Data-driven acceleration of multiscale methods for uncertainty quantification: application in transient multiphase flow in porous media

Shing Chan¹ · Ahmed H. Elsheikh¹

Received: 1 April 2019 / Accepted: 2 December 2019 / Published online: 10 December 2019
© The Author(s) 2019

Abstract

Multiscale methods aim to address the computational cost of elliptic problems on extremely large grids, by using numerically computed basis functions to reduce the dimensionality and complexity of the task. When multiscale methods are applied in uncertainty quantification to solve for a large number of parameter realizations, these basis functions need to be computed repeatedly for each realization. In our recent work (Chan et al. in J Comput Phys 354:493–511, 2017), we introduced a data-driven approach to further accelerate multiscale methods within uncertainty quantification. The basic idea is to construct a surrogate model to generate such basis functions at a much faster speed. The surrogate is modeled using a dataset of computed basis functions collected from a few runs of the multiscale method. Our previous study showed the effectiveness of this framework where speedups of two orders of magnitude were achieved in computing the basis functions while maintaining very good accuracy, however the study was limited to tracer flow/steady state flow problems. In this work, we extend the study to cover transient multiphase flow in porous media and provide further assessments.

Keywords Uncertainty quantification · Machine learning · Multiscale finite element methods · Monte Carlo methods · Approximation methods · Neural networks

Mathematics Subject Classification 62M45 · 68T01 · 65M08 · 65C05

✉ Shing Chan
sc41@hw.ac.uk

Ahmed H. Elsheikh
a.elsheikh@hw.ac.uk

¹ School of Energy, Geoscience, Infrastructure and Society, Heriot-Watt University, Edinburgh, UK

1 Introduction

In many engineering problems, obtaining accurate and complete measurements of the parameters of a system is an impossible task. For example, subsurface simulations are an extreme case where only few measurements of the underground properties are available, gathered from a few wells that are sparsely distributed over an extensive zone. This makes uncertainty quantification (UQ) an ubiquitous task in these problems since any sensible prediction must account for the uncertainties in the unknown parameters.

The standard approach to UQ in systems with complicated nonlinear behaviors is to perform Monte Carlo simulations, i.e. to aggregate several predictions obtained from numerical simulations assuming different plausible realizations of the unknown parameters. Intuitively, the number of simulations required for accurate estimations would grow with the dimensionality of the unknown parameters. On top of this, the computational cost of the simulations also grows with the dimensionality of the unknown parameters. Clearly, this compound effect makes UQ extremely expensive when the dimensionality of the unknown parameters is very large. This is precisely the case in subsurface simulations and other geology-related problems, prompting the development of several techniques to accelerate UQ. Most of these techniques involve finding the right balance between accuracy and speed—the idea being that estimates based on a large number of slightly noisy observations can be more useful than estimates based on accurate but very few observations. This is the basis of surrogate modeling techniques (Wang and Shan 2007; Razavi et al. 2012), where the high-fidelity forward simulator is replaced by a much faster surrogate simulator, allowing more simulations perhaps at the expense of simulation quality. In the case of data-driven surrogates, the surrogate is constructed using a limited dataset of simulation results obtained from the high-fidelity forward simulator, typically gathered in an offline phase.

Another alternative that is not specific to UQ is to directly address the computational cost of the forward simulator itself by reducing the dimensionality of the problem. A straightforward approach is upscaling (Christie et al. 2001; Frippiat and Holeyman 2008), where a coarser version of the computational grid is devised in a way that the accuracy of the simulation is not compromised. A similar family of techniques called *multiscale finite element/volume methods* (Hou and Wu 1997) has gained popularity in recent years. Similar to upscaling techniques, multiscale methods achieve dimensionality reduction by resorting to coarser grids. In contrast to standard upscaling, however, this is achieved through the use of *numerical basis functions* that transfer information from the fine grid to the coarse grid. These basis functions are obtained by solving localized problems in the fine domain, thus capturing heterogeneities in the fine resolution.

In our recent work Chan and Elsheikh (2017), we introduced a framework that combines both surrogate modeling and multiscale methods. The idea is to incorporate a data-driven surrogate model into a multiscale method to replace part of its computational components to enable further acceleration. We noted that when multiscale finite element/volume methods are employed to run the simulations several times in an uncertainty propagation pipeline, the numerical basis functions need to be computed repeatedly for each parameter realization. We showed that it is possible to reduce the redundancy of these computations by constructing an effective low-cost surrogate to

compute the basis functions, replacing the relatively more expensive computation of solving the localized problems. This surrogate is modeled using snapshots from a limited number of simulation results—specifically, we collect samples of basis functions for a few number of runs (that are otherwise discarded after completion of each run) and use them to fit the surrogate model. This fitting can be done in an offline phase or in parallel to the simulation runs. Once fitted, the surrogate can then be engaged in subsequent runs for acceleration.

Within multiscale finite volume methods, several variations exist starting from Jenny et al. (2003) and quickly extended in following works Jenny et al. (2005), Hajibeygi et al. (2008, 2011), Wang et al. (2014), Tene et al. (2016) and Møyner et al. (2015), mostly differing on how the basis functions are computed. Here we adopted the variation presented in Lunati and Jenny (2006, 2008), but we note that the proposed framework is not limited to this specific multiscale method and can be applied to any method that relies on solving localized problem to obtain the basis functions. For the surrogate model, we use neural networks motivated by their well-known high expressive power (Hornik et al. 1989, 1990) enabled by the ever increasing computing power and data availability as well as recent advances in the field of machine learning. We emphasize that our goal is not to provide a new multiscale method, but to offer a general adaptable framework to further accelerate multiscale methods within UQ pipelines.

The study in our previous work Chan and Elsheikh (2017) was limited to tracer flow/steady state problems in porous media flow where the uncertain parameter is the subsurface permeability. In these simplified cases, the effective permeability is not affected by the relative permeability and remains constant in time. In this work, we extend our framework to address transient multiphase flow in porous media, and also provide further numerical assessments.

The rest of this paper is organized as follows: in Sect. 2, we briefly describe the two components of the proposed hybrid framework: multiscale methods (in particular multiscale finite volume methods) and artificial neural networks for regression. Our framework is described in Sect. 3. In Sect. 4, we assess the framework for uncertainty quantification in two-phase flow in porous media. Finally, we conclude our work in Sect. 5.

2 Background

We give a brief description of the two components employed in our framework: multiscale methods and regression using feedforward neural networks. Note that the use of neural networks is optional and any other regression model could be considered. As in Chan and Elsheikh (2017), we choose neural networks due to their high representational power. Regarding multiscale methods, we adopt the Multiscale Finite Volume method (MsFV) as presented in Lunati and Jenny (2006, 2008), but note that the framework is applicable to any multiscale method that shows modularity in the computation of the basis functions.

2.1 Multiscale finite volume method

Consider the following system of equations describing incompressible and immiscible multiphase flow of oil-water (Aarnes et al. 2007):

$$-\nabla \cdot \mathbf{K}\lambda(s)\nabla p = q \quad (1)$$

$$\varphi \frac{\partial s}{\partial t} + \nabla \cdot (f(s)v) = \frac{q_w}{\rho_w} \quad (2)$$

Equation (1) is the global pressure equation, where p denotes the global pressure, $\mathbf{K}$ denotes the intrinsic permeability tensor, $\lambda = \lambda_w + \lambda_o$ denotes the total mobility (where λ_w and λ_o are the water and oil mobilities, respectively), and $q = q_w/\rho_w + q_o/\rho_o$ (where q_w , ρ_w , q_o and ρ_o are the water source, water density, oil source and oil density, respectively). Equation (2) is the water saturation equation, where s denotes the water saturation, φ denotes porosity (volume of liquid over total volume of the media), v is the total velocity, and $f(s) = \lambda_w/(\lambda_w + \lambda_o)$ is the fractional flow function of water. Equation (1) and (2) are coupled through the saturation s and the velocity $v = -\mathbf{K}\lambda\nabla p$.

Typically, the computationally demanding part of this problem resides in the solution of the elliptic Eq. (1). After discretization using standard finite volume method, Eq. (1) becomes a system of linear equations $\mathbf{A}\mathbf{p} = \mathbf{q}$, which in subsurface flow problems is often extremely large due to the very fine grids for the spatial discretization that are normally required for accurate flow modeling. *Multiscale methods* aim to reduce this system of equations by using a coarser grid in a clever manner. Much like in finite element methods, multiscale methods consider a set of basis functions $\{\phi^1, \dots, \phi^{N_C}\}$, $\phi^i : \Omega \mapsto [0, 1]$ over the domain Ω , and assumes that the solution space of Eq. (1) is in the span of the basis functions, i.e. $p = \sum_i \phi^i p_C^i$. However, unlike standard finite element methods where these basis functions are pre-chosen piecewise polynomials, multiscale methods use numerically computed basis functions which are obtained by solving localized problems over the domain.

Briefly, computing a numerical basis function involves solving a local elliptic problem over a small region of the domain, typically coinciding with the support region of the basis function. The idea is that numerical basis functions obtained in this way will capture heterogeneities of the underlying domain, thus being much more informative than pre-chosen polynomial basis functions of standard finite element methods. In this way, multiscale methods can typically employ a much smaller number of basis functions for the same level of accuracy, resulting in much coarser grids. Once the basis functions ϕ^i are obtained, $p = \sum_i \phi^i p_C^i$ is plugged into Eq. (1) and—in the case of multiscale *finite volume* methods (MsFV)—standard finite volume discretization is carried out thereafter to obtain a *coarse* system of equations $\mathbf{A}_C \mathbf{p}_C = \mathbf{r}_C$. Hence, we now solve for vector $\mathbf{p}_C$ in the coarse resolution, and then use $p = \sum_i \phi^i p_C^i$ to obtain the pressure in the fine resolution. Acceleration is achieved since solving multiple small problems (the local problems plus the coarse system) is often faster than solving a single large problem, at least for most solvers of system of equations which have complexities that are polynomial in the problem size. The “multiscale” aspect of

the method is based on the possibility of iteratively repeating the procedure to further coarsen the domain and obtain discretizations at several scales, although most current implementations are only 2-scale.

The core of the method resides in obtaining these basis functions via the local problems. In essence, the local problems mimic Eq. (1) constrained to a region of the domain, and the boundary conditions are handled in special ways depending on the particular version of multiscale method. We refer the readers to Chan and Elsheikh (2017) and Lunati and Jenny (2006, 2008) for further details on multiscale finite volume methods. For the sake of accuracy, we also clarify that the version of MsFV that we follow (Lunati and Jenny 2006, 2008) also includes a correction term in the coarse discretization (which is the reason we write $\mathbf{r}_C$ and not $\mathbf{b}_C$ for the coarse system), but note that this will not be relevant to our further discussion and main contribution. Our main contribution is to further accelerate multiscale methods by replacing the computation of the local problems with a data-driven surrogate model to directly generate the basis functions. The data-driven surrogate is fitted using samples of the local solutions, as described in Sect. 3.

2.2 Feedforward neural networks for surrogate modeling

A feedforward neural network f is a composition of functions

$$f(\mathbf{x}) = f^{(n)}(f^{(n-1)}(\dots(f^{(1)}(\mathbf{x}))\dots))$$

where $f^{(i)}(\mathbf{z}) = \sigma_i(\mathbf{W}_i \mathbf{z} + \mathbf{b}_i)$, i.e. an affine transformation followed by an element-wise non-linearity. The functions $f^{(i)}$ are called the *layers* of the neural network. The functions $f^{(1)}$ and $f^{(n)}$ are also called the *input* and *output* layers, respectively, and the layers in between are called *hidden* layers. The component-wise non-linearities σ_i are called *activations*. Each component of the resulting vector $\sigma_i(\mathbf{W}_i \mathbf{z} + \mathbf{b}_i)$ is referred to as an *unit* of the layer i , and the number of components is referred to as the *size of the layer i* .

In practice, determining the neural network architecture—i.e. choosing n , σ_i and size of each layer—is largely guided by experience, heuristics, and prior information about the problem at hand (Bengio 2012), and possibly further tuned using hyperparameter optimization. The remaining parameters $\mathbf{W}_i$, $\mathbf{b}_i$ are determined by solving an optimization problem. Given set of observations $(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_{N_{\text{train}}}, \mathbf{y}_{N_{\text{train}}})$ comprising the *training set*, the optimization objective, or *training*, is

$$\arg \min_{\boldsymbol{\theta}} \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} \|\mathbf{y}_i - f(\mathbf{x}_i; \boldsymbol{\theta})\|_2^2 \quad (3)$$

where $\boldsymbol{\theta} := [\mathbf{W}_1; \mathbf{b}_1, \dots, \mathbf{W}_n; \mathbf{b}_n]$. This minimization problem is usually performed using gradient-based techniques, where the required gradients are obtained using an automatic differentiation algorithm (LeCun et al. 1989).

A good performance of the model in the training set does not necessarily translate to good performance in new unseen inputs, therefore regularization techniques are also

employed during the optimization. An ubiquitous technique is early stopping, where during the optimization the model performance is periodically assessed on a separate set of samples called the *validation set*: when the performance in this separate set stalls, the optimization is early stopped. Another technique is dropout (Srivastava et al. 2014), where some units of the neural network are randomly dropped out during the optimization. This approximates ensemble modeling as “several models” (each with different randomly dropped out units) are considered during the training procedure. Dropout is widely adopted in modern neural networks where fully connected layers are employed.

In terms of optimization stability, great progress has been made in recent years with batch normalization (Ioffe and Szegedy 2015), weight initialization methods (He et al. 2015) and new architecture designs (He et al. 2016; Srivastava et al. 2015), enabling stable training of deep neural networks with as many as 1000 layers (He et al. 2016). In particular, batch normalization has become ubiquitous in modern neural networks for its ability to combat the issue of exploding and vanishing gradients, as well as related issues with saturating activation functions, by ensuring that the inputs to each layer are normalized.

3 Methodology

For a concrete example, consider using a multiscale method in an uncertainty propagation task where we need to solve Eq. (1) and (2) for a large number of realizations of the permeability. When a multiscale method is used, for each individual realization we need to solve many local problems to obtain the basis functions. We aim to accelerate this process by constructing a surrogate to directly generate the local solutions, i.e. the basis functions. Let κ denote a local “permeability patch” that defines the local problem and ϕ the corresponding basis function (the permeability patch is basically defined by the support region of the basis function). The goal is to construct a surrogate model $\phi = f(\kappa)$ to generate ϕ directly from κ and that has a low evaluation cost, i.e. cheaper than solving the localized problems.

To train such model, a learning dataset is obtained by collecting the local solutions (κ_i, ϕ_i) from a small number of runs of the multiscale method, which are otherwise discarded after completion of the run. Concretely, consider realizations of the permeability $\mathbf{K}_1, \dots, \mathbf{K}_{N_K}$ and simulation time from 0 to t_f , and assume a time discretization $0 = t_0 < t_1 < \dots < t_{N_t} = t_f$. Let $m \ll N_K$ be the budget number of simulation runs of the multiscale method. For each individual permeability realization $\mathbf{K}_i$, $i = 1, \dots, m$ we have *snapshots* $\tilde{\mathbf{K}}_i^1, \tilde{\mathbf{K}}_i^2, \dots, \tilde{\mathbf{K}}_i^{N_t}$ where $\tilde{\mathbf{K}}_i^j := \mathbf{K}_i \lambda_{t_j}$, and $\lambda_{t_j} := \lambda(s(t_j))$. In turn, for each $\tilde{\mathbf{K}}_i^j$ we derive a set of pairs $D_i^j := \{(\kappa_1^{ij}, \phi_1^{ij}), (\kappa_2^{ij}, \phi_2^{ij}), \dots, (\kappa_{N_C}^{ij}, \phi_{N_C}^{ij})\}$ where N_C is the number of basis functions. The learning dataset is the union of all those pairs, i.e. $D := \cup_{i,j} D_i^j$. This dataset is later split into training set, validation set, and possibly test set, but it is important that such split be done *realization-wise* to reduce the correlation between samples.

In our work, we use neural networks for the surrogate and the training is performed as outlined in Sect. 2.2. Once trained, the surrogate is plugged into the multiscale method to solve for the remaining realizations $\mathbf{K}_{m+1}, \dots, \mathbf{K}_{N_K}$. The surrogate replaces the computation of the local problems, providing a faster way to obtain the basis functions in subsequent runs. In the case of shallow neural networks, a basis function is obtained through a few matrix-vector multiplications, which are almost always faster than solving elliptic problems of similar dimensionality.

To ensure the partition of unity property of the basis functions (which is not necessarily fulfilled in the learning approach), we perform a simple post-processing step as follows:

$$\phi_{new}^i(x) = \frac{\phi^i(x)}{\sum_k \phi^k(x)}$$

Using the generated basis functions, the resulting coarse system of equations is solved following the original multiscale method without modification—the physics is incorporated at this stage. Hence our framework is a data-physics hybrid: first the surrogate for the basis functions is obtained in a fully data-driven manner, then the physics is solved on a now coarse grid using the data-derived basis functions.

3.1 Dataset splits

We emphasize that critical to the effectiveness of our method in the transient case is to properly split the dataset $D := \cup_{i,j} D_i^j$ into train-validation-test sets at the level of permeability realizations $\mathbf{K}_i$. A valid train-validation split is of the form $D_{\text{train}} = \cup_{i \in A} \cup_j D_i^j$ and $D_{\text{val}} = \cup_{i \in B} \cup_j D_i^j$ where A and B are *disjoint* subsets of the set of indices $\{1, \dots, N_K\}$ of the realizations. This means that two snapshots $D_i^{j_1}$ and $D_i^{j_2}$ of the same realization i must be contained in the same split, since samples between $D_i^{j_1}$ and $D_i^{j_2}$ are highly correlated; in fact it is entirely possible that some permeability patches between $D_i^{j_1}$ and $D_i^{j_2}$ are identical, for example if the saturation at t_{j_1} and t_{j_2} remained the same in some parts of the domain. If our surrogate model is trained on sets that were not properly split, model validation will fail since the validation and test sets will likely contain past and future observations of the same training inputs (permeability patches), resulting in a subtle case of data leakage and overfitting. This is the main difference in methodology with respect to our previous work (Chan and Elsheikh 2017) where effective permeability remained unchanged in time.

3.2 Other effects on the permeability

Although here we only consider the effects of the saturation on the effective permeability (via the mobility terms λ_w and λ_o), our method is directly applicable when other sources of variation are present as long as it reduces to solving an elliptic equation (Eq. (1)), which is the scope of multiscale finite element/volume methods. In such

case, our framework is agnostic to the sources of variation on the elliptic parameter ($\mathbf{K}$ in this case), since the surrogate for the basis function is fully data-driven, and the variations will be captured in the training data as we collect several snapshots of the parameter for each realization and for each time-step.

3.3 Computational cost and scalability

It is clear that training the surrogate has its own computational cost, and as in any surrogate modeling procedure, the justification of this cost will depend on several practical factors. In practice, the training could be performed in an offline phase in parallel with the multiscale method which initially provides the stream of high-fidelity training samples (κ_i, ϕ_i) , and once the surrogate achieved a certain threshold of accuracy, it could be then engaged to the multiscale method in subsequent runs. A more important aspect is the scalability of the framework: we note that the framework is scalable with the size of the spatial and temporal discretizations simply because more samples can be collected per run, since a larger discretization grid and smaller time-step size result in more basis functions per simulation. Hence although the dimensionality of each basis function might increase with the domain discretization, this is compensated by the larger training dataset required to train larger surrogate models.

3.4 Limitations

Given that our current surrogate only handles a fixed input size and a fixed output size, the framework described is mostly efficient for structured regular grids, where the sizes of the permeability patches κ_i are equal (except for the boundary cases). For unstructured grids with varying permeability patches, a straightforward but rather inefficient solution is to train different surrogate models, each handling a different configuration of input and output sizes. This would also require more runs to collect sufficient samples, making the framework less efficient. A future research direction is to efficiently extend our framework to unstructured grids.

4 Numerical experiments

We consider the task of propagating 1000 realizations $\mathbf{K}_1, \dots, \mathbf{K}_{1000}$ of isotropic permeability defined over a unit square $[0, 1]^2$. Unlike our previous work (Chan and Elsheikh 2017), here we consider multiphase flow problems, hence the effective permeability $\tilde{\mathbf{K}} = \mathbf{K}\lambda$ changes over time as it depends on $\lambda(t)$. The realizations were generated assuming a zero mean Gaussian random field for the log-permeability with a spatial exponential covariance, i.e. $\text{Cov}(\mathbf{x}_1, \mathbf{x}_2) = c^2 \exp\left(-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|}{L}\right)$, where $\|\cdot\|$ denotes the Euclidean norm. We set a value of $L = 0.1$ as the correlation length, and $c = 1.0$ as the SD of the random field.

The discretization of the domain is 81×81 for the fine grid and 9×9 for the coarse grid—the latter is the grid over which the basis functions are constructed, meaning there are 9×9 basis functions. To simplify our presentation, we only focus

Fig. 1 Example of pressure solutions for each test case ($t = 0$)

on surrogating the computation of “interior basis functions”, i.e. basis functions whose support regions do not touch the domain boundary. This is because dealing with the boundary cases would necessitate training separate surrogate models; also, interior basis functions make up most of the basis functions in any practical situation. For the current discretization, we have 9×9 basis functions of which 7×7 are interior. The support region of each (interior) basis function is of size 19×19 (and so are the sizes of vectors κ and ϕ —note that we only need to store the values of ϕ in its support region). The simulated time is from $t_0 = 0$ to $t_f = 0.4$, discretized with a time-step of $\Delta t = 10^{-2}$, resulting in $N_t = 40$ time snapshots. In total, this results in $7 \times 7 \times 40 = 1960$ pairs (κ_i, ϕ_i) available per simulation run.

We consider two test cases:

Quarter-five spot problem In this problem, water is injected into a closed oil reservoir with the injector located at $(0, 0)$ and the producer located at $(1, 1)$, i.e. opposing corners of the unit square. No-flow boundary conditions are imposed on all sides of the square. We assume a unit injection/production rates, i.e. $q(0, 0) = 1$ and $q(1, 1) = -1$. Figure 1a shows an example of the pressure solution to this problem.

Uniform flow problem In this test case, we impose uniformly distributed water inflow on the left side of the square and outflow condition on the right side of the square. No-flow boundary conditions are assumed on the remaining top and bottom sides of the square. A total inflow/outflow rate of 1 is assumed. For the unit square, this means $v \cdot \hat{n} = -1$ and $v \cdot \hat{n} = 1$ on the left and right sides, respectively, where $\hat{n}$ denotes the outward-pointing unit normal to the boundary. Figure 1b shows an example of the pressure solution to this problem.

In both test cases, we impose a pressure of 0 at the center of the square domain to close the equations. Regarding the physical parameters, we assume homogeneous porosity $\varphi = 0.2$, and oil and water viscosities $\mu_w = 0.1$ and $\mu_o = 1.0$. The initial condition for the water saturation is $s(\mathbf{x}, t_0) = 0$, i.e. the reservoir initially contains only oil. We assume a quadratic mobility-saturation relationship, i.e. $\lambda_w = s^2/\mu_w$ and $\lambda_o =$

$(1-s)^2/\mu_o$. For convenience, all our parameters are unitless but were chosen to be representative of practical scenarios. For example, with our configuration of injection rate, simulation time and porosity means that $1.0 \times 0.4/0.2 = 2$ pore-volumes are injected—enough to cover the time interval where most of the interesting phenomena occur (e.g. large changes in saturation profile). A correlation length of 0.1 in the unit square is also easily interpreted as 10% of the side length. Regarding the fluid viscosities, they only affect the solutions as a function of their ratio, for which our configuration gives a ratio of 10 which is comparable to existing benchmarks (Christie et al. 2001).

4.1 Learning process

The surrogate model for the basis function prediction is a fully connected shallow neural network with 1-hidden layer and $\max(0, x)$ activation function. We use dropout (Srivastava et al. 2014) for regularization (along with early stopping). Preliminary experiments without any output activation function on the neural network produced basis functions that were very accurate, but sometimes contained values falling outside the $[0, 1]$ range, which is not acceptable for basis functions. We therefore use the hard sigmoid function as the output activation in order to bound the output values in $[0, 1]$. The hard sigmoid is the function $x \in \mathbb{R} \mapsto \max(0, \min(1, 0.2x + 0.5))$. Although this is a saturating activation function, we nevertheless obtained good results when employed along with batch normalization (Ioffe and Szegedy 2015). We use the optimizer Adam Kingma and Ba (2014) for training with default hyperparameters provided by its authors.

We set a budget of 50 simulation runs (out of 1000) to generate the learning dataset, resulting in a total of $50 \times 1960 = 98,000$ pairs (κ_i, ϕ_i) . The learning dataset is split into training and validation sets following standard 80%/20% split, where the split is done realization-wise: 40×1960 for training and 10×1960 for validation. The performance of the surrogate model is measured using the R^2 -score,

$$R^2(f; D_{val}) := 1 - \frac{\sum \|\phi^i - \hat{\phi}^i\|^2}{\sum \|\phi^i - \bar{\phi}\|^2}$$

where $\hat{\phi}^i = f(\kappa^i)$, $i = 1, 2, \dots, N_{val}$ are the *predicted* basis functions on the validation set, $\phi^1, \phi^2, \dots, \phi^{N_{val}}$ are the *true* basis functions, and $\bar{\phi} = \frac{1}{N_{val}} \sum \phi^i$ is the sample mean of the true basis functions. The R^2 -score has the following interpretation: the maximum score of 1 is achieved for perfect predictions; a score of 0 is obtained if the model naively predicts the mean; and finally the score can be arbitrarily large and negative since a regression model can perform arbitrarily bad. For a finer evaluation, we report the R^2 -score at each individual sub-dataset $D_{val}^0, D_{val}^1, \dots, D_{val}^{40}$ corresponding to the different snapshots $t_0 = 0, t_1 = 0.01, \dots, t_{40} = 0.4$, instead of reporting on the whole validation set.

We tested 15 different combinations of layer size with dropout rate: layer sizes of 64, 128, 256, 512 and 1024, trained with dropout rates of 0% (no dropout), 10%, and 40%. Each model is trained for between 15,000 and 30,000 iterations, taking between

Fig. 2 Visual assessment of predicted basis functions (using model $[64, 0]$, i.e. 64 hidden units and trained without dropout)

5 and 20 min on a Titan X GPU. In Fig. 2, we show the predicted outputs from the model of layer size 64 trained without dropout. We see that the surrogate generates basis functions that are visually extremely close to the true basis functions computed from local problems. To quantify this, we show in Fig. 3a, b the R^2 -scores obtained for the various combinations. We write $[512, 0.1]$ to indicate, for example, the neural network with 512 layer size and 10% dropout rate. The performances between test cases were found to be similar, following the same trend across snapshots. This could suggest that the learning task is not strongly influenced by the chosen boundary

Fig. 3 Model performance across snapshots

conditions. More specifically, the changes in the effective permeability $\tilde{\mathbf{K}}$ due to $\lambda(s)$ were not very significant in the learning task for these particular test cases. We see that most models benefited from a small amount of dropout regularization, while a dropout rate of 40% was too high resulting in model underfitting. Most models obtained very good scores ranging between 0.95 and 0.98 which are normally considered very high in regression problems. For a fast surrogate model, it is desirable to choose the smallest model among those satisfying the required performance threshold.

4.2 Hybrid data-driven multiscale method

We investigate the effects of embedding the different surrogates in the hybrid MsFV method. We consider three of our previously trained surrogates: 1024 hidden units trained with no dropout, 512 hidden units trained with 10% dropout, and 64 hidden units trained with 10% dropout. We choose these to see how models with very different sizes but similar R^2 -scores may affect the performance of the hybrid method. In the following, we denote the hybrid multiscale method with embedded surrogate by MsFV-NN.

We solve for the remaining 950 realizations $\mathbf{K}_{51}, \mathbf{K}_{52}, \dots, \mathbf{K}_{1000}$ using the hybrid MsFV-NN method as well as the standard MsFV (without surrogate approximation) for comparison. As a third point of comparison, we also solve using a standard finite volume method on the fine grid discretization, which will be the main high-fidelity reference. To assess the hybrid model, we compute errors with respect to the refer-

Fig. 4 Quarter-five spot problem Error correlation between MsFV and MsFV-NN

ence as follows: Let $\mathbf{u} = (u_1, \dots, u_n)$ be a vector of values corresponding to cells $\Omega_1, \dots, \Omega_n$ and let $|\Omega_i|$ be the area of cell i , the area weighted norm is defined as $\|\mathbf{u}\| = (\sum_i |u_i|^2 |\Omega_i|)^{1/2}$. Using this notation, the pressure error (e_p), the velocity error (e_v), and the saturation error (e_s) are computed as

$$\begin{aligned}
 e_p &= \frac{1}{T} \int_0^T \frac{\|\mathbf{p}^{ref}(\cdot, t) - \mathbf{p}(\cdot, t)\|}{\|\mathbf{p}^{ref}(\cdot, t)\|} dt \\
 e_v &= \frac{1}{T} \int_0^T \frac{\|\mathbf{v}_x^{ref}(\cdot, t) - \mathbf{v}_x(\cdot, t)\|}{\|\mathbf{v}_x^{ref}(\cdot, t)\|} + \frac{\|\mathbf{v}_y^{ref}(\cdot, t) - \mathbf{v}_y(\cdot, t)\|}{\|\mathbf{v}_y^{ref}(\cdot, t)\|} dt \\
 e_s &= \frac{1}{T} \int_0^T \frac{\|\mathbf{s}^{ref}(\cdot, t) - \mathbf{s}(\cdot, t)\|}{\|\mathbf{s}^{ref}(\cdot, t)\|} dt
 \end{aligned}$$

Fig. 5 Uniform flow problem Error correlation between MsFV and MsFV-NN

where $\mathbf{p}^{ref}$, $\mathbf{v}^{ref}$ and $\mathbf{s}^{ref}$ are the reference solutions, and $\mathbf{p}$, $\mathbf{v}$ and $\mathbf{s}$ are solutions using either MsFV or MsFV-NN. The errors are computed for each of the 950 runs for both MsFV and MsFV-NN methods. We compare the errors using a scatter-plot as shown in Figs. 4 and 5: the x -axis corresponds to MsFV errors and the y -axis to MsFV-NN errors. For perfect surrogate predictions, these points would be scattered exactly along the 45° line, indicated with a dashed-line in the plots. Overall, our results show a very good correlation between MsFV and MsFV-NN, where the points are well scattered around the 45° line. Out of the three neural networks considered, $[512, 0.1]$ (middle column) shows the highest correlation in all the three physical variables (pressure, velocity and saturation) in both test cases. Note that points on the 45° line only mean a perfect match between MsFV and MsFV-NN, so points below this line would mean that MsFV-NN outperformed MsFV. It is interesting to see that indeed some MsFV-NN runs had lower errors than MsFV, so the approximation method was

Table 1 Summary statistics for the *Quarter-five spot problem*

(a) Errors (mean/SD)			
	$e_p (10^{-2})$	$e_v (10^{-1})$	$e_s (10^{-2})$
MsFV	5.4547/2.5697	1.4936/0.2648	2.7851/0.6949
MsFV-NN [64, 0.1]	6.6798/2.6280	1.9508/0.3011	3.7596/0.8768
MsFV-NN [512, 0.1]	5.7203/2.5908	1.7140/0.2887	3.1294/0.7905
MsFV-NN [1024, 0]	5.8461/2.6407	1.7859/0.3017	3.2291/0.8126
(b) Uncertainty quantification (mean/SD)			
	$P_{(1/4, 1/4)} (10^{-1})$	$Q (10^{-1})$	$t_{wb} (10^{-1})$
Reference	1.6675/0.6645	2.6710/0.0507	2.8015/0.3760
MsFV	1.6695/0.6664	2.6774/0.0527	2.7535/0.3847
MsFV-NN [64, 0.1]	1.6219/0.6573	2.6815/0.0537	2.7370/0.3871
MsFV-NN [512, 0.1]	1.6685/0.6656	2.6775/0.0527	2.7570/0.3827
MsFV-NN [1024, 0]	1.6655/0.6664	2.6780/0.0529	2.7495/0.3841

Table 2 Summary statistics for the *Uniform flow problem*

(a) Errors (mean/SD)			
	$e_p (10^{-2})$	$e_v (10^{-1})$	$e_s (10^{-2})$
MsFV	3.3553/1.2492	2.5486/0.4131	2.3159/0.7419
MsFV-NN [64, 0.1]	4.5975/1.4884	3.7055/0.4878	2.9754/0.7768
MsFV-NN [512, 0.1]	3.8223/1.5525	3.1957/0.4578	2.6870/0.7993
MsFV-NN [1024, 0]	3.9148/1.4655	3.3763/0.4772	2.7371/0.7820
(b) Uncertainty quantification (mean/SD)			
	$P_{(1/4, 1/4)} (10^{-1})$	$Q (10^{-1})$	$t_{wb} (10^{-1})$
Reference	1.0561/0.4696	2.5949/0.0144	3.0550/0.3369
MsFV	1.0558/0.4702	2.5973/0.0154	3.0525/0.3354
MsFV-NN [64, 0.1]	1.0407/0.4670	2.5972/0.0154	3.0535/0.3404
MsFV-NN [512, 0.1]	1.0597/0.4756	2.5977/0.0155	3.0460/0.3382
MsFV-NN [1024, 0]	1.0523/0.4692	2.5975/0.0154	3.0570/0.3388

more accurate than the target method. This seems counterintuitive, but recall that MsFV-NN is a hybrid method where only the basis functions are approximated, while the resulting coarse system of equations is still solved using finite volume methods, thus providing an opportunity for correction. But more importantly, the true basis functions as computed from localized problems need not be optimal—in fact the derivation of optimal basis functions is an ongoing research topic and different flavours of MsFV in the literature propose different localized problems to derive the basis functions.

As found in our previous work (Chan and Elsheikh 2017), a consistent bias is seen in the velocity errors, but in the saturation errors—which depend on the velocity—this

Fig. 6 Quarter-five spot problem Estimations for several quantities of interest

bias is slightly corrected. We speculate that this is due to the error metric employed where the velocity error is computed component-wise and then summed. Finally, we summarize in Tables 1a and 2a the comparison of errors.

For further assessment, we perform statistical estimations for some quantities of interest: the mean production curve, the pressure at $(1/4, 1/4)$, the total oil produced, and the water breakthrough time (here defined as the time when the water fraction reaches 1% of the outflow). We compute these quantities for the 950 realizations using the different models and plot estimated densities. The estimated curves are shown in

Fig. 7 Uniform flow problem Estimations for several quantities of interest

Figs. 6 and 7. Overall, we see that the estimations are very close to reference, and in particular the results by MsFV and MsFV-NN are practically indistinguishable. Tables 1b and 2b summarize the statistics of the results. We see that the best performing surrogate models were exact (matched the reference) within three significant digits. Interestingly, we again found that some estimations from the surrogate models were more accurate than MsFV. Note that these are aggregate results from the 950 runs. Finally, we mention that our models could be further improved by fine-tuning the neural networks via hyperparameter optimization—see e.g. Chan and Elsheikh (2017).

We emphasize that we do not introduce a new multiscale method or claim to improve the accuracy of existing ones, but rather we propose a general framework for accelerating multiscale methods. The neural network models in this work are similar in size to the ones employed in our previous work (Chan and Elsheikh 2017), hence the speedup obtained are essentially the same (about two orders of magnitude) and we do not reproduce the details here [instead, see Table 4 of Chan and Elsheikh (2017)]. However, the performance of the framework in our previous work was only studied for tracer flow/steady state problems. Our current results support the effectiveness of the framework in transient multiphase flow where the effective permeability changes over time.

5 Conclusion

We extended a framework introduced in our previous work Chan and Elsheikh (2017) for accelerating multiscale methods within uncertainty quantification of transient multiphase flow problems in porous media. Our numerical experiments showed that a simple shallow neural network served as an effective surrogate for the computation of the localized problems to obtain the basis functions. Error correlations between the surrogate model and the base multiscale method were high across different variables and test cases. Estimated distributions for several quantities of interest were also practically indistinguishable. Overall, the surrogate model showed very high accuracy, and in some cases was more accurate than the base method due to the hybrid approach that allows for an intermediate correction step. Finally, we mention that although we applied neural networks for the surrogate model, other regression models can also be considered. Directions to extend this work include assessment in more complex permeability models such as channelized and anisotropic permeability.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Aarnes, J.E., Gimse, T., Lie, K.-A.: An Introduction to the Numerics of Flow in Porous Media Using Matlab. Springer, Berlin (2007)
- Bengio, Y.: Practical recommendations for gradient-based training of deep architectures. In: Montavon, G., Orr, G.B., Müller, K.-R. (eds.) *Neural Networks: Tricks of the Trade*, pp. 437–478. Springer, New York (2012)
- Chan, S., Elsheikh, A.H.: A machine learning approach for efficient uncertainty quantification using multiscale methods. *J. Comput. Phys.* **354**, 493–511 (2017)
- Christie, M.A., Blunt, M.J., et al.: Tenth SPE comparative solution project: a comparison of upscaling techniques. In: *Society of Petroleum Engineers, In SPE Reservoir Simulation Symposium* (2001)

- Frippiat, C.C., Holeyman, A.E.: A comparative review of upscaling methods for solute transport in heterogeneous porous media. *J. Hydrol.* **362**(1), 150–176 (2008)
- Hajibeygi, H., Bonfigli, G., Hesse, M.A., Jenny, P.: Iterative multiscale finite-volume method. *J. Comput. Phys.* **227**(19), 8604–8621 (2008)
- Hajibeygi, H., Karvounis, D., Jenny, P.: A hierarchical fracture model for the iterative multiscale finite volume method. *J. Comput. Phys.* **230**(24), 8729–8743 (2011)
- He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on ImageNet classification. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034 (2015)
- He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778 (2016)
- Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. *Neural Netw.* **2**(5), 359–366 (1989)
- Hornik, K., Stinchcombe, M., White, H.: Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Netw.* **3**(5), 551–560 (1990)
- Hou, T.Y., Wu, X.-H.: A multiscale finite element method for elliptic problems in composite materials and porous media. *J. Comput. Phys.* **134**(1), 169–189 (1997)
- Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015)
- Jenny, P., Lee, S.H., Tchelepi, H.A.: Multi-scale finite-volume method for elliptic problems in subsurface flow simulation. *J. Comput. Phys.* **187**(1), 47–67 (2003)
- Jenny, P., Lee, S.H., Tchelepi, H.A.: Adaptive multiscale finite-volume method for multiphase flow and transport in porous media. *Multiscale Model. Simul.* **3**(1), 50–64 (2005)
- Kingma, D., Ba, J.: Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014)
- LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. *Neural Comput.* **1**(4), 541–551 (1989)
- Lunati, I., Jenny, P.: Multiscale finite-volume method for compressible multiphase flow in porous media. *J. Comput. Phys.* **216**(2), 616–636 (2006)
- Lunati, I., Jenny, P.: Multiscale finite-volume method for density-driven flow in porous media. *Comput. Geosci.* **12**(3), 337–350 (2008)
- Møyner, O., Lie, K.-A., et al.: A multiscale method based on restriction-smoothed basis functions suitable for general grids in high contrast media. In: *Society of Petroleum Engineers, In SPE Reservoir Simulation Symposium* (2015)
- Razavi, S., Tolson, B.A., Burn, D.H.: Review of surrogate modeling in water resources. *Water Resour. Res.* **48**(7), W07401 (2012)
- Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**(1), 1929–1958 (2014)
- Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks. *arXiv preprint arXiv:1505.00387* (2015)
- Tene, M., Al-Kobaisi, M.S., Hajibeygi, H.: Algebraic multiscale method for flow in heterogeneous porous media with embedded discrete fractures (F-AMS). *J. Comput. Phys.* **321**(Supplement C), 819–845 (2016). <https://doi.org/10.1016/j.jcp.2016.06.012>
- Wang, G.G., Shan, S.: Review of metamodeling techniques in support of engineering design optimization. *J. Mech. Des.* **129**(4), 370–380 (2007)
- Wang, Y., Hajibeygi, H., Tchelepi, H.A.: Algebraic multiscale solver for flow in heterogeneous porous media. *J. Comput. Phys.* **259**, 284–303 (2014)