Uncertainty Quantification and Quasi-Monte Carlo Exercise 8
Sommersemester 2025
Return your written solutions either in person or by email
to vesa.kaarnioja@fu-berlin.de by Tuesday 24 June 2025, 10:15 am
Please make sure to return your source code for all programming tasks

Please note that there are a total of 4 tasks!

1. (Finite element error) Let $D = (0,1)^2$ and consider solving the Poisson problem

$$\begin{cases} -\Delta u(\boldsymbol{x}) = x_1, & \boldsymbol{x} = (x_1, x_2) \in D, \\ u|_{\partial D} = 0 \end{cases}$$

using the finite element method. We showed during the lecture that the convergence rate for a piecewise linear finite element approximation $u_h \in V_h$ satisfies

$$\|u - u_h\|_{L^2(D)} \leq Ch^2,$$

where $C > 0$ is independent of the mesh width h (as long as the FE mesh is regular and uniform). Let us try verifying this numerically.

Use the `fem.py` script on the course webpage to generate FE matrices as well as other FEM objects corresponding to different FE discretization levels of the computational domain $D = (0,1)^2$. First compute the finite element solutions u_h for $h = 2^{-1}, 2^{-2}, \ldots, 2^{-5}$ (corresponding to the arguments `level=1,2,3,4,5` in the script). Then, using the solution corresponding to the densest mesh width $h' = 2^{-5}$ as the reference, approximate the FE error by computing the values

$$\|u_{h'} - u_h\|_{L^2(D)} \quad \text{for } h = 2^{-1}, \ldots, 2^{-4}.$$

To achieve this, you can *interpolate* the coarser FE solutions onto the densest FE mesh corresponding to the reference solution with mesh width h'.

In tasks 2–3, we continue with the study of a parametric elliptic PDE problem. Let $D = (0,1)^2$ and $f(\boldsymbol{x}) = x_1$ for $\boldsymbol{x} = (x_1, x_2) \in D$. For all $\boldsymbol{y} \in [-1/2, 1/2]^{\mathbb{N}}$, let $u(\cdot, \boldsymbol{y}) \in H_0^1(D)$ be such that

$$\int_D a(\boldsymbol{x}, \boldsymbol{y}) \nabla u(\boldsymbol{x}, \boldsymbol{y}) \cdot \nabla v(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = \int_D f(\boldsymbol{x}) v(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \quad \text{for all } v \in H_0^1(D), \tag{1}$$

with the diffusion coefficient

$$a(\boldsymbol{x}, \boldsymbol{y}) = 2 + \sum_{j=1}^{\infty} y_j \psi_j(\boldsymbol{x}), \quad \boldsymbol{x} \in D, \ \boldsymbol{y} = (y_j)_{j \geq 1} \in [-1/2, 1/2]^{\mathbb{N}}, \tag{2}$$

where we define $\psi_j(\boldsymbol{x}) := j^{-2} \sin(j\pi x_1) \sin(j\pi x_2)$ for $\boldsymbol{x} = (x_1, x_2) \in D$. Moreover, we define the dimensionally-truncated solution by setting $u_s(\cdot, (y_1, \ldots, y_s)) := u(\cdot, (y_1, \ldots, y_s, 0, 0, \ldots))$ for $y_j \in [-1/2, 1/2]$, $1 \leq j \leq s$.

2. (Dimension truncation error) The dimension truncation error rate for the parametric PDE problem specified above satisfies

$$\left| \int_{[-1/2,1/2]^{\mathbb{N}}} G(u(\cdot,\boldsymbol{y}) - u_s(\cdot,\boldsymbol{y}))\,\mathrm{d}\boldsymbol{y} \right| \leq C' s^{-3+\varepsilon}, \tag{3}$$

where the constant $C' > 0$ is independent of s with arbitrary $\varepsilon > 0$ and $G \in H^{-1}(D)$. Let us try verifying this numerically.

Use the `fem.py` script on the course website to generate FE matrices as well as other FEM objects corresponding to a fixed FE discretization of the computational domain $D = (0,1)^2$ (you can use, e.g., mesh width $h = 2^{-5}$). Download also the file `offtheshelf2048.txt` from the course webpage. The file contains a 2048-dimensional generating vector $\boldsymbol{z} \in \mathbb{N}^{2048}$ which you can truncate to any dimension $s \in \{1,\ldots,2048\}$ by simply extracting the first s elements and using this as your s-dimensional generating vector. Let us denote the generating vector obtained in this way by $\boldsymbol{z}^{(s)} \in \mathbb{N}^s$.

Let us approximate the PDE solutions appearing in (3) using the finite element method. As the linear quantity of interest $G \in H^{-1}(D)$, take

$$G(v) := \int_D v(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}, \quad v \in H_0^1(D). \tag{4}$$

Note that if $v_h = \sum_{i=1}^N c_i \phi_i(\boldsymbol{x}) \in V_h$ is a finite element function, we can write

$$G(v_h) = \mathbf{1}^{\mathrm{T}} M \boldsymbol{c},$$

where M is the mass matrix, $\boldsymbol{c} := [c_1,\ldots,c_N]^{\mathrm{T}}$ are the finite element expansion coefficients, and $\mathbf{1} = [1,1,\ldots,1]^{\mathrm{T}} \in \mathbb{R}^N$.

Your task is to first compute the QMC approximations

$$I_s = \frac{1}{n}\sum_{i=1}^n G(u_{s,h}(\cdot,\boldsymbol{t}_i - \tfrac{\mathbf{1}}{\mathbf{2}})) \approx \int_{[-1/2,1/2]^s} G(u_{s,h}(\cdot,\boldsymbol{y}))\,\mathrm{d}\boldsymbol{y}, \quad \boldsymbol{t}_i := \mathrm{mod}\big(\tfrac{i\boldsymbol{z}^{(s)}}{n}, 1\big),$$

for $s = 2^k$, $k = 1,\ldots,11$, using $n = 2^{15}$ QMC cubature nodes. Then, using the 2048-dimensional solution as the reference, approximate the quantity appearing in (3) by computing the values

$$|I_{2048} - I_s| \quad \text{for } s = 2^k,\ k = 1,\ldots,10.$$

Do you observe the theoretical convergence rate $s^{-3+\varepsilon}$?

3. (QMC error) Let us consider QMC cubature for the parametric PDE problem (1)–(2). Let $s = 100$ and use the 100-dimensional generating vector $\boldsymbol{z}^{(100)} \in \mathbb{N}^{100}$, i.e., the first 100 elements of the vector contained in the file `offtheshelf2048.txt` available on the course page. We can apply a randomly shifted rank-1 lattice rule by drawing R shifts $\boldsymbol{\Delta}_1,\ldots,\boldsymbol{\Delta}_R$ from $\mathcal{U}([0,1]^s)$ and computing the cubatures

$$Q_n^{(r)} f := \frac{1}{n}\sum_{k=1}^n f(\mathrm{mod}(\boldsymbol{t}_k + \boldsymbol{\Delta}_r, 1) - \tfrac{\mathbf{1}}{\mathbf{2}}) \quad \text{for } r \in \{1,\ldots,R\},$$

where $f(\boldsymbol{y}) := G(u_{s,h}(\cdot, \boldsymbol{y}))$ for $\boldsymbol{y} \in [-1/2, 1/2]^s$ and $\boldsymbol{t}_k = \mathrm{mod}\big(\frac{k\boldsymbol{z}^{(100)}}{n}, 1\big)$. As our approximation of $\mathbb{E}[G(u_{s,h})]$, we take the average

$$\overline{Q}_{n,R} f := \frac{1}{R} \sum_{r=1}^{R} Q_n^{(r)} f.$$

We can estimate the root-mean-square error by computing

$$E_{n,R} := \sqrt{\frac{1}{R(R-1)} \sum_{r=1}^{R} (\overline{Q}_{n,R} f - Q_n^{(r)} f)^2}.$$

Fix a "reasonable" number of random shifts (e.g., you may choose $R = 4$ or $R = 8$ or $R = 16 \ldots$) and compute $E_{n,R}$ for $n \in \{2^{10}, 2^{11}, \ldots, 2^{15}\}$. What convergence rate do you observe?

To solve the PDE (1) numerically, use the finite element method. You can make the computations faster by using a coarser FE mesh, e.g., corresponding to mesh width $h = 2^{-4}$ (this is especially useful for debugging).

4. (QMC error for the lognormal model) Consider the PDE problem (1) equipped with a *lognormally* parameterized diffusion coefficient

$$a(\boldsymbol{x}, \boldsymbol{y}) := \exp\left(\sum_{j=1}^{\infty} y_j \psi_j(\boldsymbol{x}) \right), \quad \boldsymbol{x} \in D,\ y_j \in \mathbb{R},$$

where $D = (0,1)^2$, $f(\boldsymbol{x}) := x_1$, and $\psi_j(\boldsymbol{x}) := j^{-2} \sin(j\pi x_1) \sin(j\pi x_2)$ for $\boldsymbol{x} = (x_1, x_2) \in D$ as before. Let $G \in H^{-1}(D)$ be defined by (4). Let $s = 100$ be the truncation dimension and let $u_{s,h}(\cdot, \boldsymbol{y}) := u_s(\cdot, (y_1, \ldots, y_s))$ denote the dimensionally-truncated finite element approximation of the PDE solution for $\boldsymbol{y} \in \mathbb{R}^s$. In this case, we are interested in the integral

$$\mathbb{E}[G(u_{s,h})] = \int_{\mathbb{R}^s} G(u_{s,h}(\cdot, \boldsymbol{y})) \prod_{j=1}^{s} \phi(y_j)\, \mathrm{d}\boldsymbol{y} = \int_{(0,1)^s} G(u_{s,h}(\cdot, \Phi^{-1}(\boldsymbol{w}))\, \mathrm{d}\boldsymbol{w},$$

where $\phi(y) := \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}y^2}$ denotes the probability density function of the standard normal distribution $\mathcal{N}(0,1)$, $\Phi(y) := \int_{-\infty}^{y} \phi(t)\, \mathrm{d}t$ denotes the cumulative distribution function of $\mathcal{N}(0,1)$, and $\Phi^{-1}(\boldsymbol{w}) = [\Phi^{-1}(w_1), \ldots, \Phi^{-1}(w_s)]^{\mathrm{T}}$, where $\Phi^{-1}(w_j)$ denotes the inverse cumulative distribution function of $\mathcal{N}(0,1)$.

Modify the program you wrote in task 3 to estimate the root-mean-square error by computing

$$E_{n,R} := \sqrt{\frac{1}{R(R-1)} \sum_{r=1}^{R} (\overline{Q}_{n,R} f - Q_n^{(r)} f)^2},$$

for $n \in \{2^{10}, 2^{11}, \ldots, 2^{15}\}$, where

$$Q_n^{(r)} f := \frac{1}{n} \sum_{k=1}^{n} f(\Phi^{-1}(\mathrm{mod}(\boldsymbol{t}_k + \boldsymbol{\Delta}_r, 1))), \quad r \in \{1, \ldots, R\},$$

$$\overline{Q}_{n,R} f := \frac{1}{R} \sum_{r=1}^{R} Q_n^{(r)} f,$$

with $f(\boldsymbol{y}) := G(u_{s,h}(\cdot, \boldsymbol{y}))$ for $\boldsymbol{y} \in \mathbb{R}^s$, $\boldsymbol{\Delta}_r \sim \mathcal{U}([0,1]^s)$, and $\boldsymbol{t}_k := \mathrm{mod}\big(\frac{k\boldsymbol{z}^{(s)}}{n}, 1\big)$, where $\boldsymbol{z}^{(s)}$ denotes the same generating vector as in task 3.

What convergence rate do you observe? As in task 3, you are free to choose a "reasonable" finite element discretization level and the number of random shifts (e.g., $R = 4$ or $R = 8$ or $R = 16 \ldots$).

Hint: In Python, you can use the function `scipy.stats.norm.ppf` to evaluate the inverse cumulative distribution function for $\mathcal{N}(0,1)$.