \frac{1}{R_0}\sum_{r=1}^{R_0}\frac{1}{N_0}\sum_{n=1}^{N_0} P_0(\mathbf{x}_0^{(r,n)}) + \sum_{\ell=1}^{L}\frac{1}{R_\ell}\sum_{r=1}^{R_\ell}\left\{\frac{1}{N_\ell}\sum_{n=1}^{N_\ell}\left(P_\ell(\mathbf{x}_\ell^{(r,n)}) - P_{\ell-1}(\mathbf{x}_\ell^{(r,n)})\right)\right\}. \tag{12}$$

This can again be rewritten as a sum of $L+1$ estimators, analogous as in Equation (4),

$$Q_L^{\text{MLQMC}} = \sum_{\ell=0}^{L}\frac{1}{R_\ell}\sum_{r=1}^{R_\ell}\frac{1}{N_\ell}\sum_{n=1}^{N_\ell}\Delta P_\ell^{(r,n)}, \tag{13}$$

where we defined $\Delta P_\ell^{(r,n)} := P_\ell(\mathbf{x}_\ell^{(r,n)}) - P_{\ell-1}(\mathbf{x}_\ell^{(r,n)})$ with $P_{-1}^{(r,n)} := 0$.

The major difference between MLMC and MLQMC is the choice of the sample points. For the MLMC estimator in Equation (3), the sample points $\mathbf{x}_\ell^{(n)}$ are chosen at random from a given distribution. For MLQMC however, the sample points $\mathbf{x}_\ell^{(r,n)}$ are optimally chosen in the stochastic space, based on a deterministic rule. Here, we use a rank-1 lattice rule, similar to [9]. By using these deterministic points, three practicalities must be addressed in order for MLQMC to work.

The first practicality pertains to the additional bias that is being generated on the stochastic quantities of the computed solution when using these deterministic points. In order to recover unbiased estimates, a number of random shifts $\Xi_r, r = 1, 2, ..., R_\ell$ with $\Xi_r \in [0,1]^s$, has to be introduced on each level. This leads to the extra summation in the MLQMC estimator in Equation (12), see Section 2.9 of [18].

The representation of the MLQMC sample points with inclusion of the aforementioned random shifts is as follows:

$$\mathbf{x}^{(r,n)} = \text{frac}\left(\frac{n}{N}\mathbf{z} + \Xi_r\right) \quad \text{for} \quad n = 0, ..., N-1, \tag{14}$$

where $\text{frac}(x) = x - \lfloor x \rfloor, x > 0$, $\mathbf{z}$ is an s-dimensional vector of positive integers and N is the number of points in the lattice rule.

The second practicality relates to the number of points computed according to Equation (14). It is clear that the total number of points N needs to be known beforehand and thus cannot increase adaptively. Such a type of *closed* lattice rule cannot easily be used in an adaptive algorithm where additional samples are added on the fly. By rewriting Equation (14) as

$$\mathbf{x}^{(r,n)} = \text{frac}\left(\phi_2(n)\mathbf{z} + \Xi_r\right) \quad \text{for} \quad n \in \mathbb{N}, \tag{15}$$

with ϕ_2 the radical inverse function in base 2, it is however possible to obtain an *open* lattice rule, i.e., removing the need to know N beforehand [19]. The radical inverse function in base 2 mirrors the binary representation of a number around its binary point ensuring that $\phi_2(n) \in [0,1)$.

The third practicality concerns the computation of the points $\mathbf{x}^{(r,n)}$. These points are spatially located in the unit cube, that is $\mathbf{x}^{(r,n)} \in [0,1)^s$. In order to use these points in a Monte Carlo simulation that uses normally distributed random numbers, every set of points must be mapped from the $[0,1)^s$ to $\mathbb{R}^s$. For this the inverse, Φ^{-1}, of the univariate standard normal cumulative distribution function, $\Phi(y) = \int_{-\infty}^{y} \exp\left(-t^2/2\right)/\sqrt{2\pi} dt$, is applied point wise to Equation (15), see [20]. This is written as

$$\mathbf{x}^{(r,n)} = \Phi^{-1}\left(\text{frac}\left(\phi_2(n)\mathbf{z} + \Xi_r\right)\right) \quad \text{for} \quad n \in \mathbb{N}. \tag{16}$$

The discussion pertaining to MLMC is applicable to MLQMC, with the exception of the part related to the computation of the number of samples. Unlike MLMC, the number of samples for MLQMC is not based on a formula as in Equation (10). In order to satisfy the statistical constraint, i.e., Equation (9), an adaptive algorithm is used, see [9]. Starting with an initial number of samples, this algorithm multiplies the number of samples on the level with maximum ratio $\mathcal{V}_\ell / C_\ell$, with a constant factor until the variance of the estimator $\mathbb{V}\left[Q_L^{\text{MLQMC}}\right] = \sum_{\ell=0}^{L} \mathcal{V}_\ell$ is smaller than $\epsilon^2/2$. In our implementation this multiplication constant is chosen as 1.2, and $\mathcal{V}_\ell$ is computed as

$$\mathcal{E}_\ell := \frac{1}{R_\ell} \sum_{r=1}^{R_\ell} \left\{ \frac{1}{N_\ell} \sum_{n=1}^{N_\ell} \left(\Delta P_\ell^{(n,r)}\right) \right\}, \tag{17}$$

and

$$\mathcal{V}_\ell := \frac{1}{R_\ell(R_\ell - 1)} \sum_{r=1}^{R_\ell} \left\{ \left(\frac{1}{N_\ell} \sum_{n=1}^{N_\ell} \left(\Delta P_\ell^{(n,r)}\right) - \mathcal{E}_\ell \right)^2 \right\}. \tag{18}$$

The number of samples is multiplied by 1.2 (instead of the naive value 2) because, in the current setting where each sample involves the solution of a complex PDE, an exponential growth with a doubling of the number of samples is a dramatic event that causes huge jumps in the total cost of the estimator. Reducing this value from 2 to 1.2, as in, for example [21], is a compromise that causes a more gradual increase of the computational cost of the estimator, while, at the same time ensuring that enough progress is made. Note that increasing the amount of samples one at the time, i.e., the other extreme, would not be efficient in a parallel implementation, such as the one in this paper. We note

that if the updated number of samples is not an integer, the least integer greater than this value is taken in order to have an integer value as updated number of samples.

In this work, we use a shifted rank-1 lattice rule in order to generate the MLQMC sample points $\mathbf{x}_\ell^{(r,n)}$, as is commonly done in other work regarding MLQMC, see for example [9,10,22]. Rank-1 lattice rules belong to one of two major classes of QMC points, i.e. lattices. The other class consists of digital nets/sequences, of which Sobol' sequences are one of the oldest example, see [23]. One of the major differences between these two classes, is that rank-1 lattice rules rely on a generating vector $\mathbf{z}$ to construct the points, while digital nets/sequences rely on a set of generating matrices $C_1, ..., C_s$. The shifting procedure, also referred to as randomization, as done in Equation (14) for rank-1 lattice rules, also differs for both classes. Shifting in rank-1 lattice rules is mathematically easier than digital shifting or scrambling in digital nets/sequences. (Digital) shifting allows for a faster convergence rate compared to the standard Monte Carlo method if the function under consideration is sufficiently smooth. We note that scrambling, which is only applicable to digital nets/sequences, can further improve the QMC convergence rate. For a more thorough discussion on this subject, we refer to [18].

The generating vector $\mathbf{z}$ we use for p-MLQMC is available from [24]. This vector was constructed with the component-by-component (CBC) algorithm with decreasing weights, $\gamma_j = 1/j^2$.

2.2. Mesh Hierarchies

Multilevel methods rely on a hierarchy of levels to achieve variance reduction and, by doing so, reduce the total computational cost. In common practice, this hierarchy of levels is defined as a hierarchy of successively refined nested grids or meshes with a discretization based on the Finite Element Galerkin approximation. For a bounded domain Ω in $\mathbb{R}^2$, a mesh $\mathcal{T}$ on Ω is a partition into a number $M(\mathcal{T})$ of open disjoint subdomains $\mathcal{T} = \{T_j\}_j^{M(\mathcal{T})}$ with a mesh width h. We follow Ciarlet's definition of a finite element defined as triples $(T, \mathcal{P}, \mathcal{N})$, see [25].

Definition 1. *A finite element is defined as* $(T, \mathcal{P}, \mathcal{N})$ *with*

1. $T \subseteq \mathbb{R}$ *a closed subset with nonempty interior and piecewise smooth boundary, i.e., the element domain,*
2. $\mathcal{P}$ *a finite-dimensional space of functions on* T*, i.e., the space of shape functions, and*
3. $\mathcal{N} = (N_1, N_2, ..., N_k)$ *a basis for* $\mathcal{P}'$*, i.e., the set of nodal variables.*

Classically, the mesh hierarchy used for MLMC and MLQMC is based on an h-refinement scheme, i.e., a succession of meshes $\mathcal{T}_0, \mathcal{T}_\ell, ..., \mathcal{T}_L$ such that $h_0 > h_\ell > ... > h_L$ and $M(\mathcal{T}_0) < M(\mathcal{T}_\ell) < ... < M(\mathcal{T}_L)$. Figures 1 and 2 show a hierarchy of nested uniform h-refined meshes, for the numerical examples considered in Sections 3.1 and 3.3, where the nodal points are at the intersections of the line segments.

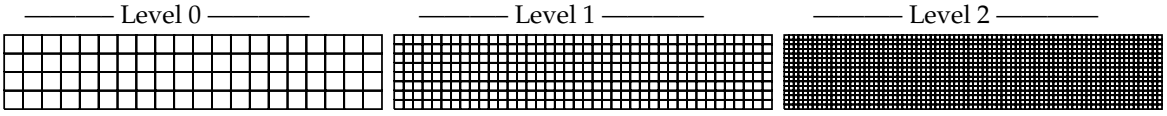

Figure 1. h-refined hierarchy of meshes used for model problem 1.

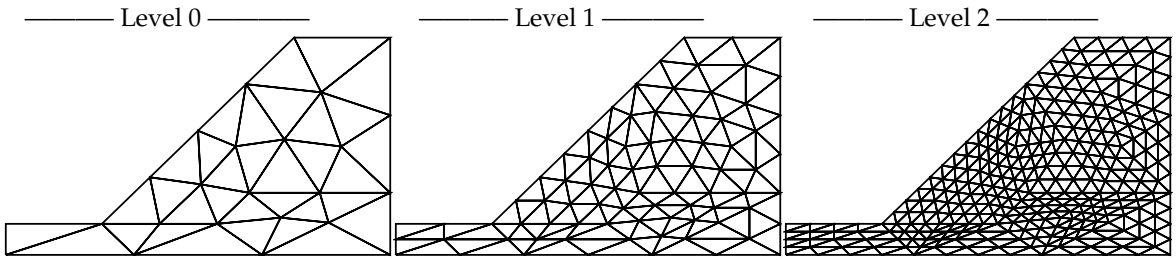

Figure 2. h-refined hierarchy of meshes used for model problem 2.

In the present paper, we propose to use a hierarchy of meshes based on a p-refinement scheme i.e., increasing the polynomial order of the interpolating shape functions, $(T, \mathcal{P}_1, \mathcal{N}_1) \in \mathcal{T}_0 < (T, \mathcal{P}_{\ell+1}, \mathcal{N}_{\ell+1}) \in \mathcal{T}_\ell < ... < (T, \mathcal{P}_{L+1}, \mathcal{N}_{L+1}) \in \mathcal{T}_L$, such that $\dim \mathcal{P}_1 < \dim \mathcal{P}_{\ell+1} < ... < \dim \mathcal{P}_{L+1}$. Figures 3 and 4 show a hierarchy of nested p-refined meshes, where the nodal points are represented as red dots.

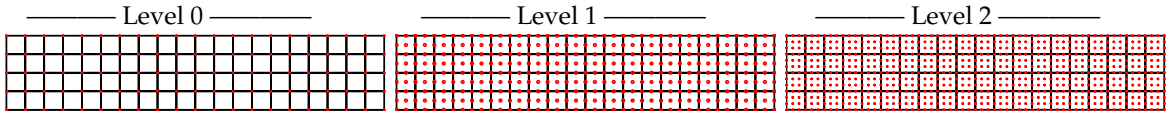

Figure 3. p-refined hierarchy of meshes used for model problem 1.

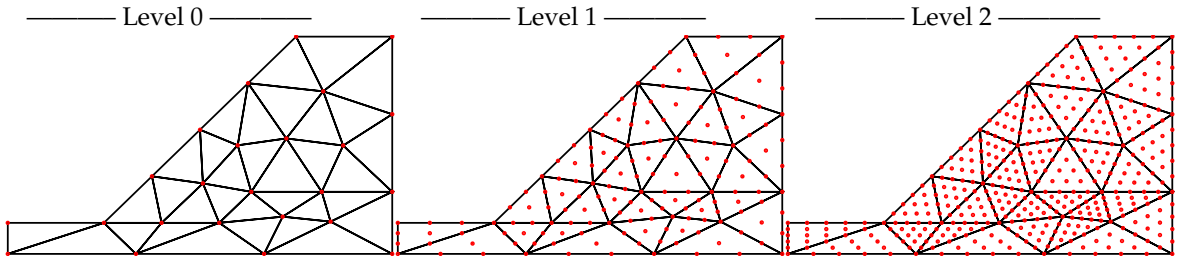

Figure 4. p-refined hierarchy of meshes used for model problem 2.

In this paper we will only consider two-dimensional uniform, Lagrange type elements. In order to construct both mesh refinement hierarchies, we use a combination of the open source mesh generator Gmsh [26] and MATLAB [27].

2.3. Algorithm

Algorithm 1 describes the implementation for p-MLQMC. The output produced by the algorithm consists of the expected value of the estimator for a given tolerance ϵ on the root mean square error. The algorithm computes the expected value of the estimator and its variance on each level according to Equations (17) and (18). The algorithm starts with $R_\ell = 10$ and $N_\ell = 2$ on each level ℓ. The value of R_ℓ is typically chosen to be small, in the range of 10 to 32, in order to have a fair comparison with MLMC, see ([28], page 3678) and ([9], page 6). In Algorithm 1, the bias B is evaluated by Equation (11). The expected value $\mathbb{E}\left[Q_L^{\text{MLQMC}}\right] = \sum_{\ell=0}^{L} \mathcal{E}_\ell$, where $\mathcal{E}_\ell$ follows from Equation (17).

For this p-MLQMC algorithm to be able to work well, it is important to adequately generate the mesh hierarchy based on p-refinement. The challenge consists of selecting adequate points, on each level, in which to compute the discrete values of the random field. We generate these points starting from a given Gauss quadrature set on the finest level, i.e., level 3 for both our model problems. This is illustrated for one element for model problem 1 in Figure 5 and for model problem 2 in Figure 6. The □ and △, respectively represent the location of the actual Gauss quadrature points for a square and a triangular element, while the ● represent the location of the points where the values of the random field will be computed. This approach contrasts with a more intuitive one, which would consist of selecting the location of the actual Gauss quadrature points for the generation of the values of the random field. We observed numerically that our approach yields a larger decay of the expected value and variance of the multilevel differences compared to choosing the actual Gauss quadrature points for sampling from the random field, since, in that case the sets of quadrature points on different levels are not contained in each other, i.e., $\square_0 \not\subseteq \square_\ell \not\subseteq ... \not\subseteq \square_L$. Such an inclusion property does hold however for our sets of random field evaluation points, i.e., $\bullet_0 \subseteq \bullet_\ell \subseteq ... \subseteq \bullet_L$. This implies that the sets of random values that are generated in these sets of points will also be included in one another. This will lead to a good decrease of $\mathbb{V}[\Delta P_\ell]$, as is required by the multilevel methodology. Note that on level 3, both □ and △ have the same spatial location as ●. We use the Gauss quadrature set on the finest mesh as a starting point for the construction of subsets on coarser meshes. The computed discrete random

values are used during the numerical integration of the element stiffness matrix by means of Gauss quadrature points, see Section 2.6 for further details.

Algorithm 1: p-MLQMC.

Data:
Meshes, Requested tolerance (ϵ).

$L = 0$, B $\leftarrow \infty$, $Var \leftarrow \infty$;
while $\frac{\epsilon}{\sqrt{2}} < B$ **do**
  if $L > 2$ **then**
    while $\frac{\epsilon^2}{2} < Var$ **do**
      Find the level ℓ with largest $\mathcal{V}_\ell / \mathrm{C}_\ell$ among all $\ell \in \{0, 1, ..., L\}$;
      $N_\ell \leftarrow N_\ell * 1.2$;
      Take N_ℓ samples on level ℓ for each random shift $\Xi_r, r = 1, 2, ..., R_L$;
      Calculate $\mathcal{E}_\ell$ and $\mathcal{V}_\ell$;
      $Var \leftarrow \sum_{\ell=0}^{L} \mathcal{V}_\ell$;
      Calculate B ;
  else
    Take N_L samples on level L for each random shift $\Xi_r, r = 1, 2, ..., R_L$;
    Compute C_L, and calculate $\mathcal{E}_L$ and $\mathcal{V}_L$ according to Equation (17) and (18);
    $Var \leftarrow \sum_{\ell=0}^{L} \mathcal{V}_\ell$;
  $L \leftarrow L + 1$;
error $\leftarrow \sqrt{Var + B^2}$;
$E \leftarrow \sum_{\ell=0}^{L} \mathcal{E}_\ell$;
return E, error

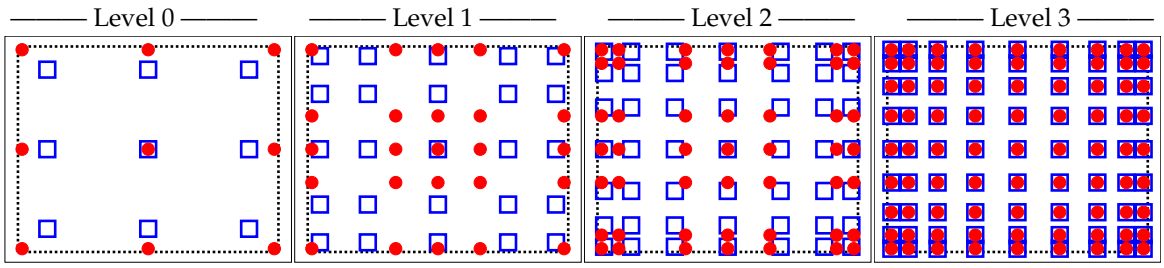

Figure 5. Quadrature points □ and points where the discrete values of the random field are to be generated on one element ● for model problem 1.

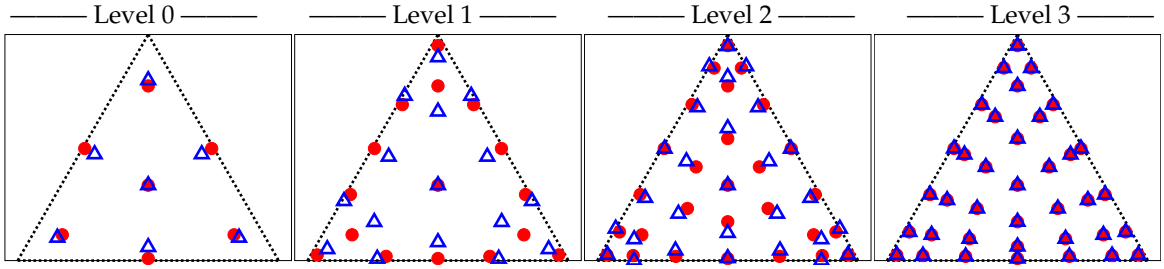

Figure 6. Quadrature points △ and points where the discrete values of the random field are to be generated on one element ● for model problem 2.

We will now describe an algorithm which generates the location of the points in which the discrete values of the random field are to be computed. Figures 5 and 6, show the location of these points ●. As input to this algorithm, we require a set of Gauss quadrature points in natural coordinates

for one element of the finest considered mesh. An algorithm for generating the point sets on each level is given in Algorithm 2. The output consists of a vector of points in which the discrete values of the random field are to be computed. This vector contains all the locations of the points on the finest level. Because the vector is sorted, taking subsets hereof gives the locations of the points on the coarser levels. Figure 7 shows in which sequence the points are chosen on a unit square and on a unit triangle. For example, for model problem 2, the evaluation points at the coarsest level are points $\{1, 2, 3, 4, 5, 6, 7\}$, the second coarsest level points $\{1, ..., 7, 8, 9, 10, 11, 12, 13, 14, 15, 16\}$, the third coarsest level points $\{1, ..., 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28\}$, and at the finest level points $\{1, ..., 28, 29, 30, 31, 32, 33, 34, 35, 36, 37\}$. See also Tables 1 and 2 which, among other, list the number of quadrature points used per level. Once the locations of the points for one element in natural coordinates for each level have been obtained, the points are mapped to each element of the mesh by means of a transformation matrix. This results in a point set, listing the locations in global coordinates where the discrete values of the random field for the model problem are to be computed. As such, Algorithm 2 only works for two dimensional problems. Algorithm 2 consists of two main loops, the first loop starts on line 3, and returns a set of points which are closest to points of a given quadrature set on level ℓ. The second loop, starting on line 12, reorders the obtained set of closest points as in Figure 7, following the principle illustrated in Figure 8. Extending Algorithm 2 for three dimensions can be done be rewriting lines 12 to 24 to accommodate a reordering in three dimensions or to not consider a reordering at all. This would mean that lines 12 to 24 have to be omitted when computing the locations for the points in three dimensions.

Table 1. Number of elements, degrees of freedom, element order and number of quadrature points for model problem 1.

	h-ML(Q)MC				p-ML(Q)MC			
Level	**Nel**	**DOF**	**Order**	**Nquad**	**Nel**	**DOF**	**Order**	**Nquad**
0	80	210	1	9	80	210	1	9
1	320	738	1	9	80	738	2	25
2	1280	2754	1	9	80	1586	3	49
3	5120	10626	1	9	80	2754	4	81
4	20480	41730	1	9	/	/	/	/

Table 2. Number of elements, degrees of freedom, element order and number of quadrature points for model problem 2.

	h-ML(Q)MC				p-ML(Q)MC			
Level	**Nel**	**DOF**	**Order**	**Nquad**	**Nel**	**DOF**	**Order**	**Nquad**
0	33	48	1	7	33	48	1	7
1	132	160	1	7	33	338	3	16
2	528	582	1	7	33	892	5	28
3	2112	2218	1	7	33	1720	7	37
4	8448	8658	1	7	/	/	/	/

In its current form, Algorithm 1 is level-adaptive, in the sense that levels are added to the estimator ($L \leftarrow L + 1$) only when needed to reduce the bias. This means that the maximum level L that satisfies the bias constraint is not known a priori. The p-MLQMC method in Algorithm 1 takes as input a number of meshes $\{\mathcal{T}_\ell\}_{\ell=0...L}$ with maximum L, generated during a preprocessing step using the point selection method in Algorithm 2. If the bias constraint were not satisfied with the predefined maximum level L, it would force us to recompute a hierarchy of meshes $\{\mathcal{T}_\ell\}_{\ell=0...L}$ with a higher level L by means of Algorithm 2, and restart Algorithm 1 with these new meshes. The implication hereof is that samples from earlier runs of Algorithm 1 with maximum level L cannot be reused when restarting Algorithm 1 with maximum level $L + 1$. The reason is that the inclusion principle is needed for the locations where

discrete values of the random field are generated, as stated here above. In our implementation, we run Algorithm 2 only once for an a priori determined maximum level L, so that the set of points where the random field must be computed on each level remains the same during the course of Algorithm 1.

Algorithm 2: Point Set Generation

Data:
Max level L, Element type, Quadrature point set per level $\{Q_\ell\}_{\ell=0}^{L}$ sorted with respect to their ascending x-coordinate value with number of points $\{K_\ell\}_{\ell=0}^{L}$

$P_L \leftarrow Q_L$;
$\ell \leftarrow L-1$;
while $\ell \geq 0$ **do**
  $i \leftarrow 1$;
  $P_\ell \leftarrow \varnothing$;
  while $i \leq K_\ell$ **do**
    Find the point $p \in P_{\ell+1}$,which is not in P_ℓ, closest to the i^{th} point of Q_ℓ ;
    $P_\ell \leftarrow P_\ell \cup \{p\}$; // Add it to the array
    $i \leftarrow i+1$;
  $\ell \leftarrow \ell - 1$;
$\ell \leftarrow 0$;
while $\ell \leq L$ **do**
  $F_\ell \leftarrow P_\ell \subset$ Area 1; // Add to set F_ℓ the points from P_ℓ that are contained by Area 1, see Figure 8;
  $P_\ell \leftarrow \varnothing$ // Clear P_ℓ;
  $i \leftarrow 1$;
  $M \leftarrow \text{length}(F_\ell)$;
  while $i \leq M$ **do**
    $p \leftarrow$ the i^{th} point of F_ℓ;
    while $p \notin P_\ell$ **do**
      $P_\ell \leftarrow P_\ell \cup \{p\}$;
      Find p_{new} that is rotated over the element's center with respect to p with an angle of
        - 90 degrees if the element is a rectangle, **OR**
        - 120 degrees if the element is a triangle, // See Figure 8
      $p \leftarrow p_{new}$;
    $i \leftarrow i+1$
  $\ell \leftarrow \ell + 1$
Rearrange $\mathcal{P}$ such that it is sorted as follows:
$\mathcal{P} \leftarrow \{P_0, (P_1 \setminus P_0), ..., (P_\ell \setminus P_{\ell-1}), ..., (P_L \setminus P_{L-1})\}$;
return $\mathcal{P}$

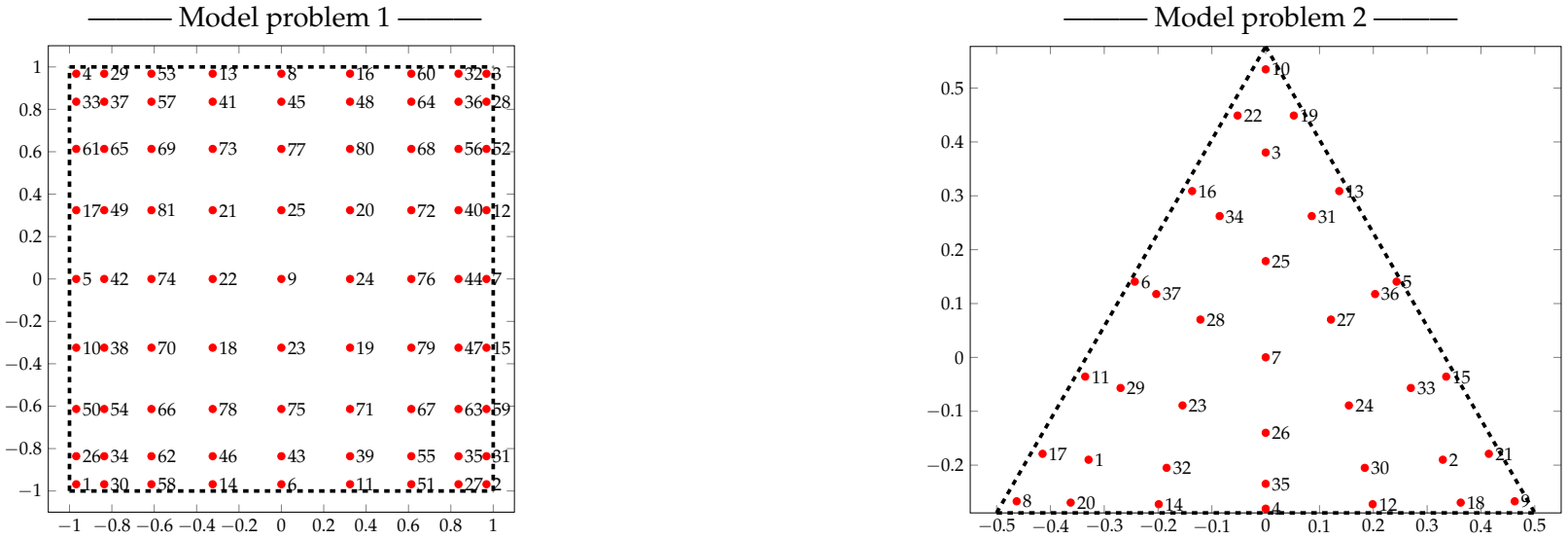

Figure 7. Sequence of points on a unit element for model problem 1 and 2.

Figure 8. Areas of a unit element and rotation of one point from Area 1 for model problem 1 and 2.

2.4. Cost Complexity Theorem

Having introduced the p-MLQMC method, we now present a complexity theorem for MLQMC, which also covers the MLMC method, when $\delta = 1$. More details can be found in [22] and in ([29], page 76). We refer to [8] for the original MLMC complexity theorem and its generalization on which the current theorem is based, see [17].

Theorem 1. *Given the positive constants* $\alpha, \beta, \gamma, c_1, c_2, c_3$ *such that* $\alpha \geq \frac{1}{2}\min\left(\beta, \delta^{-1}\gamma\right)$ *with* $\delta \in (1/2, 1]$, *and assume that the following conditions hold:*

1. $|\mathbb{E}[P_\ell - P]| \leq c_1 2^{-\alpha \ell}$,
2. $\mathbb{V}\left[Q_\ell^{\mathrm{ML(Q)MC}}\right] \leq c_2 2^{-\beta \ell} N_\ell^{-1/\delta}$ *and*
3. $C_\ell \leq c_3 2^{\gamma \ell}$.

Then, there exists a positive constant c_4 *such that for any* $\epsilon < \exp(-1)$ *there exists an* L *and a sequence* $\{N_\ell\}_{\ell=0}^{L}$ *for which the multilevel estimator,* $Q_L^{\mathrm{ML(Q)MC}}$ *has an* MSE $\leq \epsilon^2$, *and*

$$\mathrm{cost}(Q^{\mathrm{ML(Q)MC}}) \leq \begin{cases} c_4 \epsilon^{-2\delta} & \text{if} \quad \delta\beta > \gamma, \\ c_4 \epsilon^{-2\delta} \left(\log\ \epsilon\right)^{1+\delta} & \text{if} \quad \delta\beta = \gamma, \\ c_4 \epsilon^{-2\delta - (\gamma - \delta\beta)/\alpha} & \text{if} \quad \delta\beta < \gamma. \end{cases} \tag{19}$$

The constant α in Assumption 1 of Theorem 1, represents the rate at which $\mathbb{E}\left[\Delta P_\ell\right]$ decreases with increasing level ℓ. The constant β in Assumption 2, stands for the decay rate of $\mathcal{V}_\ell$, i.e., the variance of the estimator of the difference on level ℓ. The constant γ in Assumption 3, is determined by the efficiency of the solver. This factor will be different for h-refinement and p-refinement. All three factors will be estimated on the fly in our numerical experiments. The parameter δ depends on the Quasi-Monte Carlo (QMC) rule used.

The theorem can now be interpreted as follows. When the variance $\mathcal{V}_\ell$ decreases faster with increasing level ℓ than the cost increases, i.e., $\delta\beta > \gamma$, the dominant computational cost is located on the coarsest level, which is small because C_0 is small. Conversely, if the variance decreases slower with increasing level ℓ than the cost increases, i.e., $\delta\beta < \gamma$, the dominant computational cost will be located on the finest level. The cost will be small because $\mathcal{V}_L$ is small.

Observe that if $\mathbb{E}[P_\ell] \to \mathbb{E}[P]$, then $V_\ell \to 0$, see Equation (7), as ℓ increases. Hence, the number of samples N_ℓ will be a decreasing function of ℓ. This means that most samples will be taken on the coarse mesh, where samples are cheap, whereas increasingly fewer samples are required on the finer, but more expensive meshes. In practice, the number of samples must be truncated to $\lceil N_\ell \rceil$, the smallest integer larger than or equal to N_ℓ.

Using Equation (10), the total cost of the MLMC estimator, from Equation (8), can be written as

$$\mathrm{cost}(Q^{\mathrm{MLMC}}) = \frac{2}{\epsilon^2}\left(\sum_{\ell=0}^{L}\sqrt{V_\ell C_\ell}\right)^2. \tag{20}$$

Following this theorem, we note that the optimal cost of the MLMC estimator, which corresponds to $\delta = 1$, is proportional to ϵ^{-2} when the variance over the levels decreases faster than the cost per level increases, i.e., $\beta > \gamma$. Similarly for the MLQMC estimator, we can hope to achieve an optimal cost proportional to ϵ^{-1}. Note that this is only true in the limit when $\delta \to 1/2$. We will show in our numerical experiments that the theoretically derived asymptotic cost complexity is close to what we observe. Quasi-Monte Carlo methods work so well because they are based on the concept of *weighted* function spaces and *low effective dimension*. Not all higher stochastic dimensions are of equal importance. A series of decreasing weights can be used to quantify this importance. The idea is then to analyze the problem at hand in this *weighted* function space. The weights can be used to design a

good lattice rule that yields a good convergence of the QMC method. A more thorough analysis can be found in [28,30,31].

2.5. Modeling the Spatial Variability

The uncertain spatial variability of the Young's modulus (model problem 1) and the soil's cohesion (model problem 2) is represented by means of a random field. The Young's modulus is represented as a gamma random field, while the soil's cohesion is represented as a lognormal random field. The starting point for the construction of both fields is the generation of a (truncated) Gaussian random field. This is done by using a Karhunen–Loève (KL) expansion, see [32].

Consider a Gaussian random field $Z(\mathbf{x}, \omega)$, where ω is a random variable with a given covariance kernel. For both cases we choose the Matérn covariance kernel,

$$C(\mathbf{x}, \mathbf{y}) := \sigma^2 \frac{1}{2^{\nu-1}\Gamma(\nu)} \left(\sqrt{2\nu} \frac{\|\mathbf{x}-\mathbf{y}\|_2}{\lambda} \right)^{\nu} K_{\nu} \left(\sqrt{2\nu} \frac{\|\mathbf{x}-\mathbf{y}\|_2}{\lambda} \right), \tag{21}$$

with ν the smoothness parameter, K_{ν} the modified Bessel function of the second kind, σ^2 the variance and λ the correlation length. The parameter values, ν, λ and σ will be given later on for both cases. The KL expansion can be formulated as:

$$Z(\mathbf{x}, \omega) = \overline{Z}(\mathbf{x}) + \sum_{n=1}^{\infty} \sqrt{\theta_n} \xi_n(\omega) b_n(\mathbf{x}). \tag{22}$$

Here, $\overline{Z}(\mathbf{x})$ denotes the mean of the field, and is set to zero. The $\xi_n(\omega)$ denote i.i.d. standard normal random variables. The symbols θ_n and $b_n(\mathbf{x})$ respectively denote the eigenvalues and eigenfunctions of the covariance kernel corresponding to Equation (21). They are the solutions of the eigenvalue problem

$$\int_D C(\mathbf{x}, \mathbf{y}) b_n(\mathbf{y}) \mathrm{d}\mathbf{y} = \theta_n b_n(\mathbf{x}). \tag{23}$$

These can be approximated by means of a numerical collocation scheme, i.e., by solving

$$\int_D C(\mathbf{x}_k, \mathbf{y}) b_n(\mathbf{y}) \mathrm{d}\mathbf{y} = \theta_n b_n(\mathbf{x}_k), \quad k = 1, 2, \ldots, M, \tag{24}$$

in some well-chosen collocation points $\mathbf{x}_k$. Following the Nyström method, see [33], the integral in Equation (24), is approximated by a numerical integration scheme which uses the collocation points as quadrature nodes, i.e.,

$$\sum_{q=1}^{M} w_q C(\mathbf{x}_k, \mathbf{y}_q) \widetilde{b_n}(\mathbf{y}_q) = \widetilde{\theta}_n \widetilde{b_n}(\mathbf{x}_k), \quad k = 1, 2, \ldots, M. \tag{25}$$

In matrix notation, this becomes

$$\Sigma W \widetilde{B}_n = \widetilde{\theta}_n \widetilde{B}_n, \tag{26}$$

where Σ is a symmetric positive semi-definite matrix with entries $\Sigma_{k,q} = C(\mathbf{x}_k, \mathbf{y}_q)$, W is a diagonal matrix containing the weights w_q on its diagonal and $\widetilde{B}_n$ is a vector with entries $\widetilde{B}_{n,q} = \widetilde{b_n}(\mathbf{y}_q)$. The matrix eigenvalue problem, Equation (26), can be reformulated as

$$\Psi \widetilde{B}_n^* = \widetilde{\theta}_n \widetilde{B}_n^*, \tag{27}$$

where $\widetilde{B}_n^* = \sqrt{W}\widetilde{B}_n$ and $\Psi = \sqrt{W}\Sigma\sqrt{W}$. Matrix Ψ is symmetric positive semi-definite. This implies that the eigenvalues $\widetilde{\theta}_n$ are nonnegative real and the eigenvectors $\widetilde{B}_n^*$ are orthogonal to each other. Using Equation (25), the Nyström eigenfunctions $\widetilde{b_n}(\mathbf{x})$ are obtained:

$$\widetilde{b}_n(\mathbf{x}) = \frac{1}{\widetilde{\theta}_n} \sum_{q=1}^{M} \sqrt{w_q} \widetilde{B}^*_{n,q} C(\mathbf{x}, \mathbf{y}_q), \tag{28}$$

where $\widetilde{B}^*_{n,q}$ stands for the q-th element of eigenvector $\widetilde{B}^*_n$. These eigenvalues and eigenfunctions can be used as an approximate eigenpair in the KL expansion after a suitable normalization.

In an actual implementation, the number of KL terms in Equation (22) is truncated to a finite value s, representing the stochastic dimension, i.e.,

$$Z(\mathbf{x}, \omega) = \overline{Z}(\mathbf{x}) + \sum_{n=1}^{s} \sqrt{\theta_n} \xi_n(\omega) b_n(\mathbf{x}) \,. \tag{29}$$

The transformation of a Gaussian random field to respectively a gamma and a lognormal random field is detailed in Sections 3.1.4 and 3.3.4.

2.6. Incorporating the Uncertainty in the Model

By using the Galerkin–Ritz variational formulation of the underlying weak form of the PDE describing the displacement, a system of equations as in Sections 3.1.2 or 3.3.2 is obtained. This discrete form allows one to incorporate the uncertainty, modeled by means of the random fields, in the finite element model. This is achieved by assigning the discrete values of the random field to each element. For h-ML(Q)MC, this is accomplished by means of the midpoint approach, i.e., the value of the random field is taken constant within each individual element and equal to the value of the realization of the random field at the center point of the element [34]. For p-ML(Q)MC, this is accomplished by means of the integration point method, i.e., the uncertainty is taken into account by its closest quadrature point, see Figures 5 and 6, when numerically integrating the element stiffness matrix by use of Gauss quadrature points [35]. In Figure 9, we show the realization of a gamma random field for three successive levels in case of h-refinement. Figure 10 shows a realization of a lognormal random field for model problem 2 on four successive levels.

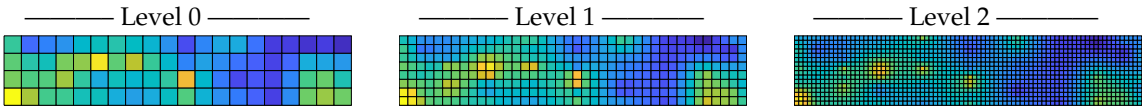

Figure 9. Three realizations of a gamma random field.

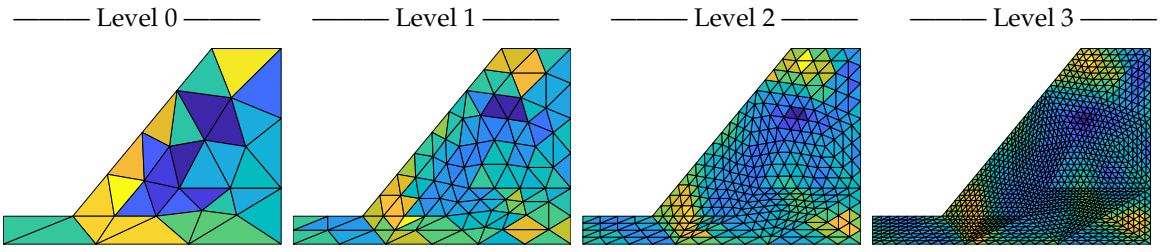

Figure 10. Four realizations of a lognormal random field.

It should be stressed that because of the telescoping sum property, i.e., Equation (2), we are free to choose the stochastic model on the different levels, i.e., we do not need to resolve all fine scale features of the random field on the coarse meshes. This is the case, for example, if we vary the number of KL terms, or equivalently, if we use random fields with an artificially enlarged correlation length. The advantage of the latter strategy is that it typically leads to improved convergence rates, α, and β. We refer to [36] Section 4, where this technique was successfully used for highly oscillatory random fields that vary on a fine scale. The artificial smoothing of the random field allows one to choose much coarser meshes than would otherwise be expected from the correlation length of the random field. A complete error analysis can be found in [36,37].

3. Model Problems and Numerical Results

In this section, we describe two model problems, and discuss some numerical results. For both, we benchmark the p-MLQMC algorithm against standard MLMC. First, we introduce the problem. We describe the model, and then expand on the technicalities of the finite element method. Hereafter we introduce the mesh hierarchies. We continue by elaborating on the construction of the model problem specific random fields and conclude by giving the quantity of interest (QoI). Second we present the numerical results. We start by giving estimates for the rates (α, β, γ) from Theorem 1 in Section 2.4. Next, we present the number of samples for the finest considered tolerance on the MSE. Then, we show how the uncertainty propagates towards the solution, i.e., the uncertainty that is present on the QoI. Lastly, we compare our p-MLQMC algorithm against the different other methods in terms of total simulation time. We first fully discuss the results for model problem 1 before following the same approach for model problem 2.

All the results have been computed on a workstation equipped with 2 physical cores, Intel Xeon E5-2680 v3 CPU's, each with 12 logical cores, clocked at 2.50 GHz, and a total of 128 GB RAM.

3.1. Model Problem 1

3.1.1. Description

The academic problem we consider consists of the deflection of a cantilever beam clamped at both sides as seen in Figure 11, assuming plane stress, i.e., no stress component in its depth. The beam has a length of 5.0 m, a height of 1.0 m, and a width of 0.25 m. The material is concrete with parameters as follows: a mass density of 2500 kg/m^3, a Poisson ratio of 0.15 and a Young's modulus subject to uncertainty, as specified below. The load of 10.000 kN is acting at mid-span. We consider a linear elastic and isotropic material model for this problem.

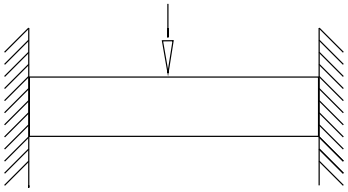

Figure 11. Cantilever beam clamped at both ends loaded in the middle.

3.1.2. Finite Element Method

The Finite Element method will be used to compute the displacement of the beam assuming plane stress. An equidistant, regular rectangular mesh consisting of Lagrange quadrilateral elements is employed. The underlying equations and solution methods are reviewed hereunder.

The system equation is of the form

$$\mathbf{K}\underline{\mathbf{u}} = \underline{\mathbf{f}}, \tag{30}$$

with $\mathbf{K}$ the global stiffness matrix, $\underline{\mathbf{f}}$ the global nodal force vector and $\underline{\mathbf{u}}$ the displacement. The global stiffness matrix and nodal force vector are obtained from the element stiffness matrices $\mathbf{K^e}$ and the element force vectors $\underline{\mathbf{f}}^{\mathbf{e}}$. These are computed numerically by evaluation of the following integrals by means of Gauss quadrature:

$$\mathbf{K^e} = \int_{\Omega} \mathbf{B}^T \mathbf{D} \mathbf{B} d\Omega \quad \text{and} \quad \underline{\mathbf{f}}^{\mathbf{e}} = \int_{\Gamma_t} \mathbf{N}^T \bar{\mathbf{t}}_n d\Gamma_t. \tag{31}$$

The element nodal force vector $\underline{\mathbf{f}}^{\mathbf{e}}$ is modeled as a Neumann boundary condition, where $\bar{\mathbf{t}}_n$ stands for the surface traction, specified as a force per unit area, and $\mathbf{N}$ is the element shape function matrix, integrated over the free element boundary Γ_t. The element stiffness matrix $\mathbf{K^e}$ is obtained

by integrating the matrix $\mathbf{B}^{\mathbf{T}}\mathbf{DB}$ over the element's surface Ω. Matrix $\mathbf{B}$ is defined as $\mathbf{LN}$ with $\mathbf{L}$ the derivative matrix specified below. $\mathbf{D}$ is the elastic constitutive matrix for plane stress, containing the element-wise material parameters,

$$\mathbf{L} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix} \quad \text{and} \quad \mathbf{D} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix}. \tag{32}$$

3.1.3. Mesh Discretization

The beam model is discretized by a succession of structured nested quadrilateral meshes. We consider two different types of mesh discretization: h-refinement and p-refinement. MLMC and MLQMC will be applied to both refinement schemes. These combinations result in h-ML(Q)MC and p-ML(Q)MC. For h-ML(Q)MC, the coarsest mesh is chosen such that the beam is discretized by at least four elements over the height and twenty elements over the length. Similarly, the discretization for p-ML(Q)MC consists of four elements over the height and twenty elements over the length. This discretization remains the same for all levels. Table 1 lists the number of elements (Nel), degrees of freedom (DOF), the element order per level (Order), and the number of quadrature points per element (Nquad) for h-ML(Q)MC and p-ML(Q)MC. The mesh hierarchies are shown in Figures 1 and 3.

3.1.4. Modeling the Spatial Variability

We select a correlation length $\lambda = 0.3$, a standard deviation $\sigma = 1.0$ and a smoothness factor $\nu = 0.6$ for the covariance kernel in Equation (21) and generation of the Gaussian random field according to Equation (22). We will perform simulations with two different stochastic dimensions. A high stochastic dimension $s = 586$ and a moderate stochastic dimension $s = 78$. The value $s = 586$ accounts for 98% of the variance in the random field, while $s = 78$ only accounts for 92%, see Figure 12.

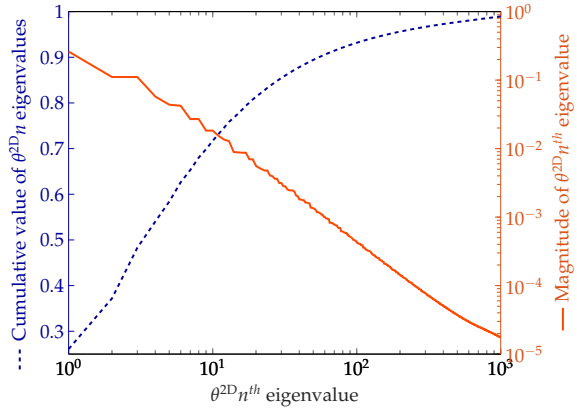

Figure 12. Magnitude of the eigenvalues and their cumulative sum for model problem 1.

Once the Gaussian field has been generated, a memoryless transformation is applied pointwise, i.e.,

$$g(y) = F^{-1}\left[\Phi(y)\right], \tag{33}$$

in order to obtain the gamma random field, see [38] for details.

Here, F denotes the marginal cumulative density function (CDF) of the target distribution, Φ denotes the marginal CDF of the standard normal distribution, and y is a discrete value of a realization of the random field. The memoryless transformation is depicted in Figure 13, with the solid line representing F, and the dashed line representing Φ. The characteristics of the Gamma distribution used to represent the uncertainty in concrete are as follows: a mean of 30 GPa and a standard deviation

of 7.74 GPa, see [39]. It is known that such a memoryless transformation distorts the underlying covariance function. In the scope of this paper, we will however not address the correction for this distortion. Instead, we will focus on the multilevel methodology. We refer the reader to [40–44] where iterative methods are described which correct this distortion.

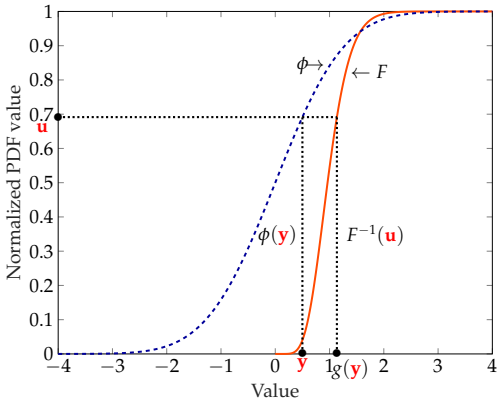

Figure 13. Memoryless transformation used to generate the gamma random field.

3.1.5. Quantity of Interest

The quantity of interest (QoI) we consider for all Monte Carlo simulations is the vertical displacement of the bottom layer of nodes of the beam, as indicated by arrows in Figure 14. The computed solution on each level is interpolated as if it was computed on the finest level. The MLQMC algorithm automatically chooses the node which has the biggest variance as the determining node for the computation of the required number of samples. By doing so, it is assured that the variance constraint is satisfied for all the other nodes.

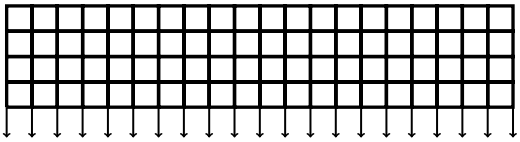

Figure 14. The vertical displacements, indicated with arrows, of the bottom nodes as the QoI for model problem 1.

3.2. Numerical Results for Model Problem 1

3.2.1. Rates

Figure 15 shows the value of $\mathbb{E}[P_\ell]$, $\mathbb{E}[\Delta P_\ell]$, $\mathbb{V}[P_\ell]$ and $\mathbb{V}[\Delta P_\ell]$ as a function of ℓ for a user requested tolerance on the RMSE of 2×10^{-5} for a moderate and a high stochastic dimension. Note that $\mathbb{E}[P_\ell]$ and $\mathbb{V}[P_\ell]$ remain constant across the levels, while $\mathbb{E}[\Delta P_\ell]$ and $\mathbb{V}[\Delta P_\ell]$ decrease for increasing ℓ. However, we observe for P-ML(Q)MC for the moderate stochastic dimension, a slight increase of $\mathbb{E}[\Delta P_L]$ with respect to $\mathbb{E}[\Delta P_{L-1}]$. However, this does not impact the results seeing that the overall descend of $\mathbb{E}[\Delta P_\ell]$ is guaranteed, i.e., $\mathbb{E}[\Delta P_0] > \mathbb{E}[\Delta P_L]$. The rates α and β are included in the figures for h-ML(Q)MC and p-ML(Q)MC. These rates respectively represent the slopes of $\mathbb{E}[\Delta P_\ell]$ and $\mathbb{V}[\Delta P_\ell]$. Figure 16 shows the cost of one solve per level, expressed as the runtime in seconds on a $\log_2$ scale. It is logical that per increasing level, the cost increases. This is because the stiffness matrix of the problem becomes larger due to a higher number of finite elements used to discretize the problem for h-ML(Q)MC. For p-ML(Q)MC this is due to a higher number of nodes. We observe that the cost per increasing level for p-ML(Q)MC grows slower than for h-ML(Q)MC. Combining this with the fact that the values of $\mathbb{V}[\Delta P_\ell]$ for p-ML(Q)MC are smaller than for h-ML(Q)MC, i.e., less samples are to be taken, we can expect that p-ML(Q)MC will outperform h-ML(Q)MC.

We can now evaluate in which regime, according to Theorem 1, h-ML(Q)MC and p-ML(Q)MC operate. We find for the moderate stochastic dimension that $\beta > \gamma$ for both h- and p-ML(Q)MC. We thus expect the h-MLMC and p-MLMC cost to be proportional to ϵ^{-2}. For the high stochastic dimension, we find that $\beta > \gamma$ for p-MLMC and $\beta < \gamma$ for h-MLMC. We respectively expect a cost proportional to ϵ^{-2} and $\epsilon^{-2.05}$. The cost for p-MLQMC and h-MLQMC cannot be evaluated from this figure, because no rate δ is available. The rate δ appears as an exponent in the integration error bound for QMC, see [28], and is not shown here.

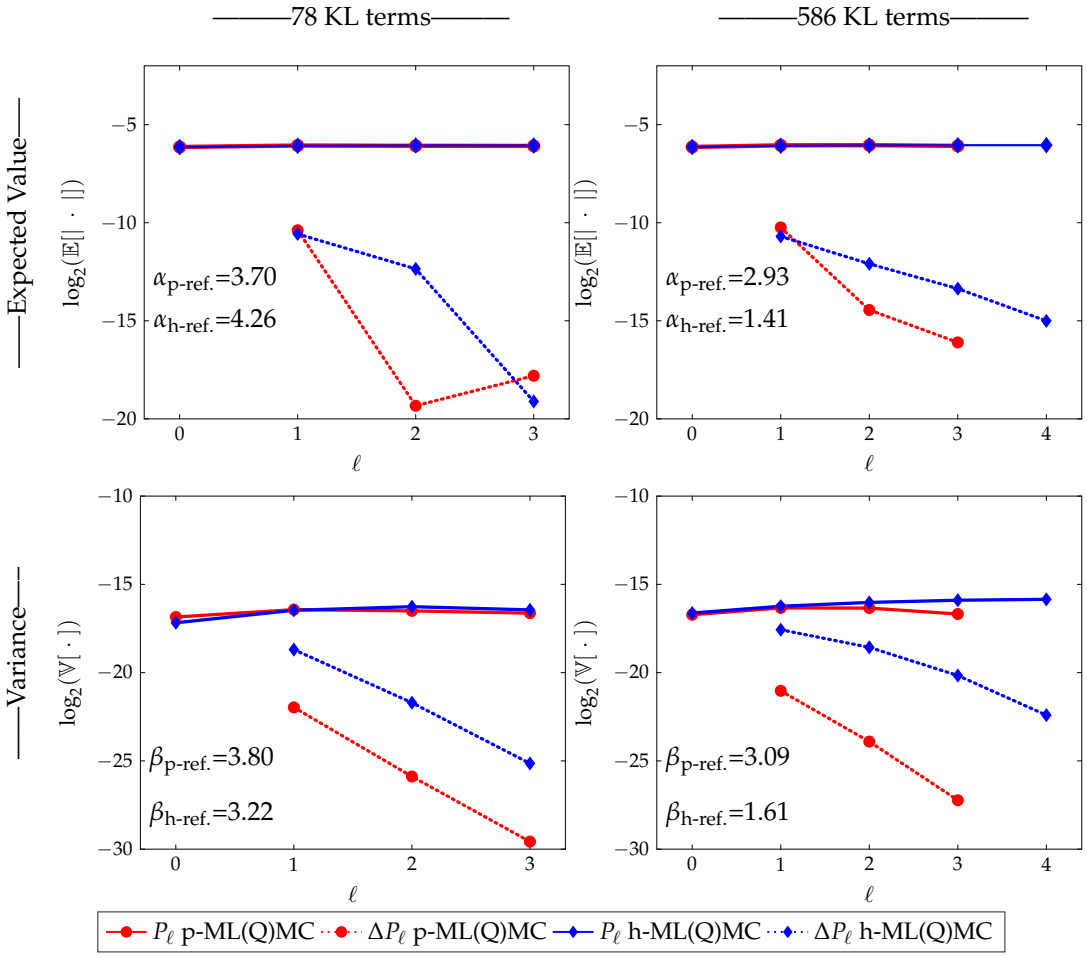

Figure 15. Variances and expected values for model problem 1.

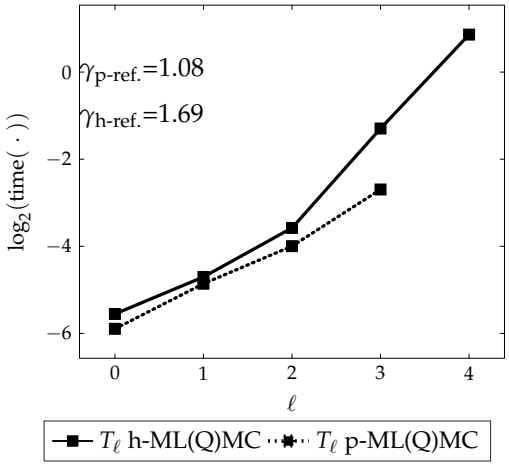

Figure 16. Times for one solve per level for model problem 1.

3.2.2. Number of Samples

Figure 17 shows the total number of samples over the different levels for a user requested tolerance on the RMSE of 2×10^{-5} for a moderate and a high stochastic dimension, for the different methods. The number of samples is decreasing as the level ℓ increases. Samples on lower levels are computationally less expensive. It is therefore advantageous to have a high number of samples on lower levels. Also, it can be seen that the sample sizes for MLQMC are lower than for MLMC. This is because the MLQMC sample points are chosen deterministically in an optimal way. This will result in a lower total computation time for MLQMC. The number of samples for p-ML(Q)MC is lower than for its respective h-ML(Q)MC counterpart. The origin of this effect is to be found in a lower value of $\mathbb{V}[\Delta P_\ell]$. We also see that for the high stochastic dimension, h-ML(Q)MC needs a total of 5 levels to satisfy its bias condition. This is to be compared with p-ML(Q)MC, which only needs a total of 4 levels, regardless of the considered stochastic dimension. Also observe that the samples for h-MC, p-MC, h-QMC and p-QMC are located on the finest level.

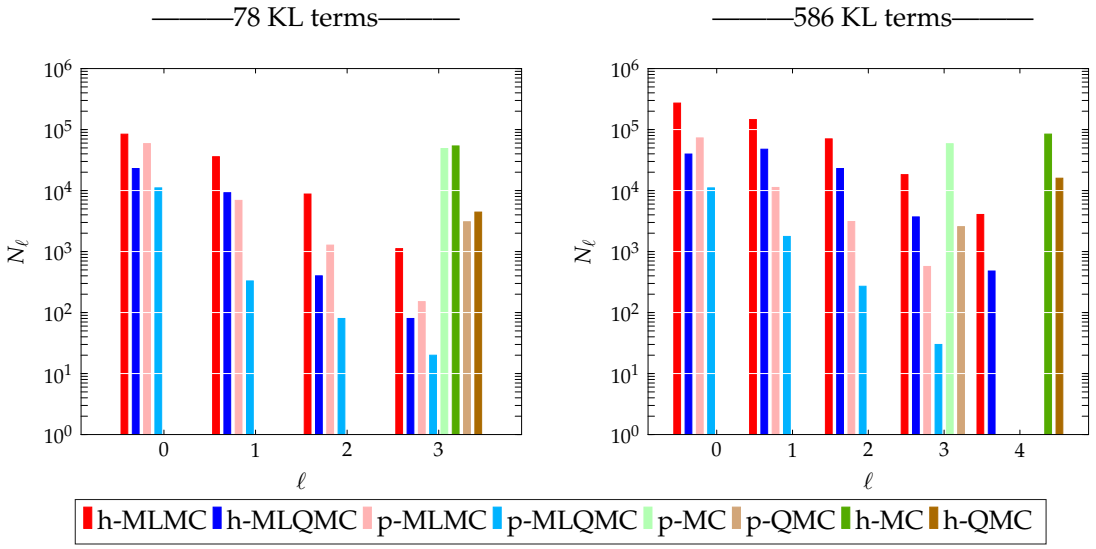

Figure 17. Number of samples for the different methods for model problem 1.

3.2.3. Uncertainty Propagation in the Solution

Figure 18 shows the uncertainty propagation towards the solution, i.e., the uncertainty on the QoI for the moderate and high stochastic dimension. Darker shades of blue indicate a higher probability for the displacement. The orange full line represents the mean, together with the 1-σ bounds in dashed orange.

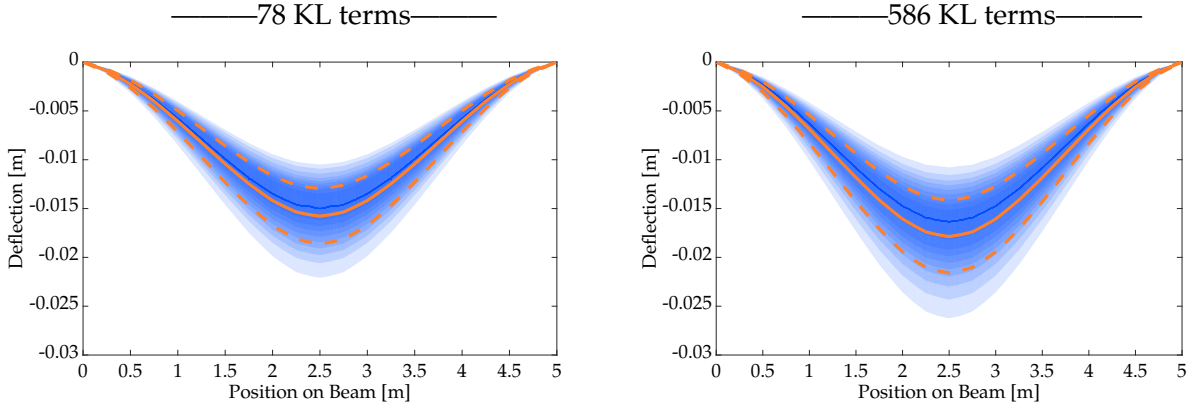

Figure 18. Uncertainty propagation towards the QoI's for model problem 1.

3.2.4. Runtime

We plot the runtimes of the different methods in Figure 19. We compare the p-MLQMC method against p-MLMC and h-ML(Q)MC. As a reference we also plot the times for h-MC, p-MC, h-QMC and p-QMC. The p-MLQMC method consistently outperforms all the other methods. In case of a moderate stochastic dimension we observe a gain of a factor 20 with respect to h-MLMC. For a high stochastic dimension, we observe a gain of a factor 100 with respect to h-MLMC. We also observe that all MLMC simulations exhibit a cost proportional to ϵ^{-2}, as expected. The MLQMC simulations on the other hand, exhibit a cost in the range of ϵ^{-1} to ϵ^{-2}, depending on the stochastic dimension. Interestingly, we note that for a moderate stochastic dimension, it remains advantageous to use h-MLQMC. The total runtime for h-MLQMC is indeed lower than for p-MLMC. This advantage, however, disappears for the high stochastic dimension. Here, all p-ML(Q)MC and p-MC simulations outperform the h-ML(Q)MC and h-MC simulations. p-MC even outperforms all h-ML(Q)MC simulations. This is due to the extra levels for the h-ML(Q)MC and h-MC simulations which are needed in order to satisfy their bias constraint.

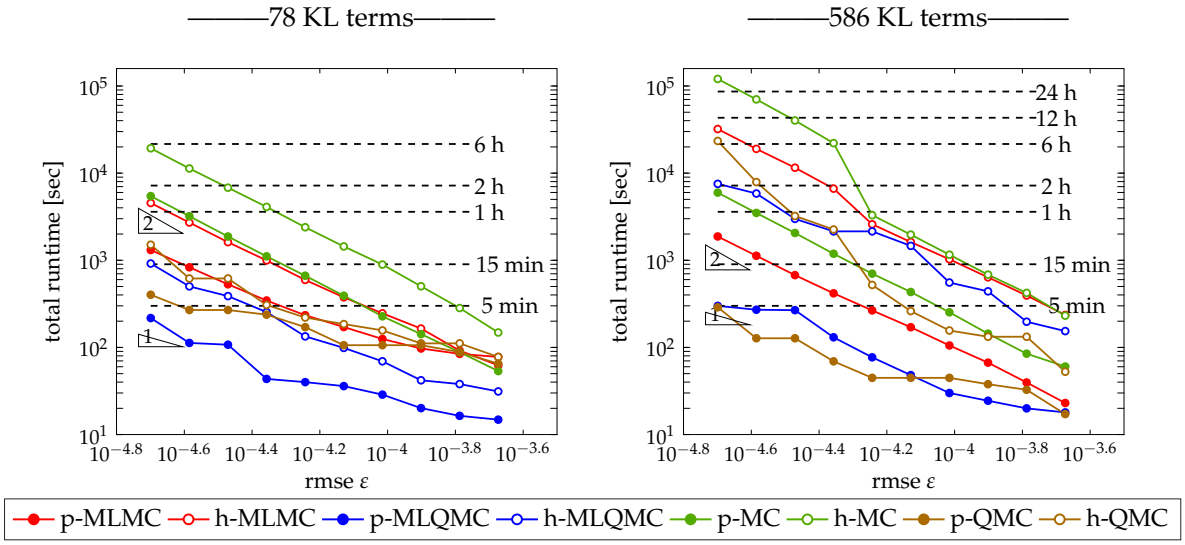

Figure 19. Runtimes for the different method in function of requested user tolerance for model problem 1.

3.2.5. Level Adaptivity

In order to better illustrate the level adaptivity of Algorithm 1, we have rerun Algorithm 2 to generate more meshes that can be used in our p-MLQMC method. The details of the hierarchy can be found in Table 3. We will simulate model problem 1 for finer tolerances. In order to more easily show that an extra level is added, we have set $\sigma = 1.5$ in Equation (21). This has an impact on the bias.

Table 3. Number of elements, degrees of freedom, element order and number of quadrature points for model problem 1 for additional tolerances.

	p-ML(Q)MC			
Level	**Nel**	**DOF**	**Order**	**Nquad**
0	80	210	1	25
1	80	738	2	81
2	80	1586	3	121
3	80	2754	4	225
4	80	4242	5	289
5	80	6050	6	441
6	80	8178	7	625
7	80	10,626	8	729

In Figure 20, we show the runtime and the sample sizes for finer considered tolerances than in Figure 19, for a moderate stochastic dimension. We observe a cost in the range of ϵ^{-1} to ϵ^{-2}.

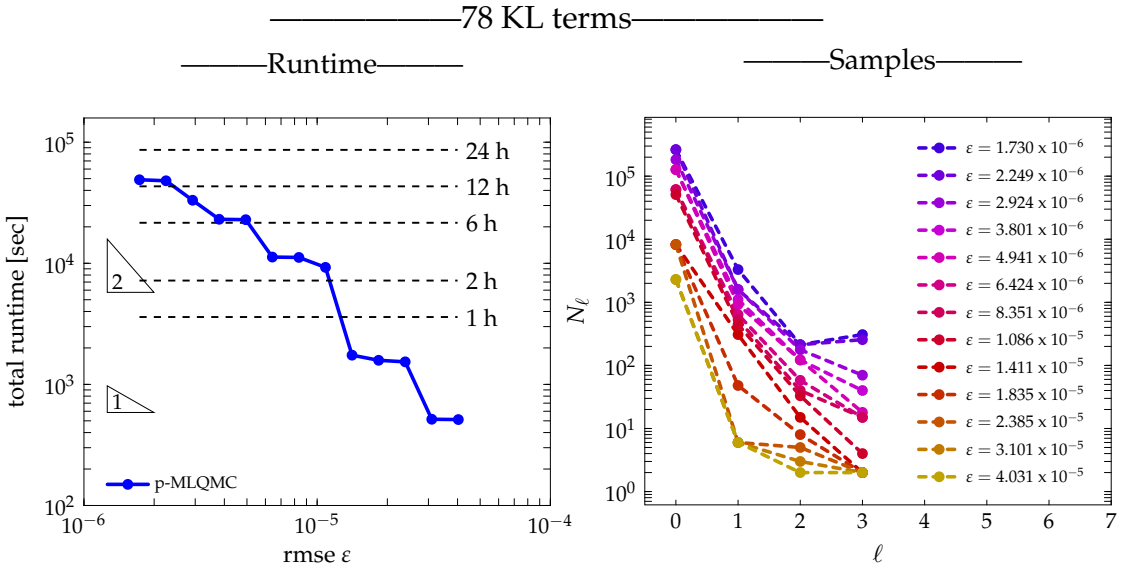

Figure 20. Runtime and sample sizes for a moderate stochastic dimension for model problem 1.

In Figure 21, we show the runtime and the sample sizes for finer considered tolerances than in Figure 19, for a high stochastic dimension. Here, we observe a cost that is closer to ϵ^{-2} than to ϵ^{-1}. Two extra levels are adaptively added in order to satisfy the bias constraint.

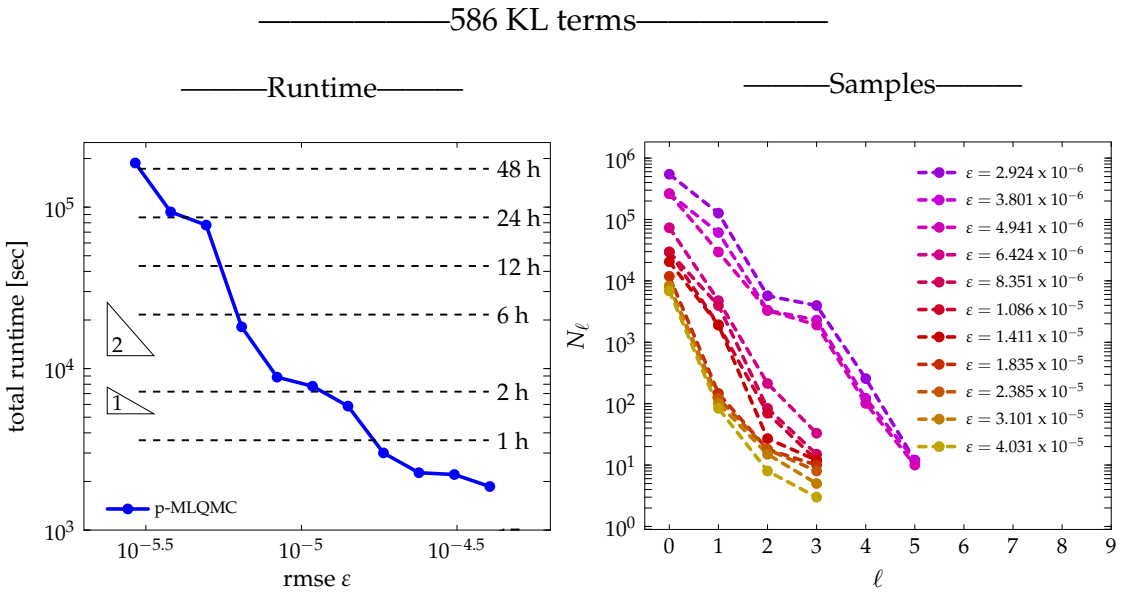

Figure 21. Runtime and sample sizes for a high stochastic dimension for model problem 1.

3.3. Model Problem 2

3.3.1. Description

Model problem 2 is a slope stability problem where the soil's cohesion has a spatially varying uncertainty [45]. In a slope stability problem, a sufficient factor of safety is required against sliding failure of the soil. However, this is complicated by a high level of heterogeneity and corresponding uncertainty of the soil's strength parameters [45]. The spatial dimensions of the slope are: a length of

20 m, a height of 14 m and a slope angle of 30°. The material characteristics are: a Young's modulus of 30 MPa, a Poisson ratio of 0.25, a density of 1330 kg/m^3 and a friction angle of 20°. Plane strain is considered for this problem, i.e., no strain component in its depth.

3.3.2. Finite Element Method

Because of the nonlinear stress-strain relation in the plastic domain, a Newton–Raphson iterative solver is used. The plastic region is governed by the Drucker–Prager yield criterion with a small amount of isotropic linear hardening for numerical stability reasons. An incremental load approach is used starting with a force of 0 N until the downward force resulting from the slope's weight is reached. The methods used to solve the slope stability problem are based on ([46], Chapter 2 Section 4, and Chapter 7 Sections 3 and 4). For this case, the system equation takes the following form:

$$\mathbf{K}\Delta\underline{\mathbf{u}} = \underline{\mathbf{r}}, \tag{34}$$

where $\Delta\underline{\mathbf{u}}$ stands for the displacement increment. The vector $\underline{\mathbf{r}}$ is the residual,

$$\underline{\mathbf{r}} = \underline{\mathbf{f}} + \Delta\underline{\mathbf{f}} - \underline{\mathbf{q}}, \tag{35}$$

where $\underline{\mathbf{f}}$ stands for the sum of the external force increments applied in the previous steps, $\Delta\underline{\mathbf{f}}$ for the applied load increment of the current step and $\underline{\mathbf{q}}$ for the internal force resulting from the stresses

$$\underline{\mathbf{q}} = \int_\Omega \mathbf{B}^T \boldsymbol{\sigma} d\Omega. \tag{36}$$

First, the displacement increment of all the nodes is computed according to Equation (34), with an initial system stiffness matrix $\mathbf{K}$ resulting from the assembly of the element stiffness matrix $\mathbf{K^e}$, computed by means of a Gauss quadrature, i.e.,

$$\mathbf{K^e} = \int_\Omega \mathbf{B}^T \mathbf{D}^{\mathrm{ep}} \mathbf{B} d\Omega. \tag{37}$$

Here $\mathbf{D}^{\mathrm{ep}}$ denotes the elastoplastic constitutive matrix. The initial state of $\mathbf{D}^{\mathrm{ep}}$ is defined as

$$\mathbf{D}^{\mathrm{ep}} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & \frac{\nu}{(1-\nu)} & 0 \\ \frac{\nu}{(1-\nu)} & 1 & 0 \\ 0 & 0 & \frac{(1-2\nu)}{2(1-\nu)} \end{bmatrix}. \tag{38}$$

Secondly, the strain increment $\Delta\varepsilon$ is computed as

$$\Delta\varepsilon = \mathbf{B}\Delta\underline{\mathbf{u}}. \tag{39}$$

Finally, the nonlinear stress–strain relationship,

$$d\boldsymbol{\sigma} = \mathbf{D}^{\mathrm{ep}} d\varepsilon, \tag{40}$$

is integrated by means of a backward Euler method. The backward Euler method essentially acts as an elastic predictor-plastic corrector; an initial stress state that is purely elastic is computed and then projected in the direction of the yield surface to obtain the plastic stress state. Due to the implicit nature of the integrated stress-strain relation, this equation must be supplemented with the integrated form of the hardening rule and the yield condition. This system of nonlinear equations is then solved with an iterative Newton–Raphson method. Afterwards, the consistent tangent stiffness matrix is computed, see [47]. This matrix is then used to compute the updated element stiffness matrix, Equation (37),

resulting in an updated system stiffness matrix **K**. The inner iteration step of solving the stress-strain relation and the updated system stiffness matrix is repeated for each outer iteration step which solves Equation (34). The outer step consists in balancing the internal forces with the external ones as to satisfy the residual, which in our case equals 10^{-4} multiplied with the load increment. The procedure used is incremental–iterative, relying on the iterative Newton–Raphson method. This process is repeated for each load increment.

3.3.3. Mesh Discretization

The problem is discretized by means of a hierarchy of unstructured triangular nested meshes. Both h-ML(Q)MC and p-ML(Q)MC will be applied. Table 2 lists the number of elements (Nel), degrees of freedom (DOF), element order per level (Order), and the number of quadrature points per element (Nquad) for h-ML(Q)MC and p-ML(Q)MC. The mesh hierarchies are shown in Figures 2 and 4.

3.3.4. Modeling the Spatial Variability

For model problem 2, we select a correlation length $\lambda = 1.0$, a standard deviation $\sigma = 1.0$ and a smoothness factor $\nu = 0.6$ in Equation (21). Here, we also consider two different stochastic dimensions for the Gaussian random field: a value $s = 10$, accounting for 92% of the variance, and a value $s = 210$, accounting for 99% of the variance, Figure 22.

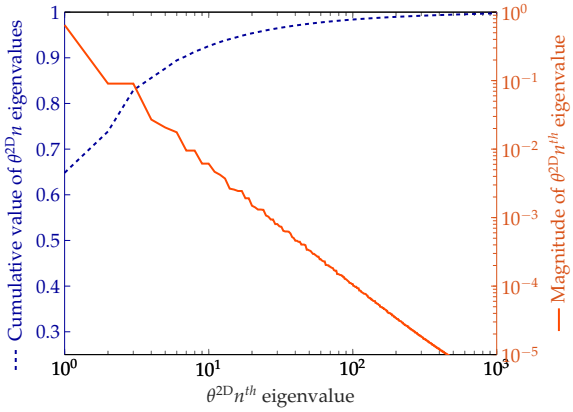

Figure 22. Magnitude of the eigenvalues and their cumulative sum for model problem 1.

The obtained Gaussian field is transformed into a lognormal random field by applying the exponential operator,

$$Z_{\text{lognormal}}(\mathbf{x}, \omega) = \exp(Z(\mathbf{x}, \omega)). \tag{41}$$

By using this operation, the underlying covariance is not distorted. The characteristics of the lognormal distribution used to represent the uncertainty of the soil's cohesion are as follows: a mean of 6 kPa and a standard deviation of 300 Pa.

3.3.5. Quantity of Interest

We consider the vertical displacement of the upper left node of the model, see Figure 23, as a quantity of interest.

3.4. Numerical Results for Model Problem 2

3.4.1. Rates

As for model problem 1, we first present the different rates for model problem 2 for a user requested tolerance on the RMSE of 5×10^{-5}. These are shown in Figure 24. Here, we observe that the α rates for p-ML(Q)MC exhibit much larger values. The bias constraint will be fulfilled with less

levels for p-ML(Q)MC than for h-ML(Q)MC. On the other hand, the β rates for p-ML(Q)MC exhibit a smaller value than for h-ML(Q)MC. These results suggest that an approach where both p-refinement and h-refinement are combined would yield a good decrease of $\mathbb{E}[\Delta P_\ell]$ and $\mathbb{V}[\Delta P_\ell]$. Following this figure, we expect that all h-ML(Q)MC simulations will require more levels than their p-ML(Q)MC counterparts to satisfy the bias constraint. We have chosen to limit the amount of levels for the h-ML(Q)MC simulations to a total of five. This is done because of the considerable cost to compute solutions on finer meshes. This inadvertently means that a bias is still present on our h-ML(Q)MC results. Our p-ML(Q)MC solutions do not exhibit such a bias, i.e., the bias constraint is satisfied with a total of four levels. In Figure 25, we again observe that the cost increase, γ, per increasing level is smaller for p-ML(Q)MC than for h-ML(Q)MC.

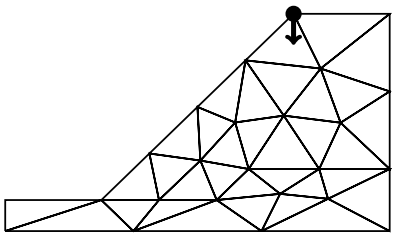

Figure 23. The vertical displacement, indicated with an arrow, of the upper left node as the QoI for model problem 2.

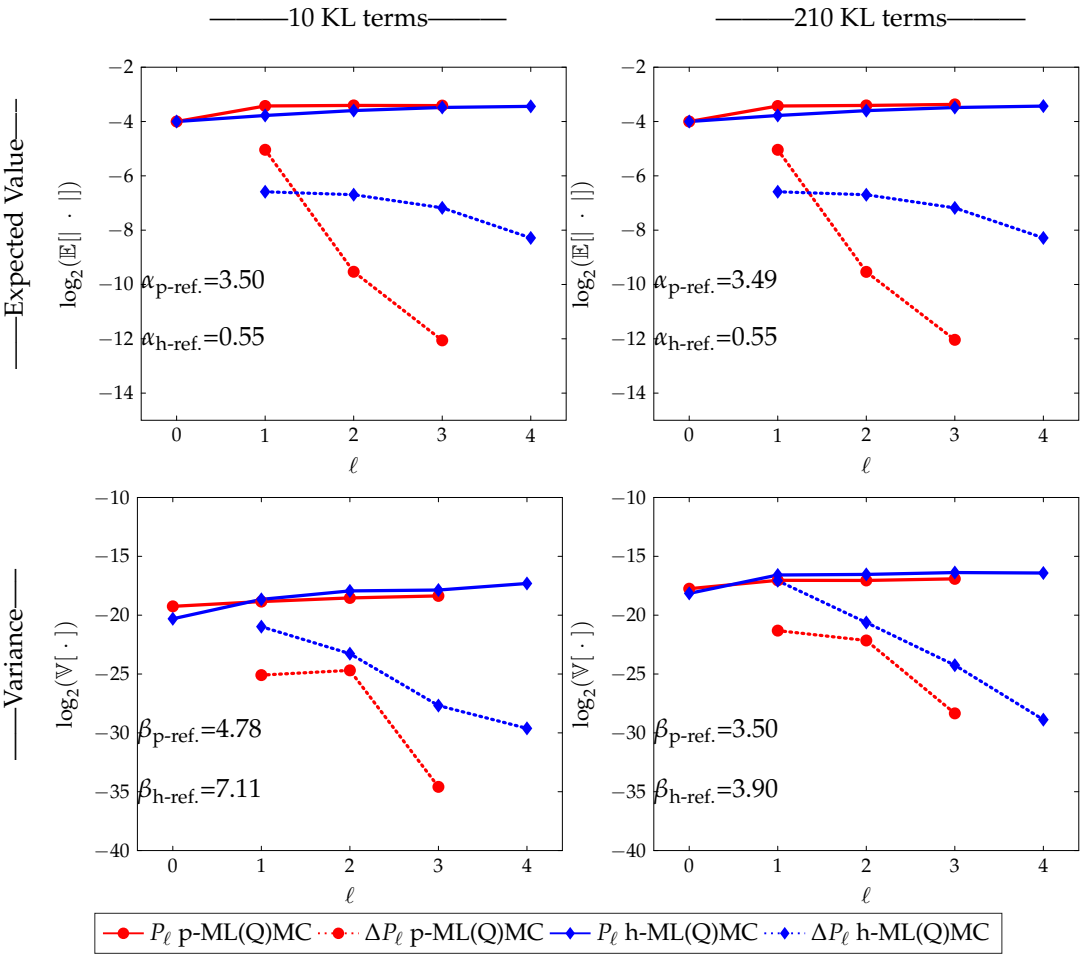

Figure 24. Variances and Expected values for model problem 2.

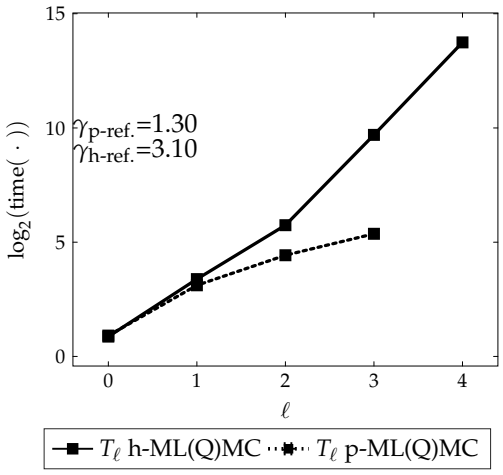

Figure 25. Times for one solve per level for model problem 2.

3.4.2. Number of Samples

Figure 26 shows the number of samples for a high and a moderate stochastic dimension for model problem 2 for a user requested tolerance on the RMSE of 5×10^{-5}. Similar conclusion can be drawn as for model problem 1. Here, our p-ML(Q)MC simulations only need four levels while h-ML(Q)MC requires at least five.

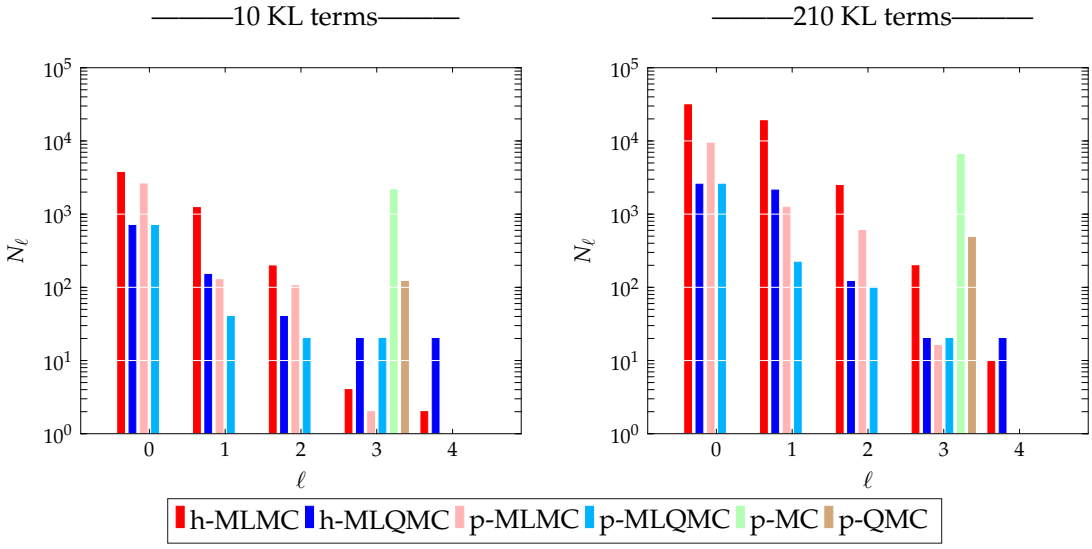

Figure 26. Number of samples for the different methods for model problem 2.

3.4.3. Uncertainty Propagation in the Solution

Figure 27 shows the uncertainty on the QoI. Here the vertical displacement of the QoI is shown in function of the applied force. The line style convention is the same as for model problem 1.

3.4.4. Runtime

We plot the runtimes for the different methods in Figure 28. We show the total runtime in seconds in function of the user requested tolerance. As already observed for model problem 1, p-MLQMC outperforms h-MLMC. For a moderate stochastic dimension we observe a speedup up to a factor 20. For the high stochastic dimension we observe a speedup up to a factor 40. Note however that for the h-ML(Q)MC simulations we have fixed the maximum level because of the prohibitively large computational cost. Because of this, we have chosen not to simulate this case with h-MC or h-QMC. Here, we clearly observe a cost proportional to ϵ^{-1} for p-MLQMC.

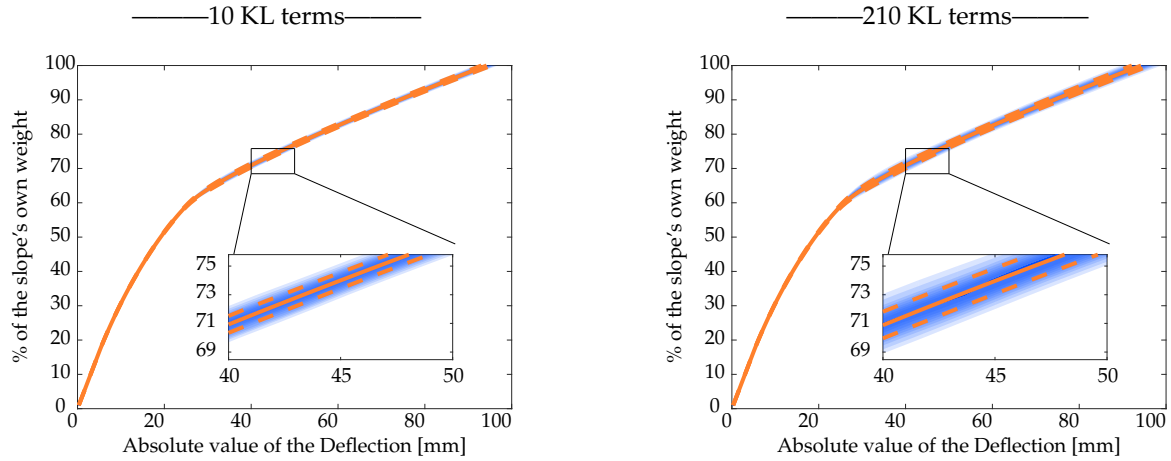

Figure 27. Uncertainty propagation towards the QoI's for model problem 2.

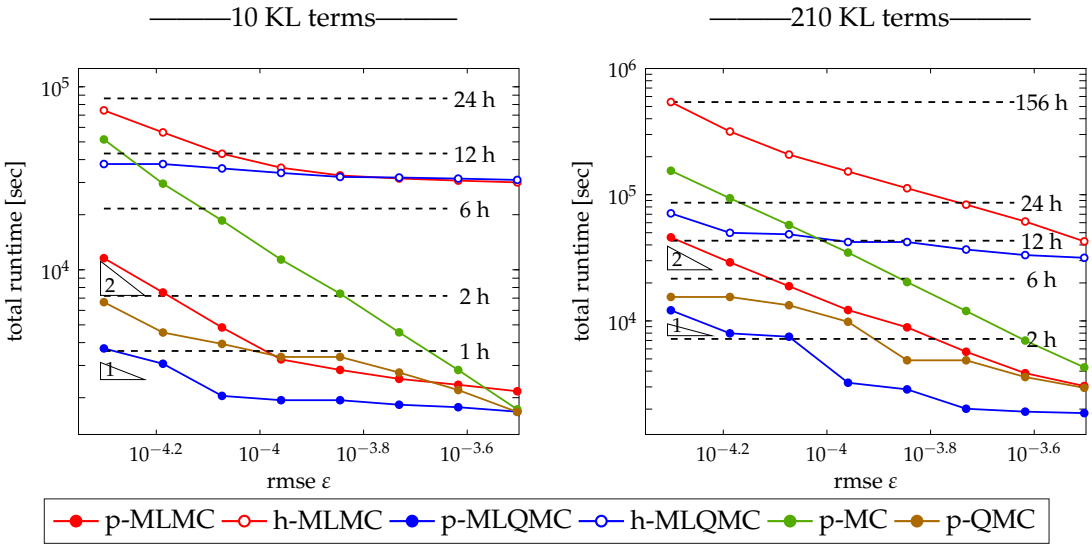

Figure 28. Runtimes for the different method in function of requested user tolerance for model problem 2.

3.4.5. Level Adaptivity

In order to better illustrate the level adaptivity of Algorithm 1, we have rerun Algorithm 2 to generate more meshes that can be used in our p-MLQMC method. The details of the hierarchy can be found in Table 4.

Table 4. Number of elements, degrees of freedom, element order and number of quadrature points for model problem 2 for additional tolerances.

	p-ML(Q)MC			
Level	**Nel**	**DOF**	**Order**	**Nquad**
0	33	48	1	7
1	33	160	2	13
2	33	338	3	19
3	33	582	4	25
4	33	892	5	28
5	33	1268	6	33
6	33	1720	7	37
7	33	2218	8	61
8	33	2792	9	73

In Figure 29, we show the runtime and the sample sizes for finer considered tolerances than in Figure 28, for a moderate stochastic dimension. We clearly observe a cost proportional to ϵ^{-1}.

In Figure 30, we show the runtime and the sample sizes for finer considered tolerances than in Figure 28, for a high stochastic dimension. Again, we observe a cost proportional to ϵ^{-1}. We also observe that additional levels are added adaptively in order to satisfy the bias constraint.

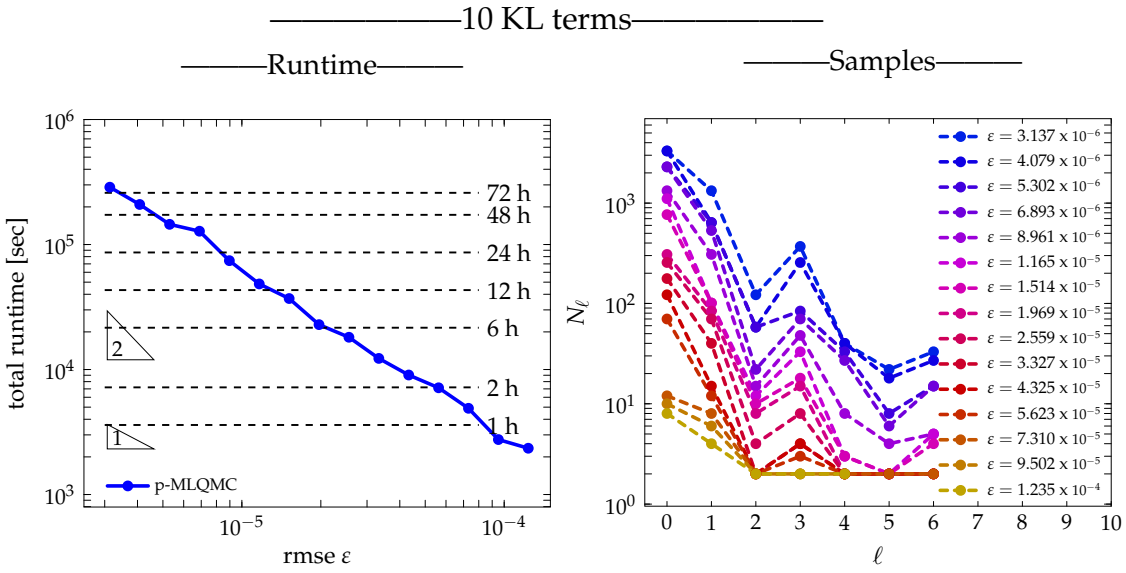

Figure 29. Runtime and sample sizes for a moderate stochastic dimension for model problem 2.

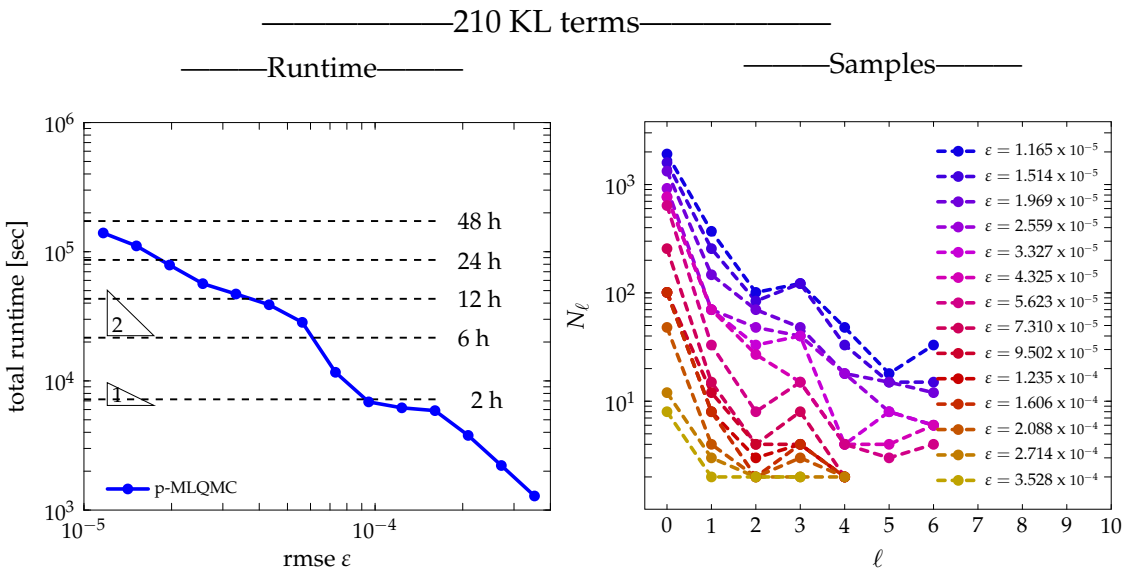

Figure 30. Runtime and sample sizes for a high stochastic dimension for model problem 2.

4. Conclusions

In this work we presented a novel multilevel algorithm, p-refined Multilevel Quasi-Monte Carlo. We benchmarked this method against existing multilevel algorithms for two engineering cases: an academic clamped beam bending problem, from the civil/mechanical engineering domain, and the slope stability problem, from the geotechnical engineering domain. We detailed how the random fields are generated for these two cases and how they were taken into account in the model. We also described two algorithms necessary for p-MLQMC, the sample algorithm itself and an algorithm which returns

the location of the points where the discrete values of the random field are to be computed. The second algorithm was developed for two dimensional problems but is extensible to three dimensional problems. We observed that the p-MLQMC method drastically reduces the computational cost with respect to a classical Multilevel Monte Carlo approach. We observed computational gains ranging from a factor 20 to 100. In turn, h-MLMC is 4 to 6 times faster than standard MC. In addition to this cost reduction, we also showed that the computational cost increase in function of a user requested tolerance ϵ, is proportional to ϵ^{-1}. For model problem 2, we have shown that the decay of $\mathbb{E}[\Delta P_\ell]$ for h-ML(Q)MC is much slower than for p-ML(Q)MC. The opposite holds for the decay of $\mathbb{V}[\Delta P_\ell]$. This behavior could be exploited in a Multi-Index (Quasi)-Monte Carlo setting in order to further reduce the total computational cost, see [11]. Higher order Quasi-Monte Carlo methods, which use higher order digital nets instead of lattice rules, could be used in order to further decrease the error on the root mean square, and so decrease the computational cost.

Author Contributions: Conceptualization, P.B., P.R. and S.V.; methodology, P.B., P.R., C.V.h. and S.F.; software, P.B., P.R., C.V.h. and S.F.; validation, P.B., P.R., G.L. and S.V.; formal analysis, P.B.; investigation, P.B. and P.R.; resources, P.R., C.V.h., S.F., G.L. and S.V.; data curation, P.B., P.R. and S.F.; writing–original draft preparation, P.B., P.R. and S.V.; writing–review and editing, P.B., P.R., C.V.h., S.F., G.L. and S.V.; visualization, P.B.; supervision, G.L. and S.V.; project administration, G.L. and S.V.; funding acquisition, S.V. All authors have read and agree to the published version of the manuscript.

Funding: The authors gratefully acknowledge the support from the Research Council of KU Leuven through project C16/17/008 "Efficient methods for large-scale PDE-constrained optimization in the presence of uncertainty and complex technological constraints".

Acknowledgments: The authors would also like to thank the Structural Mechanics Section of the KU Leuven and Jef Wambacq for supplying their StaBIL code and for providing support. The computational resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation - Flanders (FWO) and the Flemish Government – department EWI. The stochastic part of our simulations was performed with the Julia packages **MultilevelEstimators.jl** and **GaussianRandomFields.jl**.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kleiber, M.; Hien, T.D. *The Stochastic Finite Element Method Basic Perturbation Technique and Computer Implementation*; Wiley: Chichester, UK, 1992.
2. Ghanem, R.G.; Spanos, P.D. *Stochastic Finite Elements: A Spectral Approach*; Dover Publications: New York, NY, USA, 2003.
3. Babuška, I.; Nobile, F.; Tempone, R. A Stochastic Collocation Method for Elliptic Partial Differential Equations with Random Input Data. *SIAM J. Numer. Anal.* **2007**, *45*, 1005–1034. [CrossRef]
4. Fishman, G.S. *Monte Carlo: Concepts, Algorithms and Applications*; Springer: New York, NY, USA, 1996.
5. Caflisch, R.E. Monte Carlo and Quasi-Monte Carlo methods. *Acta Numer.* **1998**, *7*, 1–49, [CrossRef]
6. Niederreiter, H. *Monte Carlo and Quasi-Monte Carlo Methods*; Springer: Berlin, Germany, 2004.
7. Loh, W.L. On Latin hypercube sampling. *Ann. Stat.* **1996**, *24*, 2058–2080. [CrossRef]
8. Giles, M.B. Multilevel Monte Carlo Path Simulation. *Oper. Res.* **2008**, *56*, 607–617. [CrossRef]
9. Giles, M.B.; Waterhouse, B.J. Multilevel Quasi-Monte Carlo path simulation. *Radon Ser. Comput. Appl. Math.* **2009**, *8*, 1–18.
10. Robbe, P.; Nuyens, D.; Vandewalle, S. A Multi-Index Quasi-Monte Carlo Algorithm for Lognormal Diffusion Problems. *SIAM J. Sci. Comput.* **2017**, *39*, S851–S872. [CrossRef]
11. Robbe, P.; Nuyens, D.; Vandewalle, S. A Dimension-Adaptive Multi-Index Monte Carlo Method Applied to a Model of a Heat Exchanger. In *Monte Carlo and Quasi-Monte Carlo Methods*; Owen, A.B., Glynn, P.W., Eds.; Springer: Cham, Switzerland, 2018; pp. 429–445.
12. Ghanem, R. Hybrid stochastic finite elements and generalized Monte Carlo simulation. *J. Appl. Mech.* **1998**, *65*, 1004–1009. [CrossRef]
13. Acharjee, S.; Zabaras, N. A non-intrusive stochastic Galerkin approach for modeling uncertainty propagation in deformation processes. *Comput. Struct.* **2007**, *85*, 244–254. [CrossRef]

14. Blondeel, P.; Robbe, P.; Van hoorickx, C.; Lombaert, G.; Vandewalle, S. The Multilevel Monte Carlo method applied to structural engineering problems with uncertainty in the Young's modulus. In Proceedings of the 28th edition of the Biennial ISMA conference on Noise and Vibration Engineering (ISMA 2018), Leuven, Belgium, 17–19 September 2018; pp. 4899–4913.
15. Blondeel, P.; Robbe, P.; Van hoorickx, C.; Lombaert, G.; Vandewalle, S. Multilevel sampling with Monte Carlo and Quasi-Monte Carlo methods for uncertainty quantification in structural engineering. In Proceedings of the 13th International Conference on Applications of Statistics and Probability in Civil Engineering (ICASP13), Seoul, Korea, 26–30 May 2019. [CrossRef]
16. Motamed, M.; Appelö, D. A MultiOrder Discontinuous Galerkin Monte Carlo Method for Hyperbolic Problems with Stochastic Parameters. *SIAM J. Numer. Anal.* **2018**, *56*, 448–468. [CrossRef]
17. Giles, M.B. Multilevel Monte Carlo methods. *Acta Numer.* **2015**, *24*, 259–328. [CrossRef]
18. Dick, J.; Kuo, F.Y.; Sloan, I.H. High-dimensional integration: The Quasi-Monte Carlo way. *Acta Numer.* **2013**, *22*, 133–288. [CrossRef]
19. Hickernell, F.J.; Hong, H.S.; L'Ecuyer, P.; Lemieux, C. Extensible Lattice Sequences for Quasi-Monte Carlo Quadrature. *SIAM J. Sci. Comput.* **2000**, *22*, 1117–1138. [CrossRef]
20. Graham, I.G.; Kuo, F.Y.; Nichols, J.A.; Scheichl, R.; Schwab, C.; Sloan, I.H. Quasi-Monte Carlo finite element methods for elliptic PDEs with lognormal random coefficients. *Numer. Math.* **2015**, *131*, 329–368. [CrossRef]
21. Robbe, P. Multilevel Uncertainty Quantification Methods for Robust Design of Industrial Applications. Ph.D. Thesis, KU Leuven, Leuven, Belgium, 2019.
22. Kuo, F.Y.; Scheichl, R.; Schwab, C.; Sloan, I.H.; Ullmann, E. Multilevel Quasi-Monte Carlo methods for lognormal diffusion problems. *Math. Comput.* **2017**, *86*, 2827–2860. [CrossRef]
23. Sobol', I. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Comput. Math. Math. Phys.* **1967**, *7*, 86–112. [CrossRef]
24. Kuo, F. Lattice Rule Generating Vectors. 2007. Available online: https://web.maths.unsw.edu.au/~fkuo/lattice/index.html (accessed on 12 April 2019).
25. Ciarlet, P.G. *The Finite Element Method for Elliptic Problems*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2002. [CrossRef]
26. Geuzaine, C.; Remacle, J.F. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *Int. J. Numer. Meth. Eng.* **2009**, *79*, 1309–1331. [CrossRef]
27. *MATLAB*, version 9.2.0 (R2017a); The MathWorks Inc.: Natick, MA, USA, 2017.
28. Graham, I.; Kuo, F.; Nuyens, D.; Scheichl, R.; Sloan, I. Quasi-Monte Carlo methods for elliptic PDEs with random coefficients and applications. *J. Comput. Phys.* **2011**, *230*, 3668–3694. [CrossRef]
29. Teckentrup, A.L. Multilevel Monte Carlo Methods and Uncertainty Quantification. Ph.D. Thesis, University of Bath, Bath, UK, 2013.
30. Sloan, I.H.; Woźniakowski, H. When Are Quasi-Monte Carlo Algorithms Efficient for High Dimensional Integrals? *J. Complex.* **1998**, *14*, 1–33. [CrossRef]
31. Kuo, F.Y.; Nuyens, D. Application of Quasi-Monte Carlo Methods to Elliptic PDEs with Random Diffusion Coefficients: A Survey of Analysis and Implementation. *Found. Comput. Math.* **2016**, *16*, 1631–1696. [CrossRef]
32. Loève, M. *Probability Theory*; Springer: New York, NY, USA, 1977.
33. Atkinson, K.; Han, W. Numerical Solution of Fredholm Integral Equations of the Second Kind. In *Theoretical Numerical Analysis: A Functional Analysis Framework*; Springer: New York, NY, USA, 2009; pp. 473–549. [CrossRef]
34. Li, J.; Chen, J. *Stochastic Dynamics of Structures*; John Wiley & Sons: New York, NY, USA, 2010.
35. Brenner, C.E.; Bucher, C. A contribution to the SFE-based reliability assessment of nonlinear structures under dynamic loading. *Probab. Eng. Mech.* **1995**, *10*, 265–273. [CrossRef]
36. Teckentrup, A.L.; Scheichl, R.; Giles, M.B.; Ullmann, E. Further analysis of multilevel Monte Carlo methods for elliptic PDEs with random coefficients. *Numer. Math.* **2013**, *125*, 569–600. [CrossRef]
37. Gittelson, C.J.; Könnö, J.; Schwab, C.; Stenberg, R. The multi-level Monte Carlo finite element method for a stochastic Brinkman Problem. *Numer. Math.* **2013**, *125*, 347–386. [CrossRef]
38. Grigoriu, M. Simulation of Stationary Non-Gaussian Translation Processes. *J. Eng. Mech. (ASCE)* **1998**, *124*, 121–126. [CrossRef]
39. Simoen, E.; Moaveni, B.; Conte, J.P.; Lombaert, G. Uncertainty Quantification in the Assessment of Progressive Damage in a 7-Story Full-Scale Building Slice. *J. Eng. Mech.* **2013**, *139*, 1818–1830. [CrossRef]

40. Sakamoto, S.; Ghanem, R. Simulation of multi-dimensional non-gaussian non-stationary random fields. *Probab. Eng. Mech.* **2002**, *17*, 167–176. [CrossRef]
41. Phoon, K.; Huang, H.; Quek, S. Simulation of strongly non-Gaussian processes using Karhunen–Loeve expansion. *Probab. Eng. Mech.* **2005**, *20*, 188–198. [CrossRef]
42. Phoon, K.; Huang, S.; Quek, S. Simulation of second-order processes using Karhunen–Loeve expansion. *Comput. Struct.* **2002**, *80*, 1049–1060. [CrossRef]
43. Kim, H.; Shields, M.D. Modeling strongly non-Gaussian non-stationary stochastic processes using the Iterative Translation Approximation Method and Karhunen–Loève expansion. *Comput. Struct.* **2015**, *161*, 31–42. [CrossRef]
44. Shields, M.; Deodatis, G.; Bocchini, P. A simple and efficient methodology to approximate a general non-Gaussian stationary stochastic process by a translation process. *Probab. Eng. Mech.* **2011**, *26*, 511–519. [CrossRef]
45. Whenham, V.; De Vos, M.; Legrand, C.; Charlier, R.; Maertens, J.; Verbrugge, J.C. Influence of Soil Suction on Trench Stability. In *Experimental Unsaturated Soil Mechanics*; Schanz, T., Ed.; Springer: Berlin/Heidelberg, Germany, 2007; pp. 495–501.
46. de Borst, R.; Crisfield, M.A.; Remmers, J.J.C. *Non Linear Finite Element Analysis of Solids and Structures*; Wiley: London, UK, 2012.
47. Pérez-Foguet, A.; Rodríguez-Ferran, A.; Huerta, A. Consistent tangent matrices for substepping schemes. *Comput. Methods Appl. Mech. Eng.* **2001**, *190*, 4627–4647. [CrossRef]