A Fourier-based machine learning technique with application in engineering

Michaël Peigney

A Fourier-based machine learning technique with application in engineering

Michaël Peigney*

Lab Navier, Univ Gustave Eiffel, ENPC, CNRS, F-77447 Marne la Vallée, France

Abstract

The generic problem in supervised machine learning is to learn a function f from a collection of samples, with the objective of predicting the value taken by f for any given input. In effect, the learning procedure consists in constructing an explicit function that approximates f in some sense. In this paper is introduced a Fourier-based machine learning method which could be an alternative or a complement to neural networks for applications in engineering. The basic idea is to extend f into a periodic function so as to use partial sums of the Fourier series as approximations. For this approach to be effective in high dimension, it proved necessary to use several ideas and concepts such as regularization, Sobol sequences and hyperbolic crosses. An attractive feature of the proposed method is that the training stage reduces to a quadratic programming problem. The presented method is first applied to some examples of high-dimensional analytical functions, which allows some comparisons with neural networks to be made. An application to a homogenization problem in nonlinear conduction is discussed in detail. Various examples related to global sensitivity analysis, assessing effective energies of microstructures, and solving boundary value problems are presented.

*Corresponding author
Email address: michael.peigney@polytechnique.org (Michaël Peigney)

Keywords: Machine learning, Fourier decomposition, nonlinear homogenization, conductivity

1. Introduction

In recent years, machine learning techniques have become pervasive in a lot of fields and engineering is no exception. There is indeed a fast growing literature devoted to the use of machine learning techniques for predicting the response of a given mechanical system in terms of input parameters (see e.g. [1–4] for some examples). Implementing such techniques requires a database collecting the responses of the system for a large number of samples of the input parameters. Compared to domains in which machine learning is the most developed (such as image recognition), engineering has some specificities regarding the machine learning problems involved. For instance, problems encountered in engineering are mainly *regression* problems (i.e. predicting a function whose both input and output are continuously varying quantities) as opposed to *classification* problems (i.e predicting a function whose input or output are discrete quantities). In contrast with molecular dynamics [5], the functions to be predicted usually have no special structure.

Concerning applications in engineering, neural networks have been predominantly used so far. The main argument often invoked for justifying that choice is the so-called universal approximation theorem [6], which states that any given continuous function can be approximated (to any desired level of accuracy) by a neural network (it should be mentioned that the theorem does not give any clue on the number of neurons that should be used to reach a prescribed accuracy). Even though neural networks have proved to be an efficient tool in a lot of applications, they are not free from inconvenient. For instance, the training stage can be quite time consuming. Moreover, the minimization problem underlying the training stage has several local

minima, which notably means that several users training a neural network from the same database may obtain different results. Choosing the architecture of a multilayer neural network is also a critical and sensitive issue.

It should be emphasized that neural networks are not the only functions that enjoy a 'universal approximation' property. Fourier decomposition is one of the most famous examples. There are a lot of available results on the latter, both from the theoretical (convergence properties) and practical (Fast Fourier Transform algorithms) stand points. The guiding idea of this paper is to leverage those results to devise a Fourier-based machine learning technique that hopefully could be an alternative or a complement to neural networks for applications in engineering. Let $f(x_1, \cdots, x_d)$ denote the function to be learned. The basic principle, presented in Sect. 2, is to extend the function f into a function that is periodic with respect to each of its variables, thus allowing to one to use a multidimensional Fourier decomposition and consider partial sums of the Fourier series as approximations of f. The Fourier coefficients can be evaluated numerically from samples of the function f, using for instance a Fast Fourier Transform (FFT) algorithm. However, such a direct approach quickly breaks down as the number d of variables increases, because of the so-called curse of dimensionality: Using a FFT algorithm for calculating the Fourier coefficients requires a regular grid of sampling points, i.e. a number of samples that grows exponentially with d. This approach becomes impractical for $d \geq 4$, whereas machine learning problems arising in continuum mechanics often have a dimensionality of 6 or more. On a related note, the number of Fourier coefficients in a square partial sum of the Fourier series also grows exponentially with d. In Sects 3 and 4 are presented some strategies for countering the curse of dimensionality in the context of Fourier-based machine learning, namely regularization via a change of variables and the use of hyperbolic crosses. The latter have been originally introduced in [7]

for studying approximations a certain class of functions satisfying a bounded mixed derivative condition (we refer to [8] for an in-depth survey). Collecting all the various ideas introduced leads to a general machine-learning algorithm summarized in Sect. 5. Partial differentiation and global sensitivity analysis are discussed. The presented method is first applied to some examples of high-dimensional analytical functions, which allows some comparisons with neural networks to be made. Those examples allow us to compare the presented technique with neural networks. In Sect. 6 is presented an application to a homogenization problem in nonlinear conductivity. We use the Fourier-based machine learning technique to learn an energy function related to two-dimensional isotropic conductors whose constituent materials have a power-law behavior. Two machine learning models are generated, applying respectively to 2- and 3-phase composites. Several possible applications of the obtained machine learning models are presented: global sensitivity analysis, assessing the effective energy of given microstructures, and solving multiscale boundary value problems. Some concluding remarks follow in Sect. 7.

2. Periodic extension

Let f be a function that maps an input variable $\boldsymbol{x} = (x_1, \cdots, x_d) \in [0,1]^d$ to an output variable $f(\boldsymbol{x}) \in \mathbb{R}$. In supervised machine-learning, the generic problem is to learn the function f from a collection of M samples $\{\boldsymbol{x}^k, f(\boldsymbol{x}^k)\}_{1 \leq k \leq M}$, with the objective of being able to predict the value $f(\tilde{\boldsymbol{x}})$ for any given new input $\tilde{\boldsymbol{x}}$.

In effect, the learning procedure consists in constructing an explicit function $\tilde{f}$ that approximates the target function f in some sense. Our basic idea is to use Fourier series to build approximations of f: Let h be a $T-$*periodic* function of d variables $x_1, \cdots, x_d$ (i.e. h is periodic with period T with respect to each of its

variables). The Fourier coefficients $c(\boldsymbol{n},h)$ of h are defined for any $\boldsymbol{n} \in \mathbb{Z}^d$ by

$$c(\boldsymbol{n},h) = \frac{1}{T^d} \int_{[0,T]^d} h(\boldsymbol{x}) e^{-i\frac{2\pi}{T}\boldsymbol{n}\cdot\boldsymbol{x}} dx \tag{1}$$

where the operator $\cdot$ denotes the Euclidean scalar product in $\mathbb{R}^d$, i.e. $\boldsymbol{x}\cdot\boldsymbol{y} = \sum_{j=1}^d x_j y_j$ for any $\boldsymbol{x} = (x_1, \cdots, x_d)$ and $\boldsymbol{y} = (y_1, \cdots, y_d)$ in $\mathbb{R}^d$. Because of $T-$periodicity, the integration domain $[0,T]^d$ in (1) can be replaced by any domain of the form $[a_1, a_1+T] \times \cdots \times [a_d, a_d+T]$. The cubic partial sum $S_N h$ of rank N is defined by

$$S_N h(\boldsymbol{x}) = \sum_{\boldsymbol{n}\in[-N..N]^d} c(\boldsymbol{n},h) e^{i\frac{2\pi}{T}\boldsymbol{n}\cdot\boldsymbol{x}} \tag{2}$$

for any $\boldsymbol{x} \in \mathbb{R}^d$. Here and the following, the notation $[..]$ is used for integer intervals, e.g. $[-N\,..\,N] = \{-N, -N+1, \cdots, N-1, N\}$.

Provided h is square-integrable on $[0,T]^d$, it is well known [9] that

$$\int_{[0,T]^d} |S_N h(\boldsymbol{x}) - h(\boldsymbol{x})|^2 dx \xrightarrow[N\to+\infty]{} 0. \tag{3}$$

The given target function $f : [0,1]^d \mapsto \mathbb{R}$ is generally not periodic. For instance, there is no reason that $f(0, x_2, \cdots, x_d) = f(1, x_2, \cdots, x_d)$. However, f can be *extended* in a periodic function by considering the function $\tilde{f}$ defined on $[-1,1]^d$ by

$$\tilde{f}(x_1, \cdots, x_d) = f(|x_1|, \cdots, |x_d|). \tag{4}$$

Note that $\tilde{f}(1, x_2, \cdots, x_d) = \tilde{f}(-1, x_2, \cdots x_d)$ for any $x_2, \cdots,\ x_d$ in $[-1,1]^{d-1}$. A similar relation holds with respect to each variable, i.e.

$$\tilde{f}(x_1, \cdots, x_{j-1}, -1, x_{j+1}, \cdots, x_d) = \tilde{f}(x_1, \cdots, x_{j-1}, 1, x_{j+1}, \cdots, x_d) \tag{5}$$

for any j in $[1\,..\,d]$ and $(x_1, \cdots, x_{j-1}, x_{j+1}, \cdots, x_d)$ in $[-1,1]^{d-1}$. Property (5) allows the function $\tilde{f}$ to be viewed as a $2-$periodic function (i.e. periodic with period 2 with respect to each variable x_i).

Let $S_N f$ be the restriction on $[0,1]^d$ of the cubic partial sum $S_N \tilde{f}$ associated with the $2-$periodic function $\tilde{f}$, i.e.

$$S_N f(\boldsymbol{x}) = \sum_{\boldsymbol{n} \in [-N..N]^d} c(\boldsymbol{n}, \tilde{f}) e^{i\pi \boldsymbol{n} \cdot \boldsymbol{x}}$$

where

$$c(\boldsymbol{n}, \tilde{f}) = \frac{1}{2^d} \int_{[-1,1]^d} f(|x_1|, \cdots, |x_d|) e^{-i\pi \boldsymbol{n} \cdot \boldsymbol{x}} dx. \tag{6}$$

Let

$$|||f - S_N f|||^2 = \int_{[0,1]^d} |S_N f(\boldsymbol{x}) - f(\boldsymbol{x})|^2 dx$$

be the Mean Square Error (MSE) between f and $S_N f$. Since $|||f - S_N f|||^2 \leq \int_{[-1,1]^d} (\tilde{f} - S_N \tilde{f})^2$, property (3) implies that

$$|||f - S_N f|||^2 \xrightarrow[N \to +\infty]{} 0.$$

The family of functions $\{S_N f\}_N$ thus allows one to approximate f to any level of accuracy (in the sense of mean square error). Calculating $S_N f$ amounts to calculate the integrals $c(\boldsymbol{n}, \tilde{f})$ in (6) for all $\boldsymbol{n} \in [-N\,..\,N]^d$. In general, those integrals cannot be calculated exactly and need to be evaluated numerically from samples of the functions f (we will return to that issue later on). For an arbitrary $T-$periodic function h, constructing the partial sum $S_N h$ of rank N requires the calculation of $(2N+1)^d$ integrals of the form (1). That number can be reduced to $(N+1)^d$ for a function $\tilde{f}$ satisfying the symmetry conditions (4). The hypercube $[-1,1]^d$ in (6) can indeed be broken down into 2^d unit hypercubes $\mathcal{C}(\epsilon_1, \cdots, \epsilon_d)$ as

$$[-1,1]^d = \bigcup_{\substack{\epsilon_j \in \{-1,1\}, \\ 1 \leq j \leq d}} \mathcal{C}(\epsilon_1, \cdots, \epsilon_d)$$

where

$$\mathcal{C}(\epsilon_1, \cdots, \epsilon_d) = \prod_{1 \leq j \leq d} [\min(\epsilon_j, 0), \max(\epsilon_j, 0)].$$

It follows that

$$c(\boldsymbol{n}, \tilde{f}) = \frac{1}{2^d} \sum_{\substack{\epsilon_j \in \{-1,1\}, \\ 1 \leq j \leq d}} \int_{\mathcal{C}_{(\epsilon_1, \cdots, \epsilon_d)}} f(|x_1|, \cdots, |x_d|) e^{-i\pi \boldsymbol{n} \cdot \boldsymbol{x}} dx. \tag{7}$$

In the sum above, each integral can be transformed into an integral over the unit hypercube $[0,1]^d$ by using the change of variables $x'_j = \epsilon_j x_j$ for $j = 1, \cdots, d$. We find

$$\int_{\mathcal{C}_{(\epsilon_1, \cdots, \epsilon_d)}} f(|x_1|, \cdots, |x_d|) e^{-i\pi \boldsymbol{n} \cdot \boldsymbol{x}} dx = \int_{[0,1]^d} f(\boldsymbol{x}') \exp(-i\pi \sum_{j=1}^{d} \epsilon_j n_j x'_j) dx'. \tag{8}$$

Replacing in (7) and swapping the sum with the integral yields

$$c(\boldsymbol{n}, \tilde{f}) = \int_{[0,1]^d} f(\boldsymbol{x}) h(\boldsymbol{n}, \boldsymbol{x}) dx \tag{9}$$

where

$$h(\boldsymbol{n}, \boldsymbol{x}) = \frac{1}{2^d} \sum_{\substack{\epsilon_j \in \{-1,1\}, \\ 1 \leq j \leq d}} \exp(-i\pi \sum_{j=1}^{d} \epsilon_j n_j x_j) = \prod_{1 \leq j \leq d} \cos \pi n_j x_j. \tag{10}$$

Relations (9) and (10) imply that

$$c(\boldsymbol{n}, \tilde{f}) = c_{(|n_1|, \cdots, |n_d|)}(\tilde{f}) \tag{11}$$

for any $\boldsymbol{n} = (n_1, \cdots, n_d)$. The cubic partial sum $S_N f$ can thus be constructed by calculating only the $(N+1)^d$ coefficients $c(\boldsymbol{n}, \tilde{f})$ corresponding to positive multi-indices. In more detail, some manipulations reported in Appendix A show that

$$S_N f(x) = \sum_{\boldsymbol{n} \in [0..N]^d} 2^{\sigma(\boldsymbol{n})} c(\boldsymbol{n}, \tilde{f}) h(\boldsymbol{n}, \boldsymbol{x}) \tag{12}$$

where $\sigma(\boldsymbol{n})$ is the number of non-zero elements in $\boldsymbol{n} = (n_1, \cdots, n_d)$.

Example: Consider the function $f(x_1, \cdots, x_d)$ defined on $[0,1]^d$ by

$$f(x_1, \cdots, x_d) = (x_1 x_2 \cdots x_d)^{\frac{3}{2}}.$$

The coefficients $c(\boldsymbol{n}, \tilde{f})$ in (6) can be calculated in closed form and read as

$$c(\boldsymbol{n}, \tilde{f}) = \left(\frac{3}{2\pi^2}\right)^d \prod_{1 \le j \le d} \frac{1}{n_j^2} \left((-1)^{n_j} - \frac{C(\sqrt{2n_j})}{\sqrt{2n_j}}\right)$$

where $C(u) = \int_0^u \cos \frac{\pi x^2}{2} dx$ is the Fresnel integral. Parseval's identity gives

$$\frac{|||f - S_N f|||^2}{|||f|||^2} = 1 - \left(\frac{16}{25} + \frac{18}{\pi^4} \sum_{j=1}^{N} \frac{1}{j^4} \left((-1)^j - \frac{C(\sqrt{2j})}{\sqrt{2j}}\right)^2\right)^d \tag{13}$$

where $|||f|||^2 = 1/4^d$. In Fig. 1 is plotted the relative error $|||f - S_N f|||^2 / |||f|||^2$ for several values of the dimension d. As the dimension d increases, so does the minimal rank N needed for the relative error to be smaller than a prescribed threshold. For instance, in order to obtain a relative error below 10^{-3}, the minimal rank N is equal to 10 if $d = 10$, instead of 5 if $d = 2$. Such values of N may seem relatively small, but it should be kept in mind that calculating $S_N f$ amounts to calculate $(N+1)^d$ Fourier coefficients. In particular, calculating $S_{10} f$ in dimension 10 amounts to calculate 11^{10} Fourier coefficients. This would entail prohibitive computational costs in the general situation where the integrals defining the Fourier coefficients need to be estimated numerically. In the following are devised some strategies for reducing the number of Fourier coefficients needed for reconstructing f with a given accuracy.

3. Regularization via a change of variables

It can be observed that the periodic extension procedure presented in Sect. 2 involves some loss of regularity: if the function f is (say) of class C^1, its periodic

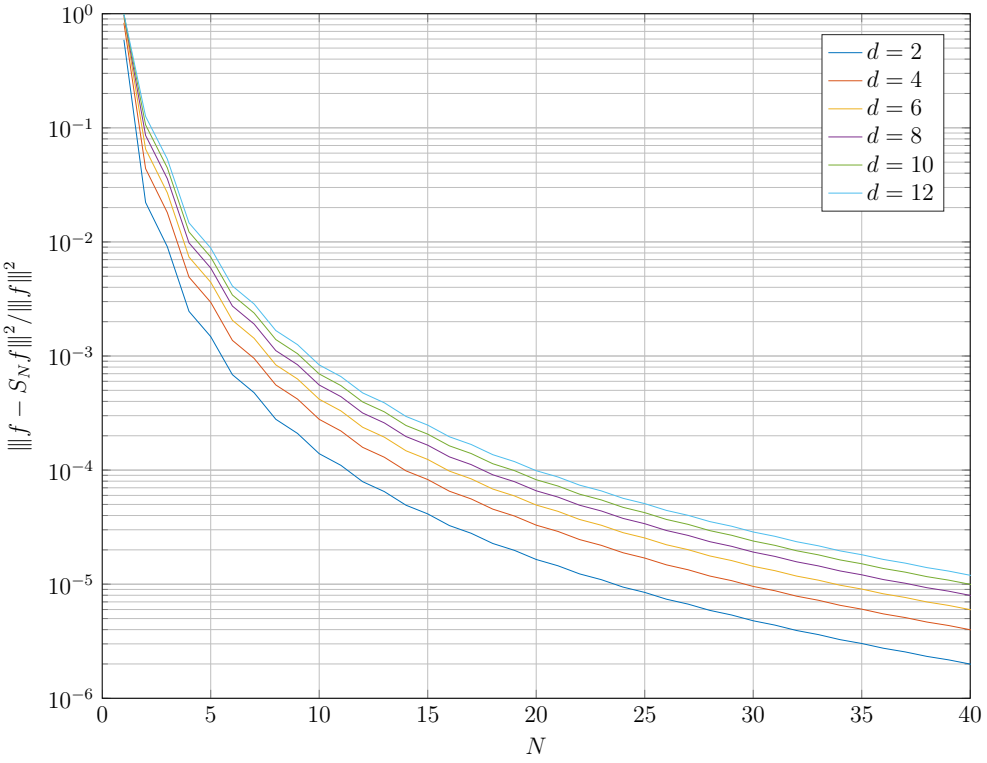

Figure 1: Relative mean square error $|||f - S_N f|||^2 / |||f|||^2$ as a function of the rank N, in the case $f(x_1, \cdots, x_d) = (x_1 x_2 \cdots x_d)^{\frac{3}{2}}$ with $d = 2, 4, 6, 8, 10, 12$.

extension $\tilde{f}$ is only *piecewise* C^1 because of discontinuities of the partial derivative $\partial \tilde{f} / \partial x_j$ on the hyperplanes $x_j = k$ with $k \in \mathbb{Z}$, see Fig. 2(left). This has an impact on the number of Fourier coefficients required to reconstruct f with a given accuracy (in the sense of mean square error). Recall indeed that the Fourier coefficients of a given $T-$periodic function h of class C^p in $\mathbb{R}^d$ verify [9]

$$c(\boldsymbol{n}, h) = O\left(\frac{1}{\|\boldsymbol{n}\|^p}\right), \tag{14}$$

where $\|\boldsymbol{n}\| = \sqrt{\sum_{j=1}^d n_j^2}$ is the Euclidean norm in $\mathbb{R}^d$. The smoother the function h is, the faster its Fourier coefficients decrease with $\|\boldsymbol{n}\|$ and the lower the expected

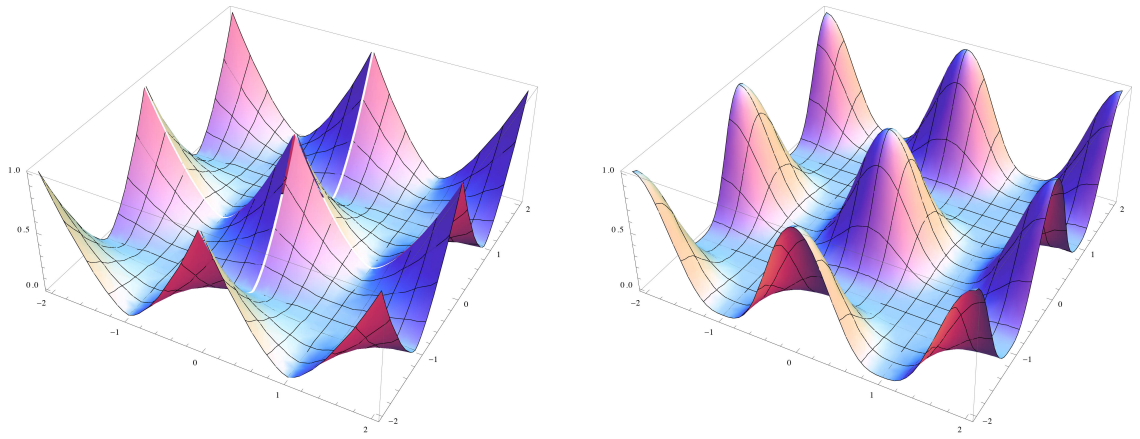

Figure 2: Periodic extensions of $f(x_1, x_2) = (x_1 x_2)^{\frac{3}{2}}$ (left) and $g(x_1, x_2) = f(\phi(x_1), \phi(x_2))$ (right) with $\phi(x) = \sin^2 \pi x/2$.

rank N necessary to reconstruct h is. The loss of regularity in the periodic extension procedure can be mitigated by using a change of variables, as is now explained. Consider a function $\phi : [0, 1] \to [0, 1]$ of class C^1 such that

$$\begin{aligned} &\phi(0) = 0, \phi(1) = 1, \\ &\phi'(0) = \phi'(1) = 0, \\ &\phi'(x) > 0 \text{ for } x \in (0, 1). \end{aligned} \tag{15}$$

Some useful examples of functions satisfying (15) are $\phi(x) = x^2(3 - 2x)$ and $\phi(x) = \sin^2 \frac{\pi x}{2}$ (Fig. 3). For any $\boldsymbol{x} = (x_1, \cdots, x_d)$ in $\mathbb{R}^d$ we set

$$\Phi(\boldsymbol{x}) = (\phi(x_1), \cdots, \phi(x_d)) \tag{16}$$

Conditions (15) ensure that Φ is invertible and maps the unit hypercube $[0, 1]^d$ to itself.

If f is of class C^1 then the function g defined on $[0, 1]^d$ by

$$g(\boldsymbol{x}) = f(\Phi(\boldsymbol{x})) \tag{17}$$

is also of class C^1 and the chain rule gives

$$\frac{\partial g}{\partial x_j}(\boldsymbol{x}) = \phi'(x_j)\frac{\partial f}{\partial x_j}(\Phi(\boldsymbol{x})) = 0 \tag{18}$$

for any $\boldsymbol{x}$ in $[0,1]^d$ such that $x_j \in \{0,1\}$. In geometrical terms, Eq. (18) means that the normal derivative of g vanishes on the boundary of the hypercube $[0,1]^d$.

Let $\tilde{g}$ be the $2-$periodic extension of g constructed as in (4), i.e.

$$\tilde{g}(x_1, \cdots, x_d) = f(\Phi(|x_1|, \cdots, |x_d|)) \tag{19}$$

for $x \in [-1,1]^d$. As a consequence of (18), $\tilde{g}$ is of class C^1: In particular, there is no discontinuity of the partial derivatives $\partial\tilde{g}/\partial x_j$ on the hyperplanes $x_j = k$ with $k \in \mathbb{Z}$, see Fig. 2(right). Hence the Fourier coefficient $c(\boldsymbol{n},\tilde{g})$ is expected to decrease faster with $\|\boldsymbol{n}\|$ than $c(\boldsymbol{n},\tilde{f})$ does, which is helpful for the reconstruction.

More generally, consider a change of variables $\phi : [0,1] \to [0,1]$ of class C^p satisfying (15) and the additional requirement

$$\phi^{(q)}(0) = \phi^{(q)}(1) = 0 \text{ for } 1 \leq q \leq p. \tag{20}$$

If f is of class C^p, it can be shown that the function $\tilde{g}$ in (19) is also of class C^p.

For a given function f, the idea is use a Fourier decomposition of $\tilde{g}$ rather than $\tilde{f}$. Cubic partial sums $S_N\tilde{g}$ translate as approximations T_Nf of f by the relation

$$T_Nf(\boldsymbol{x}) = S_N\tilde{g}(\Psi(\boldsymbol{x}))$$

where Ψ is the inverse function of Φ in (16). We have $\Psi(x_1, \cdots, x_d) = (\psi(x_1), \cdots, \psi(x_d))$ where ψ is the inverse function of ϕ.

Remark: Changes of variables ϕ verifying (20) can easily be constructed. If ϕ is of class C^p and satisfy (15), successive applications of the chain rule show indeed that

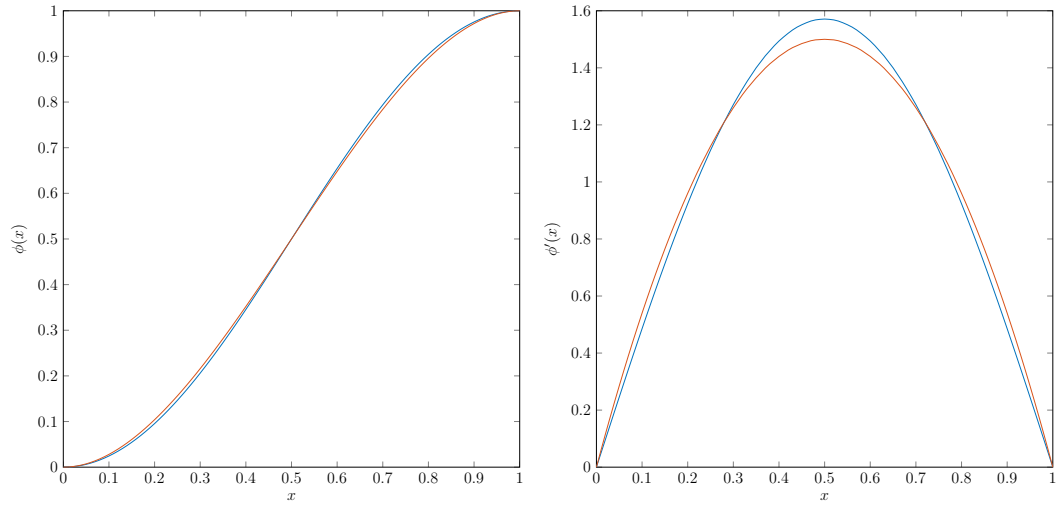

Figure 3: Examples of C^1 change of variables $\phi(x)$ (left) and their derivative $\phi'(x)$ (right): $\phi(x) = \sin^2 \frac{\pi x}{2}$ (blue), $\phi(x) = x^2(3-2x)$ (red).

the p^{th} iterate $\phi^{(\circ p)} = \phi \circ \phi \circ \cdots \circ \phi$ verifies (15) as well as the additional requirement (20).

Example: Consider again the function $f(x_1, \cdots, x_d) = (x_1 x_2 \cdots x_d)^{\frac{3}{2}}$ as in Sect. 2. Using the change of variables $\phi(x) = \sin^2 \frac{\pi x}{2}$, the function g defined in (17) specializes as

$$g(x_1, \cdots, x_d) = (\sin \frac{\pi x_1}{2} \sin \frac{\pi x_2}{2} \cdots \sin \frac{\pi x_d}{2})^3. \tag{21}$$

Calculating the Fourier coefficients $c(\boldsymbol{n},\tilde{g})$ gives

$$c(\boldsymbol{n}, \tilde{g}) = \left(\frac{12}{\pi}\right)^d \prod_{1 \le j \le d} \frac{1}{(4n_j^2 - 1)(4n_j^2 - 9)}$$

and

$$\frac{|||g - S_N g|||^2}{|||g|||^2} = 1 - \left(\frac{32}{\pi^2}\right)^d \left(\frac{8}{15} + 9 \sum_{1 \le j \le d} \frac{1}{(4n_j^2 - 1)^2 (4n_j^2 - 9)^2}\right)^d \tag{22}$$

with $|||g|||^2 = (5/16)^d$. In Fig. 4(red solid line) is plotted the relative error (22) as a function of N, in the case $d = 6$. The relative error $|||f - S_N f|||^2$ obtained originally

in (13) is represented as a blue solid line in Fig. 4. The change of variables technique is clearly beneficial, whatever the required level of error. For instance, the minimum rank N for obtaining a relative error below 10^{-5} drops from 35 to 4. The numerical results suggest that $|||f - S_N f|||^2 = O(1/N^3)$ whereas $|||g - S_N g|||^2 = O(1/N^6)$. Note that the error $|||f - T_N f|||^2$ cannot be calculated in closed form but is controlled by $|||g - S_N g|||^2$. Using the change of variables $\boldsymbol{x} = \Phi(\boldsymbol{y})$, we have indeed

$$\begin{aligned} |||f - T_N f|||^2 &= \int_{[0,1]^d} |f(\boldsymbol{x}) - T_N f(\boldsymbol{x})|^2 dx \\ &= \int_{[0,1]^d} |g(\boldsymbol{y}) - S_N g(\boldsymbol{y})|^2 |\det(\Phi'(\boldsymbol{y}))| dy \\ &\leq (\sup_{[0,1]} \phi')^d \int_{[0,1]^d} |g(\boldsymbol{y}) - S_N g(\boldsymbol{y})|^2 dy \end{aligned}$$

i.e. $|||f - T_N f|||^2 \leq (\sup_{[0,1]} \phi')^d |||g - S_N g|||^2$. As a consequence, the rate of decrease of $|||f - T_N f|||^2$ with N is the same as $|||g - S_N g|||^2$. In the present case, we thus have $|||f - T_N f|||^2 = O(1/N^6)$.

In (21) a C^1 change of variables has been used. It is tempting to repeat the argument in a sequential fashion: from the expectation that a C^{p+1} change of variables leads to a faster decrease of the error than a C^p change of variables, one is led to use a change of variables that is as smooth as possible, i.e. with the same regularity as the function f itself. There are, however, some practical limitations to that reasoning as illustrated in Fig. 4. The relative error corresponding to the C^2 change of variables $\sin^2(\pi(\sin^2 \pi x/2)/2)$ is shown as a yellow solid line in Fig. 4. For high values of N (or, equivalently, small values of the target error), a C^2 change of variables yields a faster decrease of the relative error than a C^1 change of variables, as could be expected from (14). However, for a relative error above 10^{-6}, a C^2 change of variables actually turns out be detrimental compared to a C^1 change of variables. For instance, the minimum rank N for reaching a relative error

below 10^{-4} is equal to 5 with a C^2 change of variables, instead of 3 with a C^1 change of variables. Anticipating on the applications that will be presented later on, it is worth pointing out that practical levels of required error fall in the range where a C^1 change of variables is preferable.

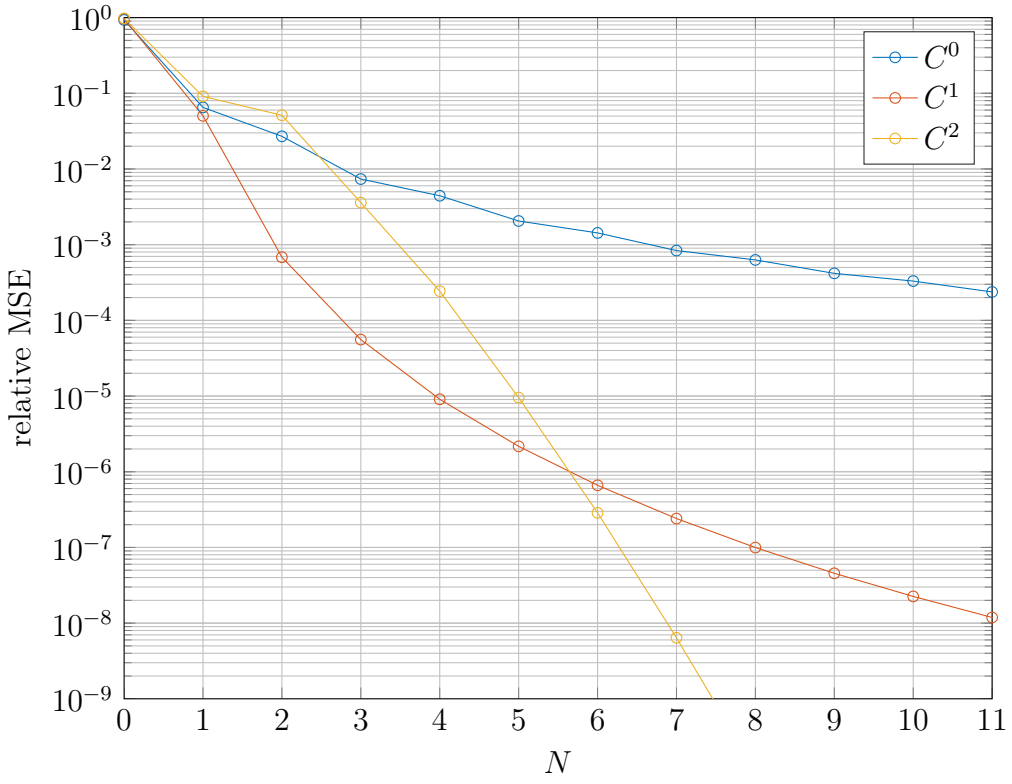

Figure 4: Relative MSE between $f(\Phi^{(\circ p)})$ and $S_N f(\Phi^{(\circ p)})$ as a function of the rank N, in the case $f(x_1, \cdots, x_d) = (x_1 x_2 \cdots x_d)^{\frac{3}{2}}$, $\phi(x) = \sin^2 \pi x/2$ with $d = 6$ and $p = 0$ (no change of variables),1 (C^1 change of variables), 2 (C^2 change of variables).

4. Hyperbolic crosses

Cubic partial sums have been considered so far, but one has to bear in mind that there is not a unique way to reconstruct a given $T-$periodic function h from its

Fourier coefficients. For instance, one could consider the *spherical* partial sums

$$\sum_{\boldsymbol{n}\in\mathbb{Z}^d:\|\boldsymbol{n}\|\leq N} c(\boldsymbol{n},h)e^{i\frac{2\pi}{T}\boldsymbol{n}\cdot\boldsymbol{x}}.$$

More generally, one can consider the partial sum

$$S_\Lambda h = \sum_{\boldsymbol{n}\,\in\,\Lambda} c(\boldsymbol{n},h)e^{i\frac{2\pi}{T}\boldsymbol{n}\cdot\boldsymbol{x}}$$

over any given set Λ of multi-indices in $\mathbb{Z}^d$. For any sequence of sets $\{\Lambda(N)\}_N$ such that $\Lambda(N)\to\mathbb{Z}^d$ as $N\to\infty$, the error $\left|\!\left|\!\left|S_{\Lambda(N)}h-h\right|\!\right|\!\right|$ converges towards 0 as $N\to\infty$[1].

Since the computational cost of calculating S_Λ is proportional to the cardinality of Λ, the question arises as to what is the best choice of sets Λ, i.e. the sets minimizing $|||S_\Lambda h-h|||$ over all sets of given cardinality. Formally, that problem can easily be solved: since the Fourier coefficients $c(\boldsymbol{n},h)$ tends to 0 as $\|\boldsymbol{n}\|\to\infty$, the sequence $\{c(\boldsymbol{n},h)\}$ can be ordered in such a way that

$$|c(\boldsymbol{n}_1,h)|\geq|c(\boldsymbol{n}_2,h)|\geq|c(\boldsymbol{n}_3,h)|\geq\cdots \tag{23}$$

where $j\mapsto\boldsymbol{n}_j$ is the one-to-one mapping from $\mathbb{N}$ to $\mathbb{Z}^d$. Parseval's identity gives

$$|||S_\Lambda h-h|||^2=\sum_{\boldsymbol{n}\,\notin\,\Lambda}|c(\boldsymbol{n},h)|^2. \tag{24}$$

From (23) and (24) it can easily be seen that the set that minimizes $|||S_\Lambda h-h|||$ for a given cardinality N is

$$\{\boldsymbol{n}_j:1\leq j\leq N\} \tag{25}$$

[1]Proof: For any given $\epsilon>0$, (3) and Parseval's identity imply that $\sum_{\boldsymbol{n}\notin[-N..N]^d}|c(\boldsymbol{n},h)|^2<\epsilon$ for some N. Since $\Lambda(M)\xrightarrow[M\to\infty]{}\mathbb{Z}^d$, there exists some M such that $[0\,..\,N]^d\subset\Lambda(M')$ for any $M'\geq M$. We have $\left|\!\left|\!\left|S_{\Lambda(M')}h-h\right|\!\right|\!\right|^2=\sum_{\boldsymbol{n}\notin\Lambda(M')}|c(\boldsymbol{n},h)|^2\leq\sum_{\boldsymbol{n}\notin[-N..N]^d}|c(\boldsymbol{n},h)|^2<\epsilon$ for any $M'>M$.

which corresponds to picking the N largest Fourier coefficients.

In practice, Eq. (25) is difficult to use because the ordering of the Fourier coefficients is not known beforehand. The main observation is that sets (25) may largely differ from cubes (or spheres) because all the directions in $\mathbb{Z}^d$ are not equivalent as far as the distribution of $|c(\boldsymbol{n},h)|$ is concerned. In other words, a given hypercube (or hypersphere) in $\mathbb{R}^d$ typically contains a lot of Fourier coefficients that are very small and are negligible for the reconstruction. This situation is especially critical as the dimension d increases because the number of Fourier coefficients in a hypercube of given length (or in a hypersphere of given radius) grows exponentially with d. Such difficulties can be avoided by making use of a special family of sets, known as hyperbolic crosses. A typical example of hyperbolic cross is the set $\mathrm{H}(N)$ defined for $N > 0$ by

$$\mathrm{H}(N) = \{\boldsymbol{n} \in \mathbb{Z}^d : \pi(\boldsymbol{n}) \leq N\}$$

where

$$\pi(\boldsymbol{n}) = \prod_{1 \leq j \leq d} \max(1, |n_j|). \tag{26}$$

An example of hyperbolic cross is shown in Fig. 5 for $d = 3$. Hyperbolic crosses have originally been introduced for $T-$periodic functions h satisfying some bounded mixed derivative condition [7, 8]. Several conditions of such type can be considered. For our purpose, it is sufficient to use the following definition: a $T-$periodic function h has bounded mixed derivative if

$$\frac{\partial^{d'} h}{\partial x_{j_1} \partial x_{j_2} \cdots \partial x_{j_{d'}}} \text{ is square-integrable on } [0, T]^d \tag{27}$$

for any $d' \in [0\,..\,d]$ and any distinct indices $\{j_1, \cdots j_{d'}\}$ in $[1\,..\,d]$. As an example, in $\mathbb{R}^2$, $T-$periodic functions h with bounded mixed derivative are characterized by the

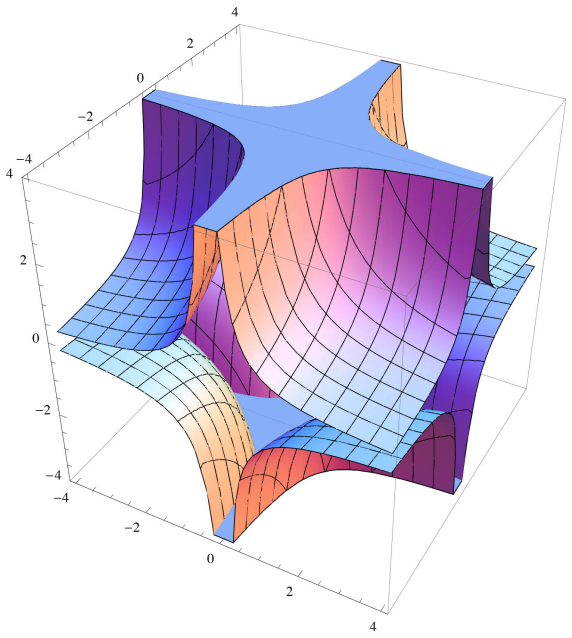

Figure 5: A smooth hyperbolic cross in $\mathbb{R}^3$, defined by $\prod_{1\le j\le 3} \max(1, |x_j|) \le 1$.

fact that

$$h,\ \frac{\partial h}{\partial x_1}, \frac{\partial h}{\partial x_2}, \frac{\partial^2 h}{\partial x_1 \partial x_2}$$

are square-integrable on $[0,T]^2$.

For functions with bounded mixed derivative in the sense of (27), it can be proved (see Appendix B) that

$$|c(\boldsymbol{n},h)| = O\left(\frac{1}{\pi(\boldsymbol{n})}\right) \tag{28}$$

The function $\pi(\boldsymbol{n})$ thus controls the magnitude of the Fourier coefficients for functions with bounded mixed derivative, in the same fashion as the euclidean norm $\|\boldsymbol{n}\|$ (or any other norm) controls the magnitude of the Fourier coefficients for functions of class C^1. For the purpose of approximating h, Eq. (28) suggests to use partial sums over sets of the form $\{\boldsymbol{n} \in \mathbb{Z}^d : \pi(n) \leq N\}$ (i.e. hyperbolic crosses), in the same way as partial sums over the sets $\{\boldsymbol{n} \in \mathbb{Z}^d : \|\boldsymbol{n}\| \leq N\}$ (i.e hyperspheres or hypercubes, depending on the choice of the norm) are used for functions of class C^1. In that regard, a salient feature of hyperbolic crosses $\mathrm{H}(N)$ is that (see Appendix C)

$$\mathrm{Card}\,\mathrm{H}(N) = O(N \log^{d-1} N) \tag{29}$$

The cardinality of $\mathrm{H}(N)$ thus grows relatively slowly with the dimension. This is in contrast with hyperspheres and hypercubes, whose cardinality grows as $O(N^d)$ i.e. exponentially with the dimension.

Results (28) and (29) are the main motivation for using partial sums over hyperbolic crosses: for a same accuracy, they tentatively allow one to build approximations using less Fourier coefficients than cubic (or spherical) partial sums would require.

We wish to apply that idea to $2-$periodic functions $\tilde{g}$ of the form (19). To do so, a natural prerequisite is that $\tilde{g}$ has bounded mixed derivative in the sense of (27).

In that regard, it can easily be verified that $\tilde{g}$ satisfies condition (27) if

- (i) the mixed derivatives of the target function f are bounded on $[0,1]^d$, i.e. $\dfrac{\partial^{d'} f}{\partial x_{j_1} \partial x_{j_2} \cdots \partial x_{j_{d'}}}$ is square-integrable on $[0,1]^d$ for any $d' \in [0\,..\,d]$ and any distinct indices $\{j_1, \cdots j_{d'}\}$ in $[1\,..\,d]$
- (ii) ϕ is of class C^1

Crucially in (ii), the change of variables ϕ does not have to be of class C^d. This is important because, as mentioned in Sect. 3, a C^1 change of variable offers better performances than a higher order change of variables for practical level of tolerance.

Partial sums over hyperbolic crosses will be referred to as *hyperbolic partial sums* in the following. We have in particular

$$S_{\mathrm{H}(N)}\tilde{g}(\boldsymbol{x}) = \sum_{\boldsymbol{n}\in\mathrm{H}(N)} c(\boldsymbol{n},\tilde{g}) e^{i\pi \boldsymbol{n}\cdot\boldsymbol{x}} \tag{30}$$

Since the Fourier coefficients $c(\boldsymbol{n},\tilde{g})$ satisfy (11), the sum over $\boldsymbol{n}$ in (30) can be restricted to positive values, as in (12). We obtain

$$S_{\mathrm{H}(N)}\tilde{g}(\boldsymbol{x}) = \sum_{\boldsymbol{n}\in\mathrm{H}^+(N)} 2^{\sigma(\boldsymbol{n})} c(\boldsymbol{n},\tilde{g}) h(\boldsymbol{n},\boldsymbol{x}) \tag{31}$$

where

$$\mathrm{H}^+(N) = \{\boldsymbol{n}\in\mathbb{N}^d : \pi(\boldsymbol{n}) \le N\} \tag{32}$$

is the intersection of the hyperbolic cross $\mathrm{H}(M)$ with the positive orthant. We refer to Appendix A for details on the derivation of (31).

Example: Consider the function g in (21) with $d = 6$. Fig. 6 provides some insight on the computational efficiency of the hyperbolic partial sum $S_{\mathrm{H}(5)}g$ compared to the cubic partial sum $S_3 g$. In blue is shown the MSE obtained by considering the

partial sum over the N' largest Fourier coefficients in the hypercube $[0,3]^d$, with $N' = 1, \cdots, \mathrm{Card}\,([0\,..\,3]^d)$. The obtained curve displays a large flat plateau, meaning that a large proportion (about 70%) of the calculated Fourier coefficients are very small and play a negligible role in the construction. In contrast, the analog curve for the hyperbolic partial sum does not exhibit such a plateau. This illustrates the fact that hyperbolic crosses are closer to the optimal sets (25) than hypercubes are.

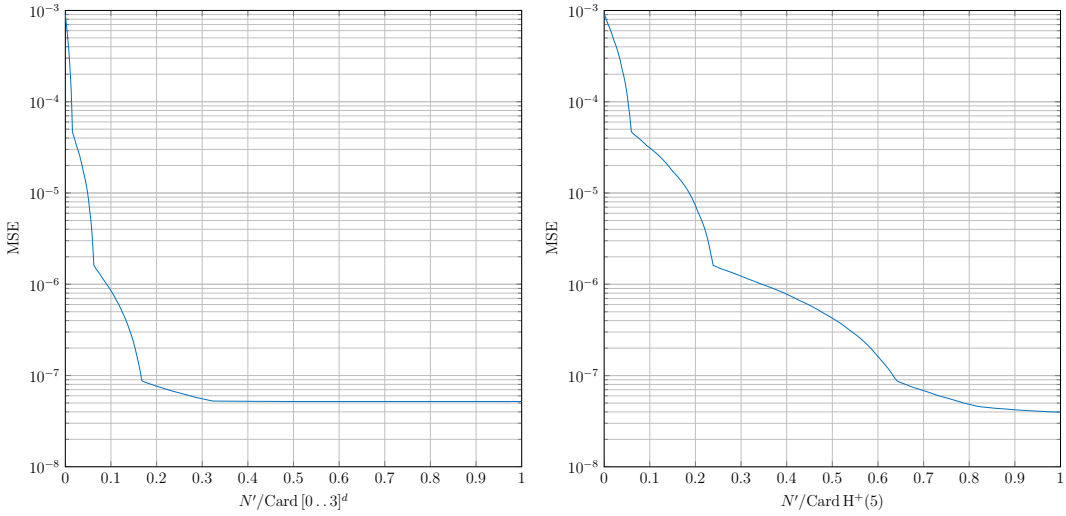

Figure 6: Mean Squared Error obtained by keeping only the N' largest coefficients in the cubic partial sum $S_3 g$ (left) and in the hyperbolic partial sum $S_{\mathrm{H}(5)} g$ (right). The function g is $g(x_1, \cdots, x_d) = (\sin \frac{\pi x_1}{2} \cdots \sin \frac{\pi x_d}{2})^3$ with $d = 6$.

Remark: For a C^1 target function with bounded mixed derivative, combining the estimates (14) and (28) gives

$$c(\boldsymbol{n}, h) = O(\frac{1}{\max(\pi(\boldsymbol{n}), \|\boldsymbol{n}\|)})$$

Instead of hyperbolic crosses $H(N)$, one may therefore consider partial sums over the sets

$$\tilde{H}(N) = \{\boldsymbol{n} \in \mathbb{N}^d : \max(\pi(\boldsymbol{n}), \|\boldsymbol{n}\|) \leq N\}$$

which is the intersection of $H(N)$ with a hypersphere of radius N.

5. Algorithm

Collecting the ideas presented so far, we choose to approximate the target function f by considering functions Tf of the form

$$Tf(\boldsymbol{x}) = S_{\mathrm{H}(N)}\tilde{g}(\Psi(\boldsymbol{x})) = \sum_{\boldsymbol{n}\in\mathrm{H}^{+}(N)} C(\boldsymbol{n}, f)h(\boldsymbol{n}, \Psi(\boldsymbol{x})) \tag{33}$$

where

$$C(\boldsymbol{n}, f) = 2^{\sigma(\boldsymbol{n})} \int_{[0,1]^d} f(\Phi(\boldsymbol{x}))h(\boldsymbol{n}, \boldsymbol{x})dx. \tag{34}$$

We recall from (10) and (16) that $h(\boldsymbol{n}, \boldsymbol{x}) = \prod_{j=1}^{d} \cos \pi n_j x_j$ and $\Phi(\boldsymbol{x}) = ((\phi(x_1), \cdots, \phi(x_d))$ where ϕ is a given function satisfying (15). The hyperbolic cross $\mathrm{H}^{+}(N)$ is the set of multi-indices $\boldsymbol{n} \in \mathbb{N}^d$ such that $\prod_{j=1}^{d} \max(1, n_j) \leq N$. In (33), Ψ is the inverse function of Φ.

In (33), all the information on the target function f is contained in the coefficients $C(\boldsymbol{n}, f)$ defined by the integrals (34). Those integrals can be estimated from a training database, i.e. from samples of the functions f. Let $\{\boldsymbol{x}^k, y^k\}_{1\leq k\leq M}$ denote the training database, where $y^k = f(\Phi(\boldsymbol{x}^k))$, M is the number of samples, and each $\boldsymbol{x}^k$ is a value in the unit hypercube $[0, 1]^d$.

It is tempting to use regularly spaced sample points $\{\boldsymbol{x}^k\}$, such as $(m_1, \cdots, m_d)/M$ where $m_k \in [-M\,..\,M]$ and M is given. This allows one to approximate the integral in (34) by the Riemann sum

$$\frac{1}{M}\sum_{k=1}^{M} y^k h(\boldsymbol{n}, \boldsymbol{x}^k). \tag{35}$$

For regularly space sample points, the coefficients (35) are essentially the $d-$dimensional discrete Fourier transform of the sampled values $\{y^k\}$. Hence one can take advantage

of Fast Fourier Transform (FFT) algorithms for calculating the coefficients (35) in an efficient way. That approach, however, breaks down rapidly as the dimension d increases because the number of sampling points in a regular grid is equal to $(2M+1)^d$ and thus grows exponentially with d.

An alternative way to evaluate the integral in (6) is to use Monte-Carlo integration, which consists in using (35) with *random* sample points $\{\boldsymbol{x}^k\}$. One of the main results in Monte Carlo integration ensures that

$$E\left(\frac{1}{M}\sum_{k=1}^{M} y^k h(\boldsymbol{n},\boldsymbol{x}^k)\right) - \frac{C(\boldsymbol{n},f)}{2^{\sigma(\boldsymbol{n})}} = O\left(\frac{1}{\sqrt{M}}\right) \tag{36}$$

where E denotes the statistical average [10]. The statistical error committed by the Monte-Carlo approximation thus grows no faster that $1/\sqrt{M}$ where M is the number of sample points. Most crucially, that result does not depend on the dimension d. A related approach – that we favor in the following – consists in using *low-discrepancy sequences* [11] instead of random sequences in (35). Loosely speaking, low-discrepancy sequences are designed to cover the hypercube $[0,1]^d$ in a more uniform fashion than random sequences, thus allowing for a better approximation of integrals. More precisely, taking $\{\boldsymbol{x}^k\}$ as a low-discrepancy sequence ensures (see e.g. [11]) that

$$\frac{1}{M}\sum_{k=1}^{M} y^k h(\boldsymbol{n},\boldsymbol{x}^k) - \frac{C(\boldsymbol{n},f)}{2^{\sigma(\boldsymbol{n})}} = O\left(\frac{(\log M)^d}{M}\right)$$

The obtained error estimate grows (slowly) with the dimension d, but is always better than the error estimate (36) for the Monte-Carlo procedure. There are several known algorithms for constructing low-discrepancy sequences. Some of the most widespread ones are Sobol sequences [12] and Halton sequences [13].

The simplest version of the proposed machine-learning procedure, Algorithm 1, consists in using (35) for calculating Tf for some given N. The output of Algorithm

1 is a list $\{\boldsymbol{n}^j, C_j\}_{1\leq j\leq J}$ that defines an approximation Tf of the target function f. That approximation Tf can be evaluated at any new input $\boldsymbol{x}$ by the formula

$$Tf(\boldsymbol{x}) = \sum_{j=1}^{J} C_j h(\boldsymbol{n}^j, \Psi(\boldsymbol{x})) \tag{37}$$

Input : $N, \{\boldsymbol{x}^k, \boldsymbol{y}^k\}_{1\leq k\leq M}$

Output: J, $\{\boldsymbol{n}^j, C_j\}_{1\leq j\leq J}$

$J \leftarrow 0$;

for $\boldsymbol{n} \in H^+(N)$ **do**

$J \leftarrow J + 1$;

$\boldsymbol{n}^J \leftarrow \boldsymbol{n}$;

$C_J \leftarrow \dfrac{2^{\sigma(\boldsymbol{n})}}{M} \displaystyle\sum_{k=1}^{M} y^k h(\boldsymbol{n}, \boldsymbol{x}^k)$;

end

Algorithm 1: Learning procedure.

In Algorithm 1, the coefficients $C_1,\cdots, C_J$ are calculated independently. In practice, better results are obtained by introducing a minimization procedure for identifying $C_1,\cdots, C_J$ simultaneously. For any given scalar c, note indeed that

$$|||g - c\, h(\boldsymbol{n}, \cdot)|||^2 = |||g|||^2 + c^2 \int_{[0,1]^d} h(\boldsymbol{n}, \boldsymbol{x})^2 dx - 2c \sum_{\boldsymbol{n}'\in\mathbb{N}^d} C(\boldsymbol{n}', f) \int_{[0,1]^d} h(\boldsymbol{n}, \boldsymbol{x}) h(\boldsymbol{n}', \boldsymbol{x}) dx$$

where the decomposition $g(\boldsymbol{x}) = \sum_{\boldsymbol{n}'\in\mathbb{N}^d} C(\boldsymbol{n}', f) h(\boldsymbol{n}', \boldsymbol{x})$ has been used. A simple calculation from (10) gives

$$2^{\sigma(\boldsymbol{n})} \int_{[0,1]^d} h(\boldsymbol{n}, \boldsymbol{x}) h(\boldsymbol{n}', \boldsymbol{x}) dx = \delta_{\boldsymbol{n}\boldsymbol{n}'}$$

where δ is the Kronecker operator. It follows that

$$|||g - c\, h(\boldsymbol{n}, \cdot)|||^2 = |||g|||^2 + \frac{(c - C(\boldsymbol{n}, f))^2 - C(\boldsymbol{n}, f)^2}{2^{\sigma(\boldsymbol{n})}}$$

so that

$$C(\boldsymbol{n}, f) = \arg\min_{c} |||g - c\, h(\boldsymbol{n}, \cdot)|||^2. \tag{38}$$

In a similar fashion, for any given $\boldsymbol{n}^1, \cdots, \boldsymbol{n}^J$ we have

$$(C(\boldsymbol{n}^1, f), \cdots, C(\boldsymbol{n}^J, f)) = \arg\min_{(c_1,\cdots,c_J)} \left|\left|\left| g - \sum_{j=1}^{J} c_j\, h(\boldsymbol{n}^j, \cdot) \right|\right|\right|^2 \tag{39}$$

Algorithm 2 consists in using (39) for identifying all the coefficients $C(\boldsymbol{n}^1, f), \cdots, C(\boldsymbol{n}^J, f)$. In more detail, the integral defining $\left|\left|\left| g - \sum_{j=1}^{J} c_j\, h(\boldsymbol{n}^j, \cdot) \right|\right|\right|^2$ is approximated by the finite sum

$$\frac{1}{M} \sum_{k=1}^{M} \left(y^k - \sum_{j=1}^{J} c_j h(\boldsymbol{n}^j, \boldsymbol{x}^k) \right)^2.$$

The coefficients $C(\boldsymbol{n}^1, f), \cdots, C(\boldsymbol{n}^J, f)$ are obtained by solving the quadratic problem

$$\min_{(c_1,\cdots,c_J)} \sum_{k=1}^{M} \left(y^k - \sum_{j=1}^{J} c_j h(\boldsymbol{n}^j, \boldsymbol{x}^k) \right)^2.$$

Compared to Algorithm 1, Algorithm 2 was found to give better results because it partially makes up for the approximation error of the Fourier integrals that comes with the discrete formula (35). The downside is an increase in the computational

cost since an additional quadratic problem needs to be solved.

Input : N,$\{\boldsymbol{x}^k, \boldsymbol{y}^k\}_{1\leq k\leq M}$

Output: $J, \{\boldsymbol{n}^j, C_j\}_{1\leq j\leq J}$

$J \leftarrow 0$;

for $\boldsymbol{n} \in H^+(N)$ **do**

  $J \leftarrow J + 1$;

  $\boldsymbol{n}^J \leftarrow \boldsymbol{n}$;

  $\{X_j^k\}_{1\leq k\leq M} \leftarrow \{h(\boldsymbol{n}, \boldsymbol{x}^k)\}_{1\leq k\leq M}$;

end

$$\{C_j\}_{1\leq j\leq J} = \arg\min_{\{c_j\}} \sum_{k=1}^{M} \left(y^k - \sum_{l=1}^{J} c_k X_l^k \right)^2 ;$$

Algorithm 2: Learning procedure (variant).

5.1. Sensitivity analysis

In some applications, the partial derivatives $\partial Tf/\partial x_i$ are required (an example will be provided later in Sect. 6). Writing $\boldsymbol{n}^j = (n_1^j, \cdots, n_d^j)$, we note from (10) and (37) that

$$\frac{\partial Tf}{\partial x_i} = -\sum_{j=1}^{J} \pi n_i^j C_j \psi'(x_i) h(\tilde{\boldsymbol{n}}^j, \Psi(\tilde{\boldsymbol{x}})) \sin \pi n_i^j \psi(x_i) \tag{40}$$

where $\tilde{\boldsymbol{x}} = (x_1, \cdots, x_{i-1}, x_i, \cdots, x_d) \in \mathbb{R}^{d-1}$, $\tilde{\boldsymbol{n}}^j = (n_1^j, \cdots, n_{i-1}^j, n_{i+1}^j, \cdots, n_d^j) \in \mathbb{N}^{d-1}$. Expression (40) can notably be used for global sensitivity analysis. In machine learning, it is indeed common practice to introduce sensitivity indices for measuring the influence of each variable x_i on the target function. This provides some useful information on the global behavior of the function under consideration. Sensitivities indices, here denoted by S_i, are usually expressed as scalars in $[0, 1]$ such that $\sum_{i=1}^{d} S_i = 1$. Among the possible choices for S_i, we adopt a definition related to the

norm of the partial derivatives:

$$S_i = \frac{\left\| \left\| \frac{\partial Tf}{\partial x_i} \right\| \right\|^2}{\sum_{i=1}^{d} \left\| \left\| \frac{\partial Tf}{\partial x_i} \right\| \right\|^2} \tag{41}$$

The higher S_i is, the bigger the influence of the variable x_i. For a machine learning model of the form (37), use of (40) leads to

$$\left\| \left\| \frac{\partial Tf}{\partial x_1} \right\| \right\|^2 = \sum_{j,k=1}^{J} C_j C_k J(n_1^j, n_1^k) I(n_2^j, n_2^k) \cdots I(n_d^j, n_d^k) \tag{42}$$

where

$$I(n, n') = \int_0^1 \phi'(y) \cos \pi n y \cos \pi n' y dy \tag{43}$$

and

$$J(n, n') = \pi^2 n n' \int_0^1 \frac{1}{\phi'(y)} \sin \pi n y \sin \pi n' y dy. \tag{44}$$

The expression of $\left\| \left\| \frac{\partial Tf}{\partial x_i} \right\| \right\|^2$ for $i \neq 1$ is obtained from (42) by permutation of the indices. For the function $\phi(y) = \sin^2 \frac{\pi y}{2}$, the integrals $I(n, n')$ and $J(n, n')$ can be calculated in closed form. They are given by

$$I(n, n') = \begin{cases} \frac{1}{2} \left(\frac{1}{1 - (n + n')^2} + \frac{1}{1 - (n - n')^2} \right) & \text{if } n + n' \text{ even} \\ 0 & \text{otherwise} \end{cases}$$

and

$$J(n, n') = \begin{cases} 4nn' \sum_{\substack{k = n' - n + 1 \\ k \text{ odd}}}^{n+n'-1} \frac{1}{k} & \text{if } n + n' \text{ even} \\ 0 & \text{otherwise.} \end{cases}$$

In the above expression of $J(n, n')$, n and n' are ordered in such a way that $n \leq n'$.

5.2. Reconstruction of analytical functions in high dimension

Algorithms 1 and 2 were implemented in Matlab. An interior-point method [14] was used to solve the quadratic programming problem in Algorithm 2. A natural question is to investigate how the proposed machine-learning procedure compares with neural networks. The latter has indeed become the most widespread machine-learning techniques in a lot of applications. Comparing machine-learning techniques is somewhat delicate because many factors are involved, such as ease of implementation, computational cost and robustness. A simple but critical performance indicator, however, is the reconstruction error for a given number of samples in the training dataset. In that regard, reconstruction errors of multilayer neural networks have been studied in detail in [2] for the analytical functions

$$\begin{aligned} &f_A(x_1,\cdots,x_d) = \|\boldsymbol{x}\|^2 = \sum_{i=1}^{d} x_i^2, \quad f_B(x_1,\cdots,x_d) = \exp(-\sum_{i=1}^{d} x_i), \\ &f_C(x_1,\cdots,x_d) = \|\boldsymbol{x}\| \end{aligned} \tag{45}$$

with d=6, 8, 10. We used the proposed machine learning procedure to reconstruct the 3 functions in (45) and compare the obtained reconstruction errors with the results of [2]. The training datasets consisted of M =20 000 samples for all examples at d=6, 50 000 samples for all examples at d=8, 100 000 samples for all examples at d=10. Algorithm 2 was used with $N = 16$ for d=6, $N = 12$ for d=8, $N = 8$ for d=10. Reconstruction errors were evaluated by calculating the Root Mean Squared Error (RMSE) over a random set $\{\tilde{\boldsymbol{x}}^k\}$ of M samples (independent from the samples of the training dataset), e.g.

$$\sqrt{\frac{1}{M}\sum_{k=1}^{M}\left(Tf_A(\tilde{\boldsymbol{x}}^k) - f_A(\tilde{\boldsymbol{x}}^k)\right)^2}$$

where Tf_A denotes the machine learning model of f_A. In Table 1 are reported the reconstruction errors for all the cases considered. In Fig. 7, 8 and 9 are plotted the

exact functions f_A, f_B, f_C and their reconstructions on the line segment $x_1 = \cdots = x_d$.

Table 1: Reconstruction errors

function	d	error (present)	error (neural networks)
f_A	6	$5\ 10^{-15}$	$\simeq 10^{-4}$
	8	$4\,10^{-15}$	$\simeq 2\,10^{-5}$
	10	$4\,10^{-15}$	$\simeq 4\,10^{-6}$
f_B	6	$5\ 10^{-3}$	$\simeq 3\,10^{-4}$
	8	$5\,10^{-2}$	$\simeq 3\,10^{-3}$
	10	$5\,10^{-1}$	$\simeq 2\,10^{-1}$
f_C	6	$3\ 10^{-3}$	$\simeq 4\,10^{-2}$
	8	$7\,10^{-3}$	$\simeq 2\,10^{-2}$
	10	$9\,10^{-3}$	$\simeq 1$

As mentioned earlier, multilayer neural networks have been used in [2] to reconstruct functions f_A, f_B and f_C in (45), testing several network architectures. The best achieved reconstruction errors (for the same number of samples as used with the Fourier-based technique) are reported in the last column of Table 1. For function f_A, we can observe in Table 1 that the proposed machine-learning technique performs better than neural networks. For function f_B, the situation is the opposite. Function f_C has the distinctive feature of being nonsmooth, which make the reconstruction more difficult. The proposed machine-learning procedure gives a better reconstruction error than neural networks, even though the local error at the singularity point is significant. The results of [2] suggest that neural networks give a better local error at the singularity point.

Those examples suggest that overall the proposed machine learning procedure is in the same ballpark as neural networks in terms of reconstruction errors. Interestingly, we can observe that those two techniques perform differently depending on the function considered. Therefore there might be an interest in combining those two techniques, which will be the subject of future work.

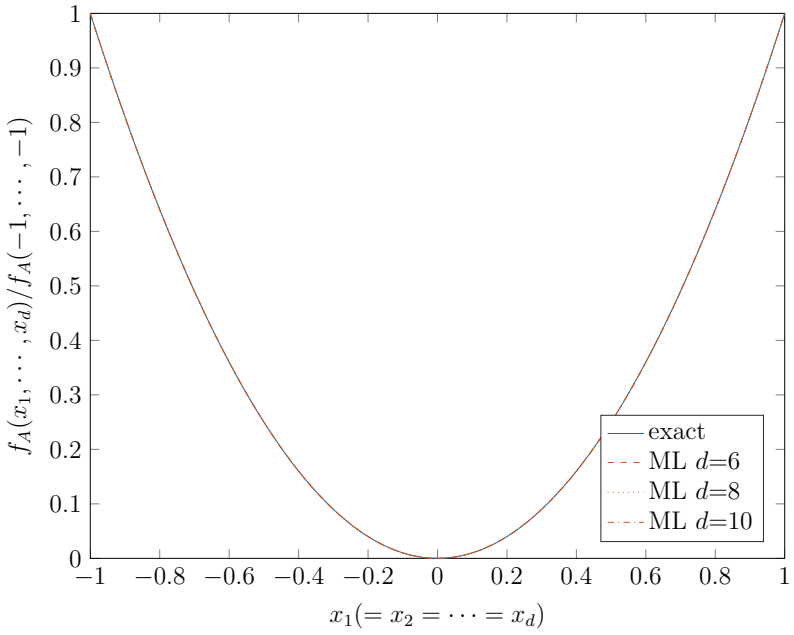

Figure 7: Reconstruction of the function f_A for $x_1 = \cdots = x_d$ with d=6,8,10. The exact function f_A is shown in blue. The red curves show the Machine Learning (ML) models for the several values of the dimension d considered.

In the examples in (45), all the variables x_1,$\cdots$,x_d have the same importance. It is interesting to also consider a function in which the variables have different nonlinearities. A simple example is

$$f_D(x_1, \cdots, x_d) = \sum_{i=1}^{d} x_i^i \tag{46}$$

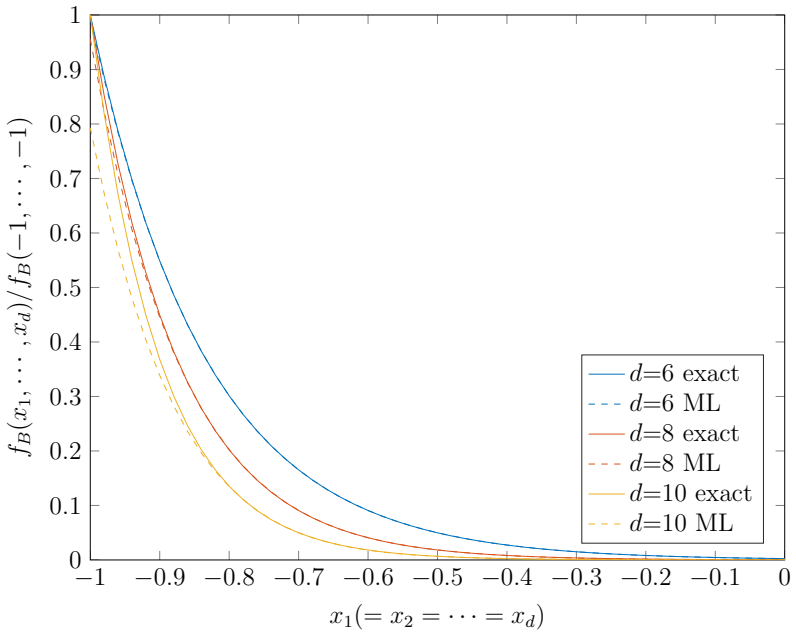

Figure 8: Reconstruction of the function f_B for $x_1 = \cdots = x_d$ with d=6,8,10. The exact function f_B is shown in solid lines for the several values of d considered. The Machine Learning (ML) models are shown in dashed lines.

In Fig. 10 are plotted the values taken by the exact function f_D and its reconstruction on the line segment $x_1 = \cdots = x_d$, in the case $d = 8$, $M = 50000$, $N = 12$. The reconstruction RMSE is 7 10^{-4}. Let us use the example (46) to discuss the influence of the number of samples M and the rank N on the reconstruction error. The curves in Fig. 11 show the reconstruction RMSE as a function of M, for several values of N. For $N = 12$ and $N = 16$, we can observe that the RMSE is quite high for $M \leq 5000$: This corresponds to situations where the number of samples is too small compared to the number of Fourier coefficients in the partial sum considered for the reconstruction to be reliable. Each curve displays a decreasing behavior for M large enough, meaning that increasing the number of samples tends to improve the accuracy of the reconstruction. Also observe that the RMSE seems to reach a limit as

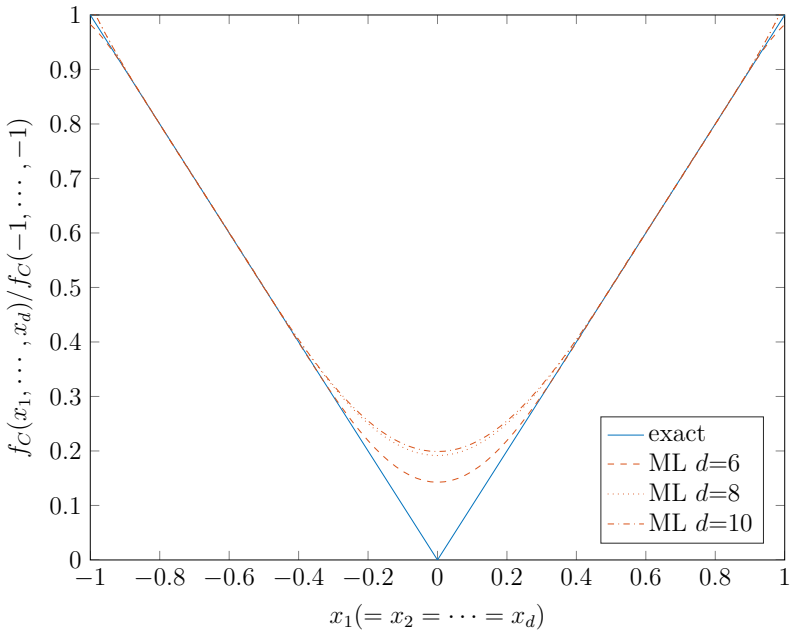

Figure 9: Reconstruction of the function f_C for $x_1 = \cdots = x_d$ with d=6,8,10. The exact function f_C is shown in blue. The red curves show the Machine Learning (ML) models for the several values of the dimension d considered.

M becomes very large: This corresponds to the limit situation where all the Fourier coefficients in the partial sum considered are calculated exactly. That limit RMSE decreases as N increases (as could be expected). However, the convergence towards that limit value gets slower as N increases. In the present case, 50 000 samples seem to be enough to reach the limit RMSE for $N = 8$ and $N = 12$.

6. Application to a nonlinear homogenization problem

In this Section, we discuss the application of the proposed machine learning procedure to a nonlinear homogenization problem in solid mechanics. Consider a two-dimensional inhomogeneous electric conductor occupying a domain Ω (one could alternatively consider thermal conductivity, magnetic permeability or diffusion since

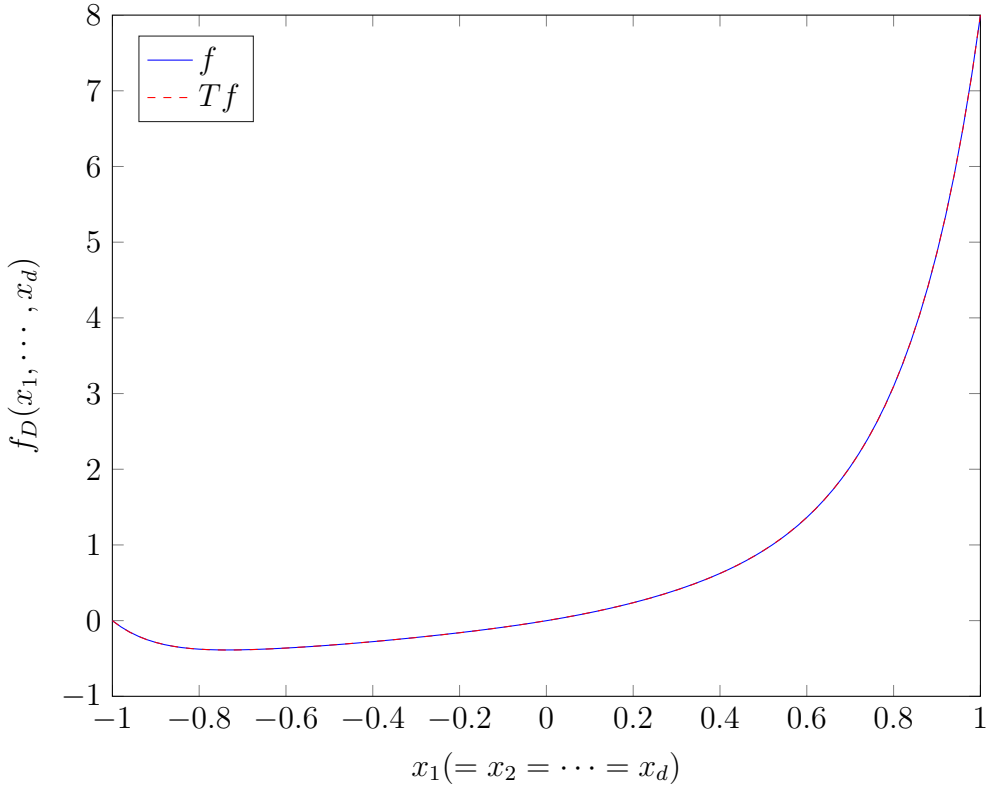

Figure 10: Reconstruction of the function f_D for $x_1 = \cdots = x_d$ ($d = 8$) The exact function f_D is shown in blue.

all those phenomenons are governed by the same equations). The electric field $\boldsymbol{e}$ and the current density $\boldsymbol{j}$ are related by the local constitutive law

$$\boldsymbol{j} = \frac{\partial w}{\partial \boldsymbol{e}}(\boldsymbol{e}, \boldsymbol{x}) \tag{47}$$

where the convex energy-density function w depends on the location $\boldsymbol{x}$. Denoting by $\bar{\boldsymbol{e}}$ (resp. $\bar{\boldsymbol{j}}$) the spatial average of $\boldsymbol{e}$ (resp. $\boldsymbol{j}$), the effective constitutive law reads as [15, 16]

$$\bar{\boldsymbol{j}} = \frac{dw_{\mathit{eff}}}{d\bar{\boldsymbol{e}}}(\bar{\boldsymbol{e}}) \tag{48}$$

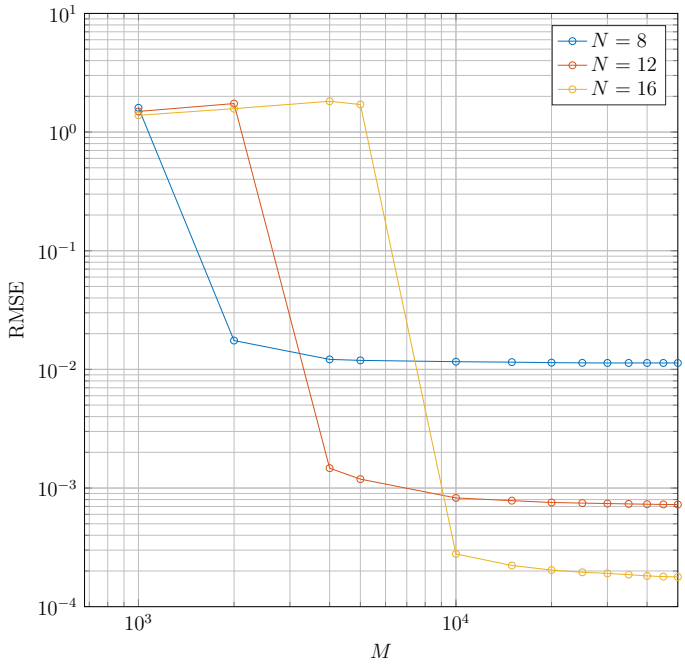

Figure 11: Influence of M and N on the reconstruction error (function f_D with $d = 8$).

where w_{eff} is the effective energy function of the composite material, defined by

$$w_{\mathit{eff}}(\bar{\boldsymbol{e}}) = \inf_{\boldsymbol{e} \in \mathcal{K}(\bar{\boldsymbol{e}})} \frac{1}{|\Omega|} \int_{\Omega} w(\boldsymbol{e}, \boldsymbol{x}) d\omega. \tag{49}$$

In (49), $\mathcal{K}(\bar{\boldsymbol{e}})$ is the set of admissible electric fields, as defined by

$$\mathcal{K}(\bar{\boldsymbol{e}}) = \{\boldsymbol{e} : \Omega \mapsto \mathbb{R}^2 | \boldsymbol{e} = \nabla V \text{ for some } V : \Omega \mapsto \mathbb{R} \text{ verifying } V(\boldsymbol{x}) = \bar{\boldsymbol{e}}.\boldsymbol{x} \text{ on } \partial\Omega\}.$$

Determining $w_{\mathit{eff}}(\bar{\boldsymbol{e}})$ formally amounts to solve the set of local equations

$$\operatorname{div} \boldsymbol{j} = 0, \ \boldsymbol{j} = \frac{\partial w}{\partial \boldsymbol{e}}(\boldsymbol{e}, \boldsymbol{x}), \ \boldsymbol{e} = \nabla V \text{ in } \Omega, \ V(\boldsymbol{x}) = \bar{\boldsymbol{e}}.\boldsymbol{x} \text{ on } \partial\Omega \tag{50}$$

where the field V is the electric potential.

For a $n-$phase composite, the energy-density function w in (47) specializes as

$$w(\boldsymbol{e}, \boldsymbol{x}) = \sum_{r=1}^{n} \chi_r(\boldsymbol{x}) w_r(\boldsymbol{e}) \tag{51}$$

where χ_r is the characteristic function of phase r ($\chi_r(\boldsymbol{x}) = 1$ is $\boldsymbol{x}$ is in phase r, $\chi_r(\boldsymbol{x}) = 0$ otherwise) and w_r is the energy-density function of the constitutive material in phase r. Assuming that the energy functions w_r are known, the effective energy w_{eff} depends on the *microstructure* of the composite, i.e. on the geometrical arrangement of the phases. Various bounds and estimates have been proposed for obtaining some information on w_{eff} when only limited information on the microstructure is available. For linear composites with statically isotropic microstructures, Hashin and Shtrikman [17] derived optimal lower and upper bounds on the effective energy. Several approaches have been proposed to extend the Hashin-Shtrikman bounds to nonlinear composites [18–21]. Here we consider the Hashin-Shtrikman-type lower bound derived in [22], which is obtained by combining the so-called translation method [23] with the idea of embedding the original problem in a problem of higher dimension. In more detail, extended fields $\boldsymbol{E}(\boldsymbol{x}) = (\boldsymbol{e}_1(\boldsymbol{x}), \boldsymbol{e}_2, \boldsymbol{e}_3(\boldsymbol{x}))$ are introduced by considering 3 electric fields $\boldsymbol{e}_1(\boldsymbol{x})$, $\boldsymbol{e}_2(\boldsymbol{x})$ and $\boldsymbol{e}_3(\boldsymbol{x})$ written side by side. Extended fields $\boldsymbol{E} = (\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3)$ can be represented by 2×3 matrices, i.e.

$$\boldsymbol{E} = \begin{pmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{pmatrix}$$

where u_i and v_i are the components of the electric field $\boldsymbol{e_i}$ in a reference basis of $\mathbb{R}^2$. Correspondingly are introduced the *extended energy* $W_r(\boldsymbol{E}) = w_r(\boldsymbol{e}_1, \boldsymbol{x}) + w_r(\boldsymbol{e}_2) + w_r(\boldsymbol{e}_3)$ of phase r, as well as the *extended effective energy*

$$W_{\mathit{eff}}(\bar{\boldsymbol{E}}) = w_{\mathit{eff}}(\bar{\boldsymbol{e}}_1) + w_{\mathit{eff}}(\bar{\boldsymbol{e}}_2) + w_{\mathit{eff}}(\bar{\boldsymbol{e}}_3), \tag{52}$$

where $\bar{\boldsymbol{E}} = (\bar{\boldsymbol{e}}_1, \bar{\boldsymbol{e}}_2, \bar{\boldsymbol{e}}_3)$. For any $\boldsymbol{T} \in \mathbb{R}^{2\times 3}$ and any quasiconvex[2] function $U(\boldsymbol{E})$ it

[2] i.e such that $|\Omega| U(\bar{\boldsymbol{E}}) \leq \int_\Omega U(\boldsymbol{E}(\boldsymbol{x}))d\omega$ for any $\boldsymbol{E}$ verifying $\boldsymbol{e}_i \in \mathcal{K}(\bar{\boldsymbol{e}}_i)$ $(i = 1, 2, 3)$

can be proved [22] that

$$W_{\mathit{eff}}(\bar{\boldsymbol{E}}) \geq \bar{\boldsymbol{E}} \cdot \boldsymbol{T} + U(\bar{\boldsymbol{E}}) - \sum_{r=1}^{n} c_r f_r \tag{53}$$

where

$$f_r = \sup_{\boldsymbol{E}} \boldsymbol{E} \cdot \boldsymbol{T} + U(\boldsymbol{E}) - W_r(\boldsymbol{E}) \tag{54}$$

and c_r is the volume fraction of phase r. In particular, consider $\bar{\boldsymbol{E}}$ of the form $\bar{\boldsymbol{E}} = |\bar{\boldsymbol{e}}|\boldsymbol{N}$ with

$$\boldsymbol{N} = \frac{1}{2}\begin{pmatrix} 1 & -1 & -2 \\ -\sqrt{3} & \sqrt{3} & 0 \end{pmatrix}$$

i.e. the 3 loading directions $\bar{\boldsymbol{e}}_1$, $\bar{\boldsymbol{e}}_2$ and $\bar{\boldsymbol{e}}_3$ have the same norm and make a $\pi/3$ angle (modulo π) between each other. For an isotropic composite, the effective energy w_{eff} only depend on $\bar{\boldsymbol{e}}$ through its norm $\|\bar{\boldsymbol{e}}\|$, hence $W_{\mathit{eff}}(|\bar{\boldsymbol{e}}|\boldsymbol{N}) = 3w_{\mathit{eff}}(\bar{\boldsymbol{e}})$. Eq. (53) thus delivers a general lower bound on w_{eff}. The trick is to choose $\boldsymbol{T}$ and the function U so that the corresponding lower bound is meaningful. The choice explored in [22] consists in taking $\boldsymbol{T}$ of the form $\tau\boldsymbol{N}$ and $U(\boldsymbol{E}) = \alpha(\det(\boldsymbol{e}_1, \boldsymbol{e}_2) + \det(\boldsymbol{e}_2, \boldsymbol{e}_3) + \det(\boldsymbol{e}_1, \boldsymbol{e}_3))$ where (τ, α) are arbitrary scalar parameters. Eqs (53) and (54) specialize as

$$w_{\mathit{eff}}(\bar{\boldsymbol{e}}) \geq \|\bar{\boldsymbol{e}}\|\tau - \frac{\sqrt{3}}{2}\alpha\|\bar{\boldsymbol{e}}\|^2 - \frac{1}{3}\sum_{r=1}^{n} c_r f_r(\alpha, \tau) \tag{55}$$

with

$$f_r(\alpha, \tau) = \sup_{(u_i, v_i)_{1\leq i\leq 3}} \frac{\tau}{2}(u_1 - \sqrt{3}v_1 - u_2 - \sqrt{3}v_2 - 2u_3) - \sum_{i=1}^{3} w_r(u_i, v_i) + \alpha \sum_{i,j=1, i<j}^{3} (u_i v_j - u_j v_i). \tag{56}$$

Eq. (55) holds for any parameters (α, τ). The best bound w^{-}_{eff} is obtained by maximizing the right hand side of (55) with respect to (α, τ), i.e. we have

$$w^{-}_{\mathit{eff}}(\bar{\boldsymbol{e}}) = \sup_{\alpha\in\mathbb{R}, \tau\in\mathbb{R}} \left\{ \|\bar{\boldsymbol{e}}\|\tau - \frac{\sqrt{3}}{2}\alpha\|\bar{\boldsymbol{e}}\|^2 - \frac{1}{3}\sum_{r=1}^{n} c_r f_r(\alpha, \tau) \right\}. \tag{57}$$

The bound w^-_{eff} in (57) is the best available to date. One downside, however, is that the numerical evaluation of w^-_{eff} is relatively tricky. Observe indeed from (57) and (56) that w^-_{eff} is defined by nested nonlinear maximization problems. Moreover, the 6-dimensional maximization problem defining f_r in (56) is not concave (because of the term $u_i v_j - u_j v_i$). Special caution must thus be taken when solving (56) in order to avoid converging towards a local stationary point that does not coincide with a global maximum as needed (a strategy for avoiding such a pitfall is detailed in [22]). For those reasons, the numerical evaluation of w^-_{eff} requires special care and may become time-consuming when many evaluations are needed. In such circumstances, machine learning techniques are relevant: Once the training stage is complete, they allow one to have almost instant access to numerical values of w^-_{eff}.

In the following, we use the Fourier-based approach for learning w^-_{eff} in the case of nonlinear composites whose constitutive materials have a power-law type behavior, i.e. the energy function w_r of phase r is of the form

$$w_r(\boldsymbol{e}) = \frac{\sigma_r}{m_r + 2} \|\boldsymbol{e}\|^{m_r+2} \tag{58}$$

where $m_r > 0$ is a nonlinearity index and $\sigma_r > 0$ is a nonlinear conductivity parameter. Let us evaluate the dimensionality of the problem. For a $n-$phase composite with power-law constitutive materials, the bound w^-_{eff} in (57) is a function of the $3n+1$ scalar parameters $(\|\bar{\boldsymbol{e}}\|, \sigma_1, m_1, c_1, \cdots, \sigma_n, m_n, c_n)$. Those parameters are not independent since the volume fractions $\{c_r\}$ need to satisfy the equality $\sum c_r = 1$. Moreover, we can always normalize all the energy functions with respect to σ_1 (for instance), which amounts to assuming that $\sigma_1 = 1$. The effective dimensionality of the problem is thus equal to $3n - 1$.

We used the presented technique to learn the energy function w^-_{eff} for $2-$ and

3−phase composites, corresponding to dimensionality equal to 5 and 8, respectively. The machine learning models of w^-_{eff} for 2− and 3−phase composites are denoted by $T_2 w_{\mathit{eff}}$ and $T_3 w_{\mathit{eff}}$, respectively.

In order to deal with dimensionless parameters, it is convenient to introduce a given reference value e_0 of the magnitude of the electric field. For 2-phase composites, the 5 dimensionless parameters $(\|\bar{\boldsymbol{e}}\|/e_0, c_1, m_1, m_2, \sigma_2/\sigma_1)$ were restricted to the domain $[0,1]^2 \times [0,5]^2 \times [1,10]$. That domain is mapped into the unit hypercube of $\mathbb{R}^5$ by defining $(x_1, \cdots, x_5)$ as

$$(x_1, \cdots, x_5) = \left(\frac{\|\bar{\boldsymbol{e}}\|}{e_0}, c_1, \frac{m_1}{5}, \frac{m_2}{5}, \frac{\sigma_2/\sigma_1 - 1}{9} \right).$$

A Sobol sequence of 80 000 samples was generated to build the training database. The machine learning model $T_2 w_{\mathit{eff}}$ was constructed using a Matlab implementation of Algorithm 2 with $N = 32$ (resulting in 8603 Fourier coefficients). The reconstruction error, as evaluated by the relative Root Mean Square Error (rRMSE)

$$\frac{\sqrt{\sum_{k=1}^{\tilde{M}} \left(w^-_{\mathit{eff}}(\tilde{\boldsymbol{x}}^k) - T_2 w_{\mathit{eff}}(\tilde{\boldsymbol{x}}^k) \right)^2}}{\sqrt{\sum_{k=1}^{\tilde{M}} w^-_{\mathit{eff}}(\tilde{\boldsymbol{x}}^k)^2}} \tag{59}$$

on a test database of $\tilde{M}$=10 000 randomly generated samples $\{\tilde{\boldsymbol{x}}^k\}$, was 0.16%.

For 3-phase composites, it is convenient to parameterize the volume fractions using the parameter $\theta = c_2/(1 - c_1)$ so that

$$c_2 = \theta(1 - c_1), \;\; c_3 = (1 - \theta)(1 - c_1) \tag{60}$$

where θ and c_1 can take any value in $[0, 1]$. The parameterization (60) allows one to automatically satisfy the constraint $c_1 + c_2 + c_3 = 1$. The 8 dimensionless parameters

$(\|\bar{\boldsymbol{e}}\|/e_0, c_1, \theta, m_1, m_2, m_3, \sigma_2/\sigma_1, \sigma_3/\sigma_1)$ were restricted to the domain $[0,1]^3 \times [0,5]^3 \times [1,10]^2$. That domain is mapped into the unit hypercube of $\mathbb{R}^8$ by using the linear mapping

$$(x_1, \cdots, x_8) = \left(\frac{\|\bar{\boldsymbol{e}}\|}{e_0}, c_1, \theta, \frac{m_1}{5}, \frac{m_2}{5}, \frac{m_3}{5}, \frac{\sigma_2/\sigma_1 - 1}{10}, \frac{\sigma_3/\sigma_1 - 1}{10} \right) \tag{61}$$

A Sobol sequence of 1 000 000 samples in the unit hypercube was generated to build the training database. The machine learning model $T_3 w_{\mathit{eff}}$ was generated using the value $N = 18$ in Algorithm 2. The relative error, as evaluated on a random test database with a formula similar to (59), was 0.54%.

Using the obtained machine learning models $T_2 w_{\mathit{eff}}$ and $T_3 w_{\mathit{eff}}$, a global sensitivity analysis can first be performed to get some insight in the behavior of the target function w^-_{eff} for $2-$ and $3-$phase composites. In Fig 12 are shown the sensitivity indices for the machine learning model $T_2 w_{\mathit{eff}}$ corresponding to $2-$phase composites. Those sensitivities have been calculated using expressions (41) and (42) for each of the 5 variables $\|\bar{\boldsymbol{e}}\|/e_0$, c_1, m_1, m_2, σ_2/σ_1. It is insightful to compare those results with the sensitivity indices corresponding to the elementary Voigt bound w^+_{eff}, which for a $n-$phase composite is given by

$$w^+_{\mathit{eff}} = \sum_{r=1}^{n} c_r w_r(\bar{\boldsymbol{e}}) \tag{62}$$

Expression (62) is an upper bound on the effective energy w_{eff}. It applies to any microstructure, without any assumption of isotropy. Expression (49) of w^+_{eff} is simple enough for the corresponding sensitivity indices to be calculated in closed-form. They are shown in red in Fig. 12. Comparing the sensitivity indices of $T_2 w_{\mathit{eff}}$ and w^+_{eff}, we can observe that in both cases $\|\bar{\boldsymbol{e}}\|/e_0$ is the primary parameter that has the most influence. A similar observation was done in [3] for elasticity. The influence of the other parameters c_1, σ_2/σ_1, m_1, m_2 is shown more clearly in Fig. 12(right). For the

upper bound w^+_{eff}, the volume fraction c_1 dominates the parameters σ_2/σ_1, m_1, m_2. Compared to w^+_{eff}, the function $T_2 w_{\mathit{eff}}$ gives more weight to the parameters m_2 and σ_2/σ_1. In particular, we can observe that m_2 becomes the dominating parameter, instead of c_1 for the function w^+_{eff}. Those results can be interpreted as a consequence of the assumption of isotropy which is taken into account in the Hashin–Shtrikman-type bound w^-_{eff}. Isotropy tends to reduce the influence of the volume fraction c_1 in favor of the other microstructural parameters.

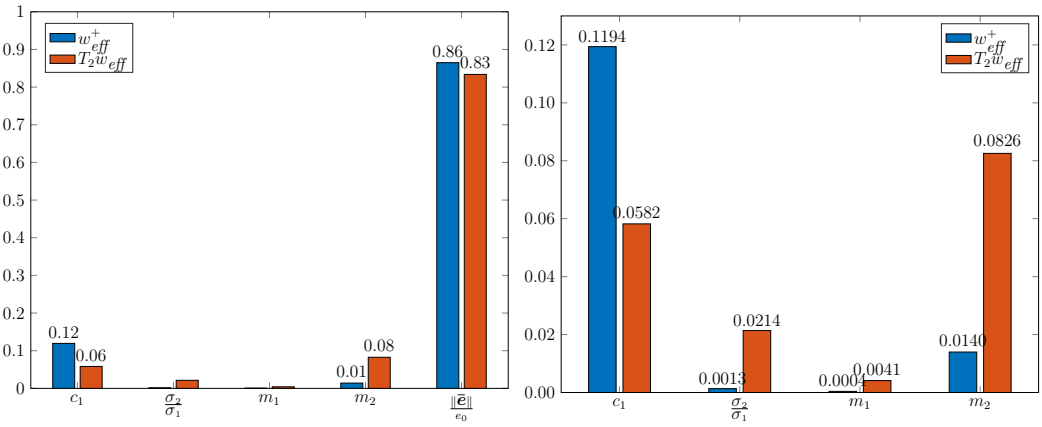

Figure 12: Sensitivity indices of the energy functions w^+_{eff} and $T_2 w_{\mathit{eff}}$ for 2-phase composites with power-law constitutive materials.

For 3-phase composites, $\|\bar{\boldsymbol{e}}\|/e_0$ is again found to be the most influential parameter, with a sensitivity index close to 0.85 for both the energy functions $T_3 w_{\mathit{eff}}$ and w^+_{eff}. The sensitivity indices of the 7 remaining parameters $(c_1, \theta, m_1, m_2, m_3, \sigma_2/\sigma_1, \sigma_3/\sigma_1)$ are shown in Fig. 13. For the energy function w^+_{eff}, the volume fraction parameters c_1 and θ dominate the other ones. For the energy function $T_3 w_{\mathit{eff}}$, the influence of c_1 and θ is mitigated in favor of the other parameters (mainly the nonlinearity indices m_2 and m_3). Also observe that the sensitivity indices with respect to m_2 and m_3 are equal. Similarly, the sensitivity indices with respect to the conductivity contrasts

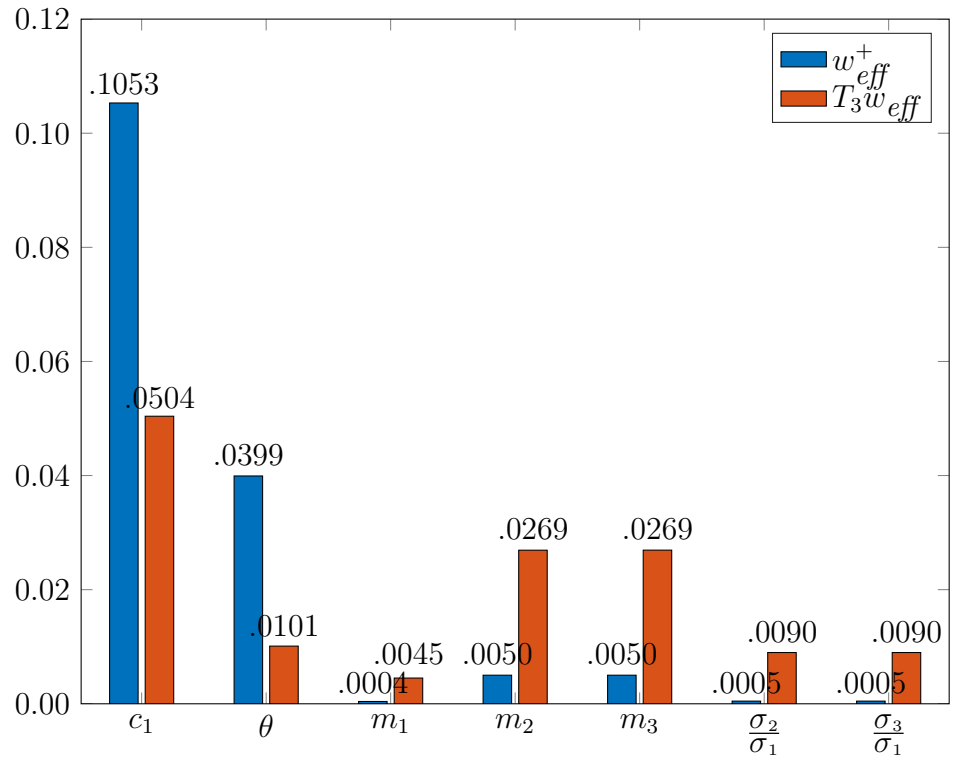

Figure 13: Sensitivity indices of the energy functions w^+_{eff} and $T_3 w_{eff}$ for 3-phase composites with power-law constitutive materials.

σ_2/σ_1 and σ_3/σ_1 are equal. This results from the fact that phases 2 and 3 play a symmetrical role.

6.1. Effective energies of microstructures

A possible application of the obtained machine learning models consists in testing specific microstructures against the bounds. In Fig. 14(left) is represented a two-phase matrix/inclusions microstructure that consists of 58 circular inclusions. Dimensionless values are used for lengths, conductivities and electric fields. The domain Ω is chosen as the unit disk. The matrix phase (phase 1) is occupied by a power-law material with conductivity $\sigma_1 = 1$ and nonlinearity index m_1. The in-

clusions (phase 2) are occupied by a power-law material with conductivity $\sigma_2 = 10$ and nonlinearity index m_2. We limit our attention to the case $m_1 = m_2$. The blue solid curve in Fig. 14 shows the values of the effective energy $w_{\mathit{eff}}(\bar{\boldsymbol{e}})$ of the considered microstructure for $\|\bar{\boldsymbol{e}}\| = 1$ and several values of the nonlinearity index m_1. Those values were obtained numerically by solving (50) with the finite-element code Freefem [24]. In a preliminary stage, quasi-isotropy of the microstructure in Fig. 14 was checked by calculating $w_{\mathit{eff}}(\bar{\boldsymbol{e}})$ for several values of $\bar{\boldsymbol{e}}$ and verifying that $w_{\mathit{eff}}(\bar{\boldsymbol{e}})$ only depends on $\bar{\boldsymbol{e}}$ through the norm $\|\bar{\boldsymbol{e}}\|$, i.e. does not vary with the direction of $\bar{\boldsymbol{e}}$ (the relative variation was found to remain below 1%). In such conditions, the energy w_{eff} needs to be above the Hashin-Shtrikman type lower bound w^-_{eff}. The solid red line in Fig. 14 shows the values taken by the lower bound as provided by the machine learning model $T_2 w_{\mathit{eff}}$. We observe that $w_{\mathit{eff}} \geq T_2 w_{\mathit{eff}}$, as expected. In more detail, we can observe that the gap $w_{\mathit{eff}} - T_2 w_{\mathit{eff}}$ is approximatively independent of the nonlinearity index m_1. Such type of comparison can be useful for assessing the optimality of specific microstructures, i.e. measuring the gap with the lower bound. As a validation of the machine learning model, the bound w^-_{eff} for the values of $(\|\bar{\boldsymbol{e}}\|, c_1, m_1, m_2, \sigma_2)$ considered in Fig. 14 has been calculated by directly solving (57). The results are shown as a green dotted line in Fig. 14. The rRMSE with the values provided by $T_2 w_{\mathit{eff}}$ is about 0.19%, which illustrates the fact that $T_2 w_{\mathit{eff}}$ delivers an accurate approximation of w^-_{eff}. The benefit of using $T_2 w_{\mathit{eff}}$ instead of w^-_{eff} lies in the computational cost: the average time of a numerical evaluation of w^-_{eff} is about 2.6 s whereas the average time of a numerical evaluation of $T_2 w_{\mathit{eff}}$ is about 0.001 s (all the reported numerical simulations were performed on a mid-range workstation equipped with an Intel i7-8700@3.2 GHz CPU).

The example considered can also be used to validate the machine learning model $T_3 w_{\mathit{eff}}$. The two-phase microstructure shown in Fig. 14(left) can indeed viewed as a

special case of a 3−phase microstructure in which two phases (say phases 2 and 3) are governed by the same energy function (i.e. $w_2 = w_3$). The solid orange line in Fig. 14 shows the values of $T_3 w_{\mathit{eff}}$ obtained using the same material parameters for phases 2 and 3, i.e. $m_2 = m_3 (= m_1)$, $\sigma_2 = \sigma_3 = 10$. The rRMSE with the values provided by w^-_{eff} is about 0.74%. This is slightly higher than the error obtained with the machine learning model $T_2 w_{\mathit{eff}}$, which could be expected since $T_3 w_{\mathit{eff}}$ lives in a higher dimensional space. The advantage of $T_3 w_{\mathit{eff}}$ is that it allows one to do energy comparisons for 3-phase microstructures. As an example, consider the microstructure shown in Fig. 15(left). That microstructure consists of two families of circular inclusions in equal volume fractions (set to 0.25). One family of inclusions (phase 2) is occupied by a material with conductivity $\sigma_2 = 5$, the other one (phase 3) is occupied by a material with conductivity $\sigma_3 = 10$. The matrix (phase 1) has a volume fraction set to 0.5 and a conductivity $\sigma_1 = 1$. The nonlinearity index m is the same in all 3 phases. The values of $T_3 w_{\mathit{eff}}$ and w^-_{eff} for the considered 3−phase microstructure are shown in Fig. 15. The blue curve shows the effective energy $w_{\mathit{eff}}(\bar{\boldsymbol{e}})$ of the considered microstructure for $\|\bar{\boldsymbol{e}}\| = 1$, as obtained from finite element simulations with several values of m. The red curve shows the values of the lower bound as predicted by the machine learning model $T_3 w_{\mathit{eff}}$. Again we can observe that the gap $w_{\mathit{eff}} - T_3 w_{\mathit{eff}}$ does not vary significantly with m. The relative gap $(w_{\mathit{eff}} - T_3 w_{\mathit{eff}})/T_3 w_{\mathit{eff}}$ is smaller than the gap obtained for the 2-phase microstructure considered previously, which probably results from the smaller contrast between the material conductivities in the 3-phase example considered.

6.2. Boundary-value problem

This next example illustrates how the Fourier-based machine learning model can be utilized for solving a Boundary Value Problem (BVP). Consider a composite

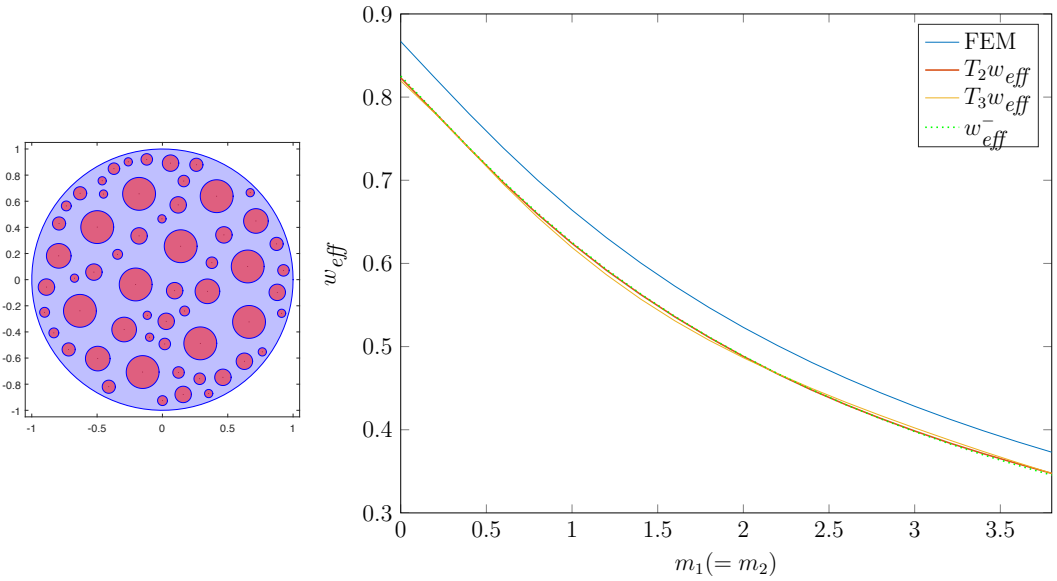

Figure 14: Effective energy of a 2-phase microstructure: Bounds and FEM simulations.

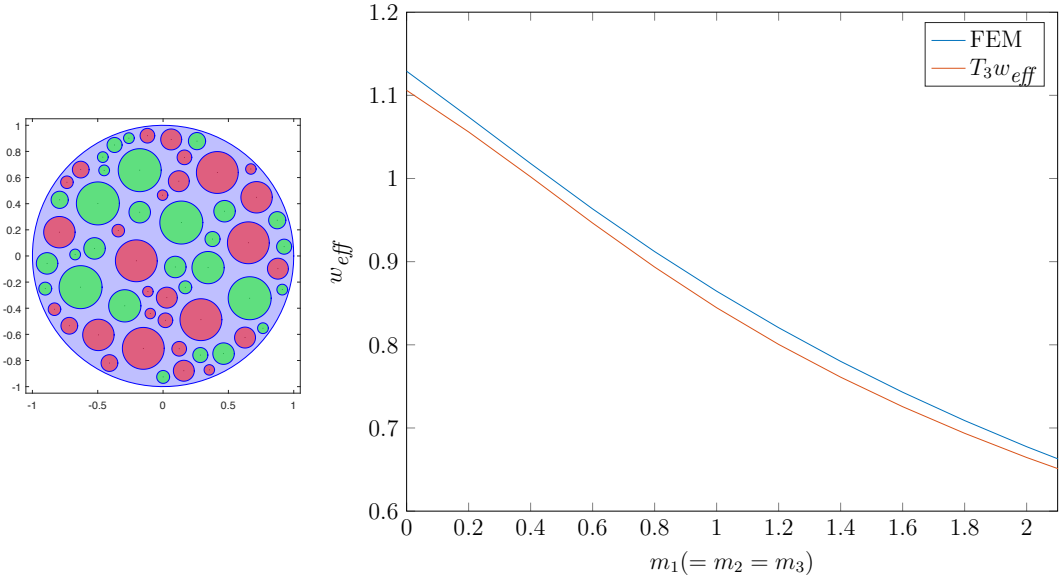

Figure 15: Effective energy of a 3-phase microstructure: Bounds and FEM simulations.

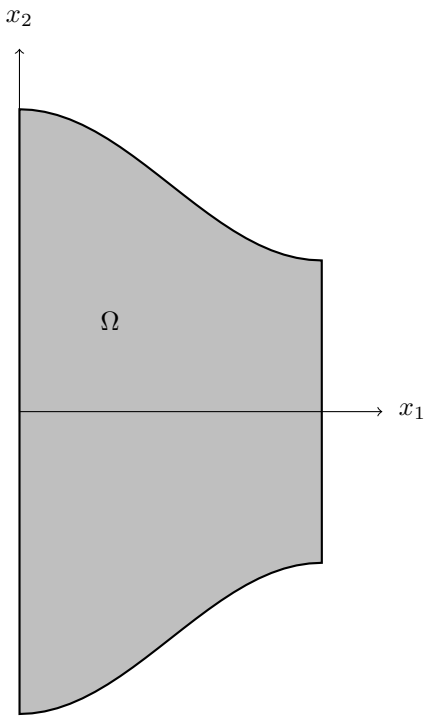

Figure 16: Domain Ω in the considered boundary value problem.

conductor occupying the 2-dimensional domain Ω shown in Fig. 16. Dimensionless values are used for lengths, conductivities and electric fields. The voltage on the left border $x_1 = 0$ is set to 0 and a prescribed voltage V_0 is applied on the right border $x_1 = 1$. The top and bottom borders of Ω are insulated. The voltage $V(\boldsymbol{x})$, electric field $\boldsymbol{e}(\boldsymbol{x})$ and current density $\boldsymbol{j}(\boldsymbol{x})$ in Ω are obtained by solving the BVP

$$\boldsymbol{e} \in \mathcal{K}_e,\ \boldsymbol{j} \in \mathcal{K}_j,\ \boldsymbol{j}(\boldsymbol{x}) = \sum_{r=1}^{n} \chi_r w_r'(\boldsymbol{e}(\boldsymbol{x})) \tag{63}$$

where

$$\mathcal{K}_e = \left\{\boldsymbol{e} : \boldsymbol{e} = \nabla V \text{ for some } V(\boldsymbol{x}) \text{ such that } V(\boldsymbol{x}) = V_0 x_1 \text{ if } x_1 \in \{0,1\}\right\}$$
$$\mathcal{K}_j = \left\{\boldsymbol{j} : \operatorname{div} \boldsymbol{j} = 0 \text{ in } \Omega, \boldsymbol{j} \cdot \boldsymbol{n} = 0 \text{ for } \boldsymbol{x} \in \partial\Omega, x_1 \notin \{0,1\}\right\}$$

By definition, the homogenized energy w_{eff} in (49) gives the effective energy of the composite material at a mesoscopic scale that is much larger than the length scale of the microstructure. If in turn that mesoscopic scale is much smaller than the length scale of Ω, then problem (63) can be replaced by the BVP

$$\boldsymbol{e} \in \mathcal{K}_e,\ \boldsymbol{j} \in \mathcal{K}_j,\ \boldsymbol{j}(\boldsymbol{x}) = \frac{\partial w_{\mathit{eff}}}{\partial \boldsymbol{e}}(\boldsymbol{e}(\boldsymbol{x})) \tag{64}$$

where V, $\boldsymbol{e}$ and $\boldsymbol{j}$ are to be understood as mesoscopic quantities. For a composite material with a statistically uniform microstructure, the *heterogeneous* energy function in (63) is thus replaced in (64) by an effective *uniform* energy function. The main obstacle in using (64), however, is that the exact expression of w_{eff} is generally not available. A possible approach for solving the problem is to use 'FE^2'-type methods, in which the mesoscopic constitutive relation at each point is obtained by solving a BVP at the microscopic scale [25–27]. Such methods offer a lot of flexibility but are generally computationally expensive. Here we choose to replace w_{eff} in (64) by an estimate that does not require solving any BVP at the microscopic scale. The bound w^-_{eff} can be used as such an estimate, leading to the problem

$$\boldsymbol{e} \in \mathcal{K}_e,\ \boldsymbol{j} \in \mathcal{K}_j,\ \boldsymbol{j}(\boldsymbol{x}) = \frac{\partial w^-_{\mathit{eff}}}{\partial \boldsymbol{e}}(\boldsymbol{e}(\boldsymbol{x})). \tag{65}$$

Solving (65) remains extremely time intensive because many evaluations of w^-_{eff} are needed. Problem (65) is indeed nonlinear and typically solved by FEM using a Newton algorithm. This requires to evaluate the function w^-_{eff} at each Gauss point and at each iteration. Since each evaluation of w^-_{eff} entails solving the nested optimization problems in (57), the computational cost of solving (65) quickly gets very high. Such difficulties can be avoided by replacing w^-_{eff} in (65) by a machine-learning model(say $T_3 w_{\mathit{eff}}$ to fix ideas), i.e. we consider the problem

$$\boldsymbol{e} \in \mathcal{K}_e,\ \boldsymbol{j} \in \mathcal{K}_j,\ \boldsymbol{j}(\boldsymbol{x}) = \frac{\partial T_3 w_{\mathit{eff}}}{\partial \boldsymbol{e}}(\boldsymbol{e}(\boldsymbol{x})). \tag{66}$$

Solving (65) by FEM (with a Newton algorithm) requires to evaluate both the first and second derivative of $T_3 w_{\mathit{eff}}$ with respect to $\boldsymbol{e}$. In that regard, we note from (40) that

$$\boldsymbol{j} = A\frac{\boldsymbol{e}}{\|\boldsymbol{e}\|} \tag{67}$$

where

$$A = -\sum_{j=1}^{J} \pi n_1^j C_j \psi'(\|\boldsymbol{e}\|) h(\tilde{\boldsymbol{n}}^j, \Psi(\tilde{\boldsymbol{x}})) \sin \pi n_1^j \psi(\|\boldsymbol{e}\|)$$

and $\tilde{\boldsymbol{x}} = (x_2, \cdots, x_8)$ with x_j defined as in (61). Differentiating (67) yields

$$\frac{\partial \boldsymbol{j}}{\partial \boldsymbol{e}} = \frac{A}{\|\boldsymbol{e}\|}\mathbb{I} + \left(B - \frac{A}{\|\boldsymbol{e}\|}\right)\frac{\boldsymbol{e}}{\|\boldsymbol{e}\|} \otimes \frac{\boldsymbol{e}}{\|\boldsymbol{e}\|} \tag{68}$$

with

$$B = \sum_{j=1}^{J} \pi n_1^j C_j h(\tilde{\boldsymbol{n}}^j, \Psi(\tilde{\boldsymbol{x}})) \left(\psi''(\|\boldsymbol{e}\|) \sin \pi n_1^j \psi(\|\boldsymbol{e}\|) + \pi n_1^j (\psi'(\|\boldsymbol{e}\|))^2 \cos \pi n_1^j \psi(\|\boldsymbol{e}\|)\right)$$

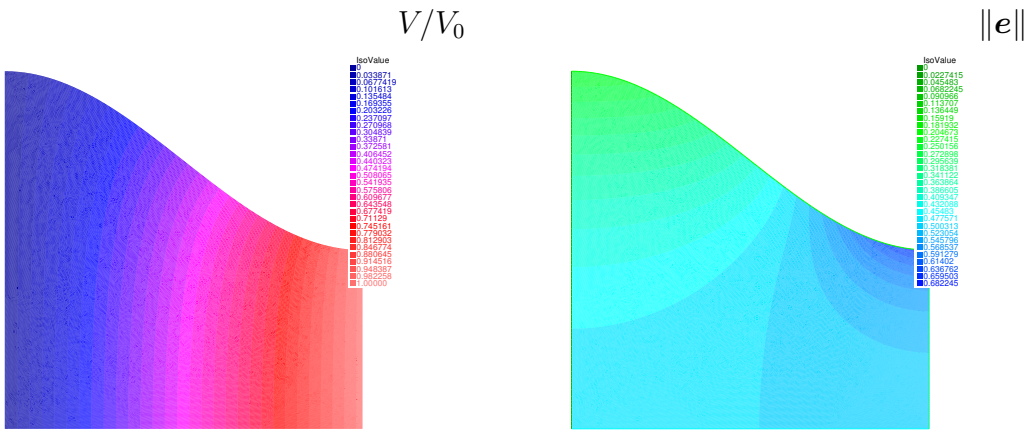

Figure 17: FEM simulations of the electric potential (left) and electric field (right) using the energy function $T_3 w_{\mathit{eff}}$ with parameters $\sigma_1 = \sigma_2 = \sigma_3 = 8$, $m_1 = m_2 = m_3 = 3$, $c_1 = c_2 = c_3 = 1/3$. Because of symmetry, only the upper half of Ω is shown.

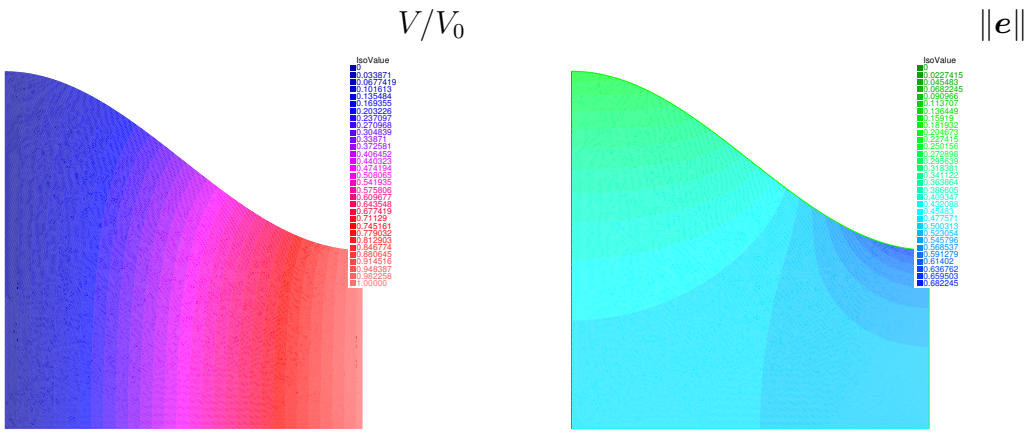

Figure 18: FEM simulations of the electric potential (left) and electric field (right) for a homogeneous material with energy $w(\boldsymbol{e}) = 2\|\boldsymbol{e}\|^4$. Because of symmetry, only the upper half of Ω is shown.

Pb (66) is an approximation of (65). To check the accuracy of that approximation, it seems natural to consider an example for which the exact value of w^-_{eff} is known. The simplest case is that of constitutive phases with the same energy functions w_r, so that $w_{\mathit{eff}} = w^-_{\mathit{eff}} = w_1 = \cdots = w_n$. In Fig. 17 are represented the electric potential V and the electric field $\boldsymbol{e}$ obtained by solving (66) in the case $\sigma_1 = \sigma_2 = \sigma_3 = 8$, $m_1 = m_2 = m_3 = 3$, $c_1 = c_2 = c_3 = 1/3$. Those results are to be compared with the reference electric potential V_{ref} and the electric field $\boldsymbol{e}_{ref}$ obtained by solving (65) with the exact value of w^-_{eff} in the special case considered, i.e. $w^-_{\mathit{eff}}(\boldsymbol{e}) = 2\|\boldsymbol{e}\|^4$. The potential V_{ref} and electric field $\boldsymbol{e}_{ref}$ are shown in Fig. 18. The relative error $|||V - V_{ref}|||/|||V_{ref}|||$ on the electric voltage is about 0.39%. The relative error $|||\boldsymbol{e} - \boldsymbol{e}_{ref}|||/|||\boldsymbol{e}_{ref}|||$ on the electric field is about 0.82 %.

In Fig 19 are represented the electric potential and the electric field obtained by solving (66) for a 3-phase composite material. We used the values $\sigma_1 = 1$, $\sigma_2 = 10$, $\sigma_3 = 5$, $m_1 = 1$, $m_2 = 3$, $m_3 = 6$, $c_1 = 0.5$, $c_2 = c_3 = 0.25$. As a partial

validation, the values taken by $\boldsymbol{e}$ and $\boldsymbol{j}$ on the top border of Ω are plotted in Fig. 20 and compared with the references values $\boldsymbol{j}_{ref}$ obtained from a direct evaluation of $\partial w^-_{\mathit{eff}}/\partial \boldsymbol{e}$ in (57). The rRMSE between the values of $\boldsymbol{j}$ and $\boldsymbol{j}_{ref}$ shown in Fig. 20 is about 0.82%. A mesh with 17002 P2 triangular elements (34405 degrees of freedom) has been used in the FEM simulations shown in Fig. 19. Ten Newton iterations were needed to reach convergence, for a total running time of 16 s. This is much lower than the expected running time for solving pb (65) using a direct evaluation of w^-_{eff} instead of the machine-learning model. The average time of a numerical evaluation of w^-_{eff} being 2.6 s, the expected running time for solving (65) is indeed about $10\times34405\times2.6 \simeq 248$ hours for the same mesh as used in Fig. 19. The benefit of the machine learning approach is mitigated by the overhead cost of building the training database and performing the learning procedure. In the present case, the training database consists of 10^6 samples, i.e. required 10^6 evaluations of w^-_{eff}. The computational time for building that database was about $2\,10^6$ s $\simeq$ 722 h. The running time for the learning procedure was about 10 h. Three FEM simulations of the type reported in Fig 19 are thus sufficient to absorb the overhead cost of setting up the machine-learning model.

7. Concluding remarks

This paper lays the foundations of a Fourier-base machine learning technique. Combining various ideas such as periodic extension, regularization and the use of hyperbolic crosses, it proved possible to tackle applications of interest in engineering. An attractive feature of the proposed method is that the training stage reduces to a quadratic programming problem. Future effort will be devoted to study other applications of the proposed method, improve its performance, and investigate the possible connections with neural networks. On that last point, it has been observed

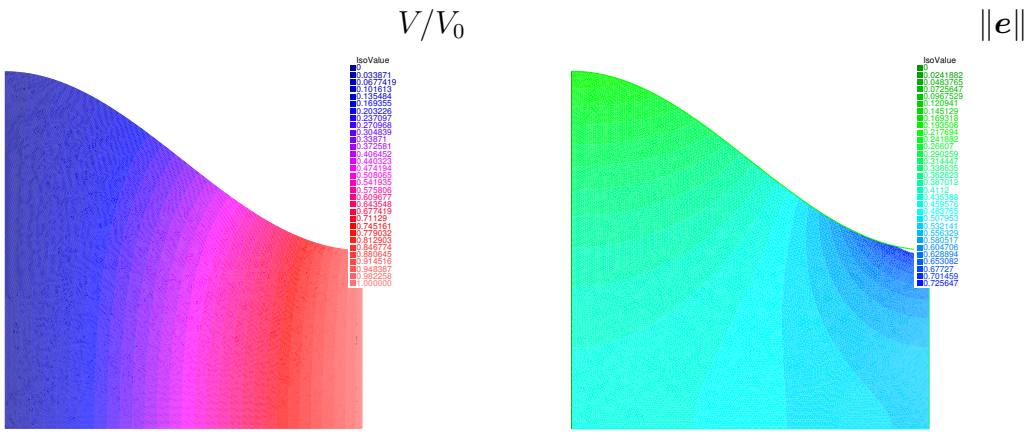

Figure 19: FEM simulations of the electric potential (left) and electric field (right) using the energy function $T_3 w_{\mathit{eff}}$ with parameters $\sigma_1 = 1$, $\sigma_2 = 10$, $\sigma_3 = 5$, $m_1 = 1$, $m_2 = 3$, $m_3 = 6$, $c_1 = 0.5$, $c_2 = c_3 = 0.25$. Because of symmetry, only the upper half of Ω is shown.

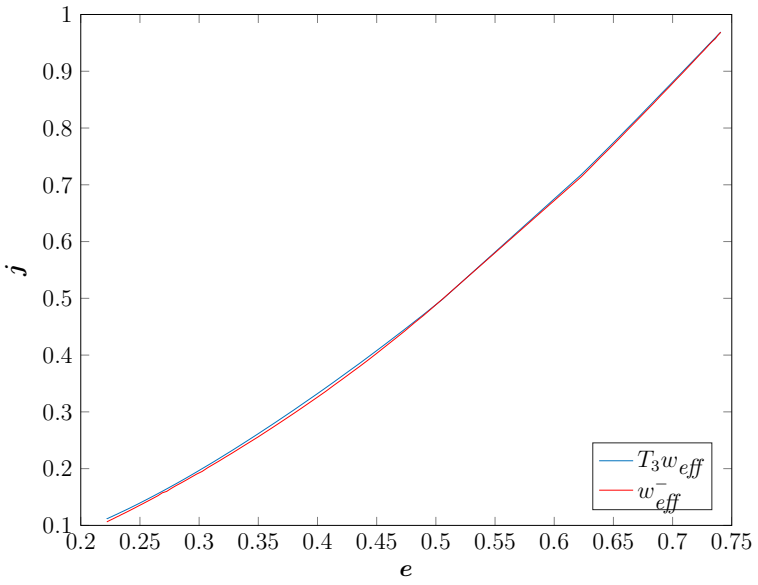

Figure 20: Values of $\boldsymbol{e}$ and $\boldsymbol{j}$ on the top border of Ω. Case $\sigma_1 = 1$, $\sigma_2 = 10$, $\sigma_3 = 5$, $m_1 = 1$, $m_2 = 3$, $m_3 = 6$, $c_1 = 0.5$, $c_2 = c_3 = 0.25$.

that the proposed technique behaves differently than neural networks, so there might be an interest in combining the two techniques to get 'the best of both worlds'. On a related note, the concept of High Dimensional Model Representation [28, 29] could possibly be useful in the present setting.

Appendix A. Expressions of $S_N f$ and $S_{\mathbf{H}(N)} f$

Recall that $S_N f$ is defined by

$$S_N f(\boldsymbol{x}) = \sum_{\boldsymbol{n} \in [-N..N]^d} c(\boldsymbol{n}, \tilde{f}) e^{i\pi \boldsymbol{n} \cdot \boldsymbol{x}} \tag{A.1}$$

For any *positive* multi-indice $\boldsymbol{a}$, let

$$\Gamma(\boldsymbol{a}) = \{\boldsymbol{n} \in \mathbb{Z}^d : |n_j| = a_j \, \forall j\}.$$

Property (11) shows that $c(\boldsymbol{n}, \tilde{f}) = c(\boldsymbol{a}, \tilde{f})$ for all $\boldsymbol{n} \in \Gamma(\boldsymbol{a})$. Expression (A.1) can thus be rewritten as

$$S_N f(\boldsymbol{x}) = \sum_{\boldsymbol{a} \in [0..N]^d} c(\boldsymbol{a}, \tilde{f}) \Big(\sum_{\boldsymbol{n} \in \Gamma(\boldsymbol{a})} e^{i\pi \boldsymbol{n} \cdot \boldsymbol{x}} \Big) \tag{A.2}$$

Let us first consider values of $\boldsymbol{a}$ such that

$$\begin{aligned} a_j &\neq 0 \quad \text{for } 1 \leq j \leq d' \\ a_j &= 0 \quad \text{for } d' < j \leq d \end{aligned} \tag{A.3}$$

for some d'. In such case, we have

$$\Gamma(\boldsymbol{a}) = \{(\epsilon_1 a_j, \cdots, \epsilon_{d'} a_{d'}, 0, \cdots 0) : \epsilon_j \in \{-1, 1\}, 1 \leq j \leq d'\}$$

so that

$$\sum_{\boldsymbol{n} \in \Gamma(\boldsymbol{a})} e^{i\pi \boldsymbol{n} \cdot \boldsymbol{x}} = \sum_{\substack{\epsilon_j \in \{-1,1\}, \\ 1 \leq j \leq d'}} \exp(i\pi \sum_{j=1}^{d'} \epsilon_j a_j x_j) = 2^{d'} \prod_{1 \leq j \leq d'} \cos \pi a_j x_j.$$

From expression (10) of h and the fact that $a_j = 0$ for $j > d'$, we find

$$\sum_{\boldsymbol{n} \in \Gamma(\boldsymbol{a})} e^{i\pi \boldsymbol{n} \cdot \boldsymbol{x}} = 2^{d'} h(\boldsymbol{a}, \boldsymbol{x}).$$

Let us now consider the general case where $\boldsymbol{a}$ is not of the form (A.3). Denoting by $\sigma(\boldsymbol{a})$ the number of non zero elements in $\boldsymbol{a} = (\boldsymbol{a}_1, \cdots, \boldsymbol{a}_d)$, there exists a permutation $\psi : [1\,..\,d] \mapsto [1\,..\,d]$ such that

$$\begin{aligned} a_{\psi(j)} &\neq 0 \quad \text{for } 1 \leq j \leq \sigma(\boldsymbol{a}) \\ a_{\psi(j)} &= 0 \quad \text{for } \sigma(\boldsymbol{a}) < j \leq d \end{aligned}$$

It follows from the previous calculations that

$$\sum_{\boldsymbol{n} \in \Gamma(\boldsymbol{a}')} e^{i\pi \boldsymbol{n} \cdot \boldsymbol{x}'} = 2^{\sigma(\boldsymbol{a})} h(\boldsymbol{a}', \boldsymbol{x}')$$

where $\boldsymbol{a}' = (a_{\psi(1)}, \cdots, a_{\psi(d)})$ and $\boldsymbol{x}' = (x_{\psi(1)}, \cdots, x_{\psi(d)})$. Observing that

$$\sum_{\boldsymbol{n} \in \Gamma(\boldsymbol{a}')} e^{i\pi \boldsymbol{n} \cdot \boldsymbol{x}'} = \sum_{\boldsymbol{n} \in \Gamma(\boldsymbol{a})} e^{i\pi \boldsymbol{n} \cdot \boldsymbol{x}}$$

and similarly that

$$h(\boldsymbol{a}', \boldsymbol{x}') = h(\boldsymbol{a}, \boldsymbol{x}),$$

we finally obtain

$$\sum_{\boldsymbol{n} \in \Gamma(\boldsymbol{a})} e^{i\pi \boldsymbol{n} \cdot \boldsymbol{x}} = 2^{\sigma(\boldsymbol{a})} h(\boldsymbol{a}, \boldsymbol{x}).$$

Substituting in (A.2) yields

$$S_N f(x) = \sum_{\boldsymbol{a} \in [0..N]^d} c(\boldsymbol{a}, \tilde{f}) 2^{\sigma(\boldsymbol{a})} h(\boldsymbol{a}, \boldsymbol{x}). \tag{A.4}$$

An expression similar to (A.4) can be obtained for the hyperbolic partial sum $S_{\mathrm{H}(N)} f$. Recall that

$$S_{\mathrm{H}(N)} f(x) = \sum_{\boldsymbol{n} \in \mathrm{H}(N)} c(\boldsymbol{n}, \tilde{f}) e^{i\pi \boldsymbol{n} \cdot \boldsymbol{x}}.$$

In a way similar to (A.2), we have indeed

$$S_{\mathrm{H}(N)}f(x) = \sum_{\boldsymbol{a}\in \mathrm{H}^+(N)} c(\boldsymbol{a},\tilde{f})(\sum_{\boldsymbol{n}\in\Gamma(\boldsymbol{a})} e^{i\pi \boldsymbol{n}\cdot\boldsymbol{x}})$$

from which it follows that

$$S_{\mathrm{H}(N)}f(\boldsymbol{x}) = \sum_{\boldsymbol{a}\in \mathrm{H}^+(N)} 2^{\sigma(\boldsymbol{a})} c(\boldsymbol{a},\tilde{f})h(\boldsymbol{a},\boldsymbol{x}).$$

Appendix B. Magnitude of Fourier coefficients of functions with bounded mixed derivative

Let $d' \in [1\,..\,d]$ and $\{j_1, \cdots j_{d'}\}$ be distinct indices in $[1\,..\,d]$. Differentiating the Fourier decomposition

$$h = \sum_{\boldsymbol{n}\in\mathbb{Z}^d} c(\boldsymbol{n},h) e^{i\frac{2\pi}{T}\boldsymbol{n}\cdot\boldsymbol{x}},$$

gives

$$\frac{\partial^{d'} h}{\partial x_{j_1}\cdots\partial x_{j_{d'}}} = \sum_{\boldsymbol{n}\in\mathbb{Z}^d} c(\boldsymbol{n},h)\left(i\frac{2\pi}{T}\right)^{d'} n_{j_1}\cdots n_{j_{d'}} e^{i\frac{2\pi}{T}\boldsymbol{n}\cdot\boldsymbol{x}}.$$

Parseval's identity implies that

$$\left(\frac{2\pi}{T}\right)^{2d'} \sum_{\boldsymbol{n}\in\mathbb{Z}^d} |c(\boldsymbol{n},h)|^2 (n_{j_1}\cdots n_{j_{d'}})^2 < \infty.$$

As a consequence, $|c(\boldsymbol{n},h)|(n_{j_1}\cdots n_{j_{d'}})$ is bounded independently of $\boldsymbol{n}$, i.e. there exists a constant $A(j_1,\cdots,j_{d'})$ such that

$$|c(\boldsymbol{n},h) n_{j_1}\cdots n_{j_{d'}}| \le A(j_1,\cdots,j_{d'}) \tag{B.1}$$

for all $\boldsymbol{n}\in\mathbb{Z}^d$. We set

$$A = \max_{d'\in[1..d]} \max_{\{j_1,\cdots j_{d'}\}} A(j_1,\cdots,j_{d'}). \tag{B.2}$$

Let now $\boldsymbol{n} = (n_1, \cdots, n_d)$ be given in $\mathbb{Z}^d$. By (26), there exists $d' \in [1\,..\,d]$ and distinct indices $j_1, \cdots, j_{d'}$ such that

$$\pi(\boldsymbol{n}) = |n_{j_1} \cdots n_{j_{d'}}|.$$

The values $n_{j_1}, \cdots, n_{j_{d'}}$ correspond to the non zero indices in $(n_1, \cdots, n_d)$. Eqs. (B.1) and (B.2) imply that

$$|c(\boldsymbol{n}, h))| \leq \frac{A}{\pi(\boldsymbol{n})}$$

where A is independent of $\boldsymbol{n}$.

Appendix C. Cardinality of hyperbolic crosses

Let $\mathrm{H}^+(M) = \{\boldsymbol{n} \in \mathbb{N}^d : \pi(\boldsymbol{n}) \leq M\}$. As M grows to infinity, $\operatorname{Card} \mathrm{H}^+(M)$ is on the order of the measure of the domain of $\mathbb{R}^d$ defined by the equation $x_1 \cdots x_d \leq M$ with $\boldsymbol{x} = (x_1, \cdots, x_d) \in [0, M]^d$, i.e.

$$\operatorname{Card} \mathrm{H}^+(M) \sim \int_{\substack{x \in [0,M]^d \\ x_1 \cdots x_d \leq M}} dx_1 \cdots dx_d.$$

Let

$$\Gamma(M, R, d) = \{\boldsymbol{x} \in (0, M]^d : \prod_{j=1}^{d} x_j \leq R\}$$

and

$$F(M, R, d) = \int_{\boldsymbol{x} \in \Gamma(M,R,d)} dx_1 \cdots dx_d.$$

Consider a given $\boldsymbol{x} = (x_1, \cdots, x_d)$ in $\Gamma(M, R, d)$. We have

$$0 \leq x_d \leq \min\{M, \frac{R}{x_1 \cdots x_{d-1}}\}.$$

Observe that $\min\{M, \frac{R}{x_1 \cdots x_{d-1}}\} = M$ if and only if $(x_1, \cdots, x_{d-1}) \in \Gamma(M, R/M, d-1)$. It follows that

$$F(M,R,d) = MF(M, \frac{R}{M}, d-1) + \int_{[0,M]^{d-1} - \Gamma(M, \frac{R}{M}, d-1)} \frac{R}{\prod_{j=1}^{d-1} x_j} dx_1 \cdots dx_{d-1}. \quad \text{(C.1)}$$

The integral in the right-hand side of (C.1) can be bounded from above by noting that

$$[0,M]^{d-1} - \Gamma(M, \frac{R}{M}, d-1) \subset [\frac{R}{M^{d-1}}, M]^{d-1}. \quad \text{(C.2)}$$

For any $(x_1, \cdots, x_{d-1}) \in [0,M]^{d-1} - \Gamma(M, \frac{R}{M}, d-1)$ and any given j, we have indeed $R/M < x_1 \cdots x_{d-1} \leq M^{d-2} x_j$ so that $R/M^{d-1} < x_j$. It follows from (C.1) and (C.2) that

$$F(M,R,d) \leq MF(M, \frac{R}{M}, d-1) + \int_{[\frac{R}{M^{d-1}}, M]^{d-1}} \frac{R}{\prod_{j=1}^{d-1} x_j} dx_1 \cdots dx_{d-1}.$$

Calculating the second integral on the right hand side gives

$$F(M,R,d) \leq MF(M, \frac{R}{M}, d-1) + R \log^{d-1} \frac{M^d}{R}. \quad \text{(C.3)}$$

Successive applications of (C.3) yields

$$F(M,M,d) \leq M(1 + \log M^{d-1} + (\log M^{d-1})^2 + \cdots + (\log M^{d-1})^{d-1}).$$

It follows that $F(M,M,d) = O(M \log^{d-1} M)$ as $M \to \infty$. Recalling that $\operatorname{Card} \mathrm{H}^+(M) \simeq F(M,M,d)$, we thus obtain that $\operatorname{Card} \mathrm{H}^+(M) = O(M \log^{d-1} M)$.

References

[1] Jörg F Unger and Carsten Könke, *Coupling of scales in a multiscale simulation using neural networks*, Computers & Structures **86** (2008), no. 21-22, 1994–2003.

[2] B.A. Le, J. Yvonnet, and Q.C. He, *Computational homogenization of nonlinear elastic materials using neural networks*, Int. J. Numer. Meth. Engng **104** (2015), 1061–1084.

[3] M.A. Bessa, R. Bostanabad, Z. Liu, A. Hu, Daniel W. Apley, C. Brinson, W. Chen, and Wing Kam Liu, *A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality*, Computer Methods in Applied Mechanics and Engineering **320** (2017), 633 –667.

[4] Zeliang Liu, CT Wu, and M Koishi, *A deep material network for multiscale topology learning and accelerated nonlinear modeling of heterogeneous materials*, Computer Methods in Applied Mechanics and Engineering **345** (2019), 1138–1168.

[5] Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, and E Weinan, *Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics*, Physical review letters **120** (2018), no. 14, 143001.

[6] George Cybenko, *Approximation by superpositions of a sigmoidal function*, Mathematics of control, signals and systems **2** (1989), no. 4, 303–314.

[7] Konstantin Ivanovich Babenko, *Approximation of periodic functions of many variables by trigonometric polynomials*, Doklady Akademii Nauk **132** (1960), no. 2, 247–250.

[8] Dinh Dũng, Vladimir Temlyakov, Tino Ullrich, and Sergey Tikhonov, *Hyperbolic cross approximation*, Springer, 2018.

[9] Loukas Grafakos, *Classical Fourier analysis*, Springer, 2008.

[10] George Fishman, *Monte Carlo: concepts, algorithms, and applications*, Springer Science, 2013.

[11] Harald Niederreiter, *Random number generation and quasi-monte carlo methods*, Vol. 63, Siam, 1992.

[12] Il'ya Meerovich Sobol', *On the distribution of points in a cube and the approximate evaluation of integrals*, USSR Computational Mathematics and Mathematical Physics **7** (1967), 86–112.

[13] John H Halton, *Algorithm 247: Radical-inverse quasi-random point sequence*, Communications of the ACM **7** (1964), no. 12, 701–702.

[14] David G Luenberger and Yinyu Ye, *Linear and nonlinear programming (fourth edition)*, Springer, 2016.

[15] R. Hill, *Elastic properties of reinforced solids: some theoretical principles*, J. Mech. Phys. Solids **11** (1963), no. 5, 357–372.

[16] J. R. Willis, *Variational estimates for the overall response of an inhomogeneous nonlinear dielectric*, Homogenization and effective moduli of materials and media, 1986, pp. 247–263.

[17] Z. Hashin and S. Shtrikman, *A variational approach to the theory of the effective magnetic permeability of multiphase materials*, J.Appl.Physics. **33** (1962), no. 10, 3125–3131.

[18] D. R. S. Talbot and J. R. Willis, *Variational principles for inhomogeneous non-linear media*, IMA Journal of Applied Mathematics **35** (1985), no. 1, 39–54.

[19] P. Ponte Castañeda, *The effective mechanical properties of nonlinear isotropic solids*, J. Mech. Phys. Solids **39** (1991), 45–71.

[20] V. Nesi, D. R. S. Talbot, and J.R. Willis, *Translation and related bounds for the response of a nonlinear composite conductor*, Proc. R Soc. London A **455** (1999), no. 1990, 3687–3707.

[21] Michaël Peigney, *On Hashin–Shtrikman-type bounds for nonlinear conductors*, Comptes Rendus Mécanique **345** (2017), no. 5, 353 –361.

[22] B.E. Peigney and M. Peigney, *Bounds for nonlinear composite conductors via the translation method*, Journal of the Mechanics and Physics of Solids **101** (2017), 93 –117.

[23] G.W. Milton, *The Theory of Composites*, Cambridge University Press, 2002.

[24] F. Hecht, *New development in freefem++*, J. Numer. Math. **20** (2012), no. 3-4, 251–265. MR3043640

[25] F. Feyel, *Multiscale FE2 elastoviscoplastic analysis of composite structures*, Computational materials science **16** (1999), no. 1-4, 344-354.

[26] K. and Kikuchi Terada N., *A class of general algorithms for multi-scale analyses of heterogeneous media*, Computer methods in applied mechanics and engineering **190** (2001), no. 40-41, 5427-5464.

[27] V. and Geers Kouznetsova M. G. D. and Brekelmans, *Multi-scale constitutive modelling of heterogeneous materials with a gradient-enhanced computational homogenization scheme*, International journal for numerical methods in engineering **54** (2002), no. 8, 1235-1260.

[28] Herschel Rabitz and Ömer F Aliş, *General foundations of high-dimensional model representations*, Journal of Mathematical Chemistry **25** (1999), no. 2-3, 197–233.

[29] Sergei Manzhos and Tucker Carrington Jr, *A random-sampling high dimensional model representation neural network for building potential energy surfaces*, The Journal of chemical physics **125** (2006), no. 8, 084109.