Predictive Modeling with Learned Constitutive Laws from Indirect Observations

Daniel Z. Huang\textsuperscript{a,*}, Kailai Xu\textsuperscript{a,*}, Charbel Farhat\textsuperscript{a,b,c}, Eric Darve\textsuperscript{a,b}

\textsuperscript{a}Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA, 94305
\textsuperscript{b}Mechanical Engineering, Stanford University, Stanford, CA, 94305
\textsuperscript{c}Aeronautics and Astronautics, Stanford University, Stanford, CA, 94305

Abstract

We present a new end-to-end framework for predictive modeling and its uncertainty quantification directly from observation data for physical systems, in which the coarse-grained models in the conservation laws are modeled with neural networks. Its counterparts, like response surfaces, are also compared in the present study, and the strength of neural networks is explored. The training process and the predictive process in this framework seamlessly combine the finite element method, neural networks and automatic differentiation. It also provides guaranteed convergence under mild assumptions and allows an efficient model form uncertainty quantification analysis. Numerical examples on a multiscale fiber reinforced plate problem and a highly nonlinear rubbery membrane problem from solid mechanics demonstrate substantial effectiveness of the framework.

Keywords: Neural Networks, Uncertainty Quantification, Finite Element Method, Homogenization

1. Introduction

Many practical problems arising from various engineering and scientific applications are heterogeneous and multi-scale in nature. The simulations of such problems based on first-principles models still remain prohibitively expensive. Coarse-grained models are often applied to approximate the effect of the microscopic interactions, in order to simplify and accelerate these simulations. For example, in solid mechanics, the constitutive relations potentially derived from cohesion between the atoms, are modeled empirically based on theoretical knowledge and ideal assumptions and calibrated on limited tensile test data. Coarse-grained models are also used in fluid mechanics, for turbulence closure and for rheological properties of non-Newtonian fluids. These modeling efforts in the best cases lead to affordable simulations of large scale engineering and scientific applications.

Coarse-grained models include purely phenomenological models, which directly relate several different empirical observations of phenomena to each other. With judicious use of relatively simple mathematical tools, these models are built in a way that is consistent with fundamental theory instead of from first principles. An alternative to phenomenological models are multiscale models, which are derived from finer scales or even from the molecular-based theories. At finer scales, established laws and simplified models are believed to be better understood. And the separation and coupling of different scales rely on asymptotic homogenization theory. Both approaches lead to certain model forms with a few parameters. Inverse analysis \cite{1}, as a prototype of a data-driven approach, has been used by various researchers to calibrate these model parameters.

\textsuperscript{*}Both authors contributed equally to this work, and are listed in alphabetical order.

Email addresses: zhengyuh@stanford.edu (Daniel Z. Huang), kailaix@stanford.edu (Kailai Xu), cfarhat@stanford.edu (Charbel Farhat), darve@stanford.edu (Eric Darve)
Along with the increase in data, data-driven approaches have been employed to coarse-grained modeling. Bayesian procedures, which express model parameters as a Gaussian random field and calibrate their hyper-parameters with data, have been applied to calibrate the Reynolds stress discrepancy \[2\] in the Reynolds-averaged Navier-Stokes (RANS) model, and even the whole differential equation \[4.5\]. Neural network approaches have gained in popularity to represent coarse-grained models in a form-free manner, in which the constitutive relation is not restricted to a prescribed form. For example, it was used to relate wall stresses with velocities in the turbulent wall model \[9\] and model the Reynolds stresses in a RANS model \[7\], which enables to predict flow separation near the wavy wall. And in solid mechanics, neural networks have been used to model the constitutive relations in a variety of materials, including concrete \[8\], sands \[9\], hyper-elastic materials \[10\], nonlinear elastic composites \[11\], crystal elastic materials \[12\], viscoelastic materials \[13\] and even multi-scale porous materials \[14\]. Most of the neural network training processes rely on the database of strain-stress pairs, strain-strain energy pairs, or strain-stress increments pairs as inputs and outputs. These data points consist of experimental measurements and numerical simulation results, which are generated from sub-scale simulations, like representative volume element (RVE) simulations \[15.16.17.18.19\], or post-processed from direct numerical simulations which need to resolve all scales of the problem. However, the comprehensive strain-stress relation measurement relying on simple mechanical tests, such as tensile or bending test, is challenging, especially for anisotropic materials. For RVE simulations, the determination of the size of the RVE \[18\] and material properties at the finer scales may be difficult. Direct numerical simulations generate high-fidelity training data sets. But for most practical problems, their computational costs are still unaffordable. To overcome these limitations, full-field measurement techniques such as digital image correlation or grid method \[20\] for structure deflection measurement, and particle image velocimetry \[21\] and magnetic resonance velocimetry \[22.23\] for fluid velocity measurement, have recently been used in the experimental mechanics community. These techniques are able to record the complete heterogeneous field, which are rich in the constitutive relations. However these data are indirect, namely there is generally no closed-form solution allowing a direct link between measurements and the stress or the underlying constitutive relations. The virtual fields method \[24,25,26,27\] has been designed to apply the finite element method (FEM) to bridge the full-field data with the constitutive relations. Inverse analysis is used to identify these constitutive parameters. Tartakovsky et al. \[28\] applied deep neural networks to directly inform the unknown constitutive relationship in the non-linear diffusion equations from the full-field data.

Inspired by this work, we built a general framework to combine traditional methods and data-driven coarse-grained models for predictive modeling. Our goal is to build a coarse-grained model with the following workflow. We conduct various experiments where given a mechanical system we apply different boundary conditions and observe the resulting deformation field. We wish to learn a constitutive law model (possibly with a coarse-grained or low-fidelity model) that is able to reproduce the observed deformations. Generally speaking a constitutive law is a function \(\mathcal{M}_\theta(u, x)\) where \(x\) is a location and \(u(x)\) is a function that represents the deformation of the object. Given some boundary conditions and the function \(\mathcal{M}_\theta(\bullet, \bullet)\) we can solve for \(u(x)\) using equations from mechanics. Our training set to learn the constitutive model \(\mathcal{M}\) consists of series of “experiments” (we use quotes since in practice these experiments may correspond to numerical calculations using for example a fine-scale or even atomistic model) with different choices of boundary conditions. For each “experiment,” we record \(u(x)\) at discrete locations. This problem is therefore not a regression problem where we must fit some given data \((u_i, x_i) \rightarrow \mathcal{M}_i\). Instead, we consider that we have a functional form \(\mathcal{M}_\theta(u, x; \theta)\) parameterized by \(\theta\) and we wish to optimize \(\theta\) such that the deformation field \(u(x)\) derived from \(\mathcal{M}_\theta(u, x; \theta)\) matches our observations.

A critical element is to determine the functional form best suited to this task. Many functional forms rely on a partitioning of the space \((u, x)\) into cells \(\Omega_i\) (for example in simplices, parallelepipeds, or simple geometrical shapes) and using low-order polynomials, or a local Fourier basis inside each \(\Omega_i\) \[29.30.31\]. However, if we examine our approach, we realize that these techniques are ill-suited. Indeed, consider a particular experiment associated with given boundary conditions. This will lead to some field \(u\), from which strain values can be computed. However, these strain values are not “uniformly” distributed in the domain. They typically will lie on some low-dimensional manifold (see Figure \[18\] for example).
collections of all strain values observed throughout all experiments has a very irregular and “anisotropic”
distribution (see Figure 18 to see a specific example of what we mean). Building an appropriate Ω
is therefore challenging and error prone. Choosing large Ω (large diameter) leads to poor reconstruction while
small Ω may lead to instabilities if no or few training points fall inside the cell. If one uses a Delaunay-
type triangulation [32, 33], the elements will be extremely distorted leading to ill-conditioned numerical
computations. Moreover the order of the basis (e.g., order of the polynomials) need to be chosen carefully. A
low-order basis will lead to large errors while a high-order basis leads to an unstable interpolation procedure.

Such approaches typically lead to an accurate reconstruction of \( M \) near observed points but may fail as
we move away. For example, Chebyshev polynomials [34, 35] are known to diverge rapidly outside the \([-1, 1]\)
interval. In addition, such approaches do not extend well to high-dimensional input data because the basis
construction typically relies on a tensor product construction which leads to an exponential number of basis
functions in the dimension of the space (that is the size of the basis scales like \( O(p^d) \) where \( p \) is the order
and \( d \) the dimension).

A more natural choice for such problems is to use radial basis functions [36, 37, 38, 39], that is an
approximation of the type:

\[
f(x) \approx \sum_i \alpha_i g_\sigma(\|x - x_i\|)
\]

where \( g_\sigma \) is for example a Gaussian function, an exponential, or a multiquadrics [40, 41], and \( x_i \) are centers
used in the approximation; \( \sigma \) is a scale parameter (for example the standard deviation of the Gaussian or
decay rate of the exponential). Such approaches work quite well even in high-dimension. However they have
many drawbacks. The main one is probably that computing the coefficients \( \alpha_i \) requires computing with the
matrix \( a_{ij} \overset{\text{def}}{=} g_\sigma(\|x_i - x_j\|) \) which is known to become ill-conditioned as the centers \( x_i \) get close to each
other. As a result even a small perturbation or error in the input data will lead to large changes in the
model coefficients, which is clearly undesirable. Methods like Kriging or Bayesian approaches [42, 43, 44, 45,
46, 47, 48] requiring assuming a priori statistical distribution which may or may not apply to the problem
at hand. The prior information typically leads to better conditioned linear systems that are easier to solve.
We note however that in Gaussian Process Regression the matrix used to calculate the model reverts to
\( g_\sigma(\|x_i - x_j\|) \) in the absence of noise in the observation and as the density of \( x_i \) increases the ill-conditioning
appears again.

In this context, deep neural networks (DNNs) offer many advantages. They possess “universal approx-
imation” properties [49, 50, 51, 52], such that few approximations on the form of the constitutive law are
required. They can be trained using nonuniform point cloud data. Using appropriate regularization, they
have good “generalization” properties, i.e. they remain accurate even away from training points. For example,
as the regularization penalization factor increases (using \( L_2 \) or \( L_1 \) regularization), the regression function
from a neural network becomes more “linear” and flat away from training points. Neural networks are known
to work well even for complex, highly inhomogeneous or anisotropic (that is dense along certain directions
and sparse along others) distribution of training points. For example, we will show in our benchmarks that
DNNs outperform piecewise linear surrogate surface (Section 6.3).

We also point out general good features of DNNs (although they are not explored in this manuscript):
they suffer less from the curse of dimensionality for high dimensional problems [53] (as DNNs are able to
use optimal projections of the input to lower dimensional manifolds), or from the Gibbs phenomenon for
 discontinuous problems [54] under certain conditions.

In addition, the implementation of neural networks is relatively straightforward and requires little modifi-
cation for different input dimensions. We have developed a suite of software libraries that make this method
more easily accessible to other researchers without deep technical expertise in automatic differentiation or
optimization. The code is accessible through the following url

https://github.com/kailaix/ADCME.jl

The applicability and accuracy of the learning procedure is analyzed based on a model problem. For
problems with a smooth underlying constitutive law, the learning process delivers an approximate constitutive
relation with an error bound depending on both the optimization error and the discretization error.
Besides learning the model, the uncertainties in the constitutive relation, due to the heterogeneity of the material or the inconsistency of the neural network model are also learned by our algorithm during the training process. Through sensitivity analysis, the uncertainty can be used to provide error bounds and intervals of confidence for the prediction.

The remainder of this paper is organized as follows. We first introduce the problem setup in Section 2 including the governing equations and the numerical scheme. Then in Section 3, we present our general framework for combining FEM and neural networks, specifically its training process and the predictive process. After that, we briefly discuss its applicability and give the accuracy analysis based on a model problem in Section 4. In Section 5, we present an approach for quantifying the predictive errors due to the heterogeneity of the material and neural network approximation. Finally, we apply the framework to a multi-scale fiber reinforced thin plate problem and a highly nonlinear rubber membrane problem in Section 6. We conclude and discuss possible generalization of the framework in Section 7.

2. Problem Setup

2.1. Model Problems

Consider a physical system described by static or steady partial differential equations

$$ P(u(x), M(u(x), x)) = F(u(x), x, p), \quad x \in \Omega \quad (1) $$

where the boundary conditions are excluded for brevity. The physical system is characterized by the generalized differential operator $P$ that defines a conservation law or other type of balance law, the state variable $u(x)$ is the solution of the physical system on the space domain $\Omega$. The generalized differential operator $M$ defines the coarse-grained model, like constitutive laws in structure mechanics and eddy viscosity models in Reynolds-averaged Navier-Stokes (RANS) equations. And $F$ represents the external force term or other source terms, which depends the parameter $p$.

The conservation law $P$ is regarded as a fundamental law of nature. However, the modeling term $M(u(x), x)$, which contains empirical assumptions and simplifications brings uncertainties and imperfectness to the mathematical description. In the proposed approach, the modeling term $M(u(x), x)$ in Eq. (1) is replaced by a neural network $M_{\theta}(u(x), x)$, which could be designed to embed as much physical information and a priori knowledge as possible. The $\theta \in \mathbb{R}^m$ denotes the hyper-parameters of the neural network. It is worth mentioning that different neural networks could be designed and applied to different computational areas, when physical properties of the problem clearly vary in different areas.

2.2. Discretization

The discretizations and solution strategies of the conservation law $P$ have been well established. When an appropriate discretization is applied to Eq. (1), it becomes

$$ P(u, M_{\theta}(u, x)) - F(u, x, p) = 0 \quad (2) $$

where $u \in \mathbb{R}^n$ is the discrete state vector corresponding to the spatial discretization of $u$, $P$ is the spatial discretization of the differential operator $P$, and $F$ is the discrete external force vector. In the present work, we mainly focus on solid mechanics applications, hence the FEM is applied to discretize the system.

Both the training process and the predictive process are based on the discrete Eq. (2). Bringing a given neural network model and the observed data $u$ into Eq. (2), the norm of the residual force is an indicator for the neural network model. Hence, the norm of the residual force is used as the loss function to train the neural network model. And in the predictive process, Eq. (2) is solved by the Newton method, which guarantees that predicted results satisfy the conservation laws. This marks the difference of current work and other data-driven paradigms [55, 56], which impose the conservation laws through constraints.
3. Data-driven Approach

In this section, data-driven techniques, applied to represent and extract the unknown modeling term $M(u, x)$ in Eq. (1) are introduced. The data-driven model $M_\theta(u(x), x)$ is one kind of phenomenological models, which avoid unaffordable computational cost paid for models derived from first principles. And data-driven models, combined only correct domain knowledge with sufficient data, are able to discern the underlying patterns and structure. Hence, they outperform the other empirical models in some previous studies [12, 7].

3.1. Neural Networks

The neural network itself is not an algorithm, but a framework to represent a complex model relating data inputs and data outputs. The framework is composed of several connected layers. Each layer takes in the output $x$ of the previous layer, transforms the inputs through an activation function $f(x)$ and outputs the result to the next layer. A nonlinear layer is defined as $f(x) = \sigma(Wx + b)$, here $W$ is the weight matrix and $b$ is the bias. $\sigma$ is called the activation function, such as the identity function, $\tanh$ and the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$. Multi-layer neural networks are compositions of many such functions, which can be conveniently written as

$$g(x) = f_1 \cdot f_2 \cdots f_L(x)$$

Such framework features the so called “universal approximation” property, which states that a one-layer feed-forward neural network with sufficient number of neurons can approximate any continuous functions on a compact subset, under mild assumptions on the activation functions [49]. Particularly, we have explicit approximation error bounds for one and two layer neural networks if the sigmoid activation functions are used [50, 51, 52]. Moreover, neural networks suffer less the curse of dimensionality for high dimensional problems [53] compared with polynomial approximations, and are able to avoid Gibbs phenomenon for discontinuous problems under certain conditions [54]. These provide us a strict mathematical justification of approximating unknown functions or models in the physical systems by using neural networks. Besides, a detailed comparison between the neural network model and the piecewise linear response surface model is presented in Section 6.3, which demonstrates the superior regularization and generalization properties of the neural network model.

In the present study, a three-layer neural network is applied for computational efficiency to approximate the unknown functions, such as the nonlinear constitutive law in this paper. However, when the physical properties of the underlying model are available, we can design special architecture to enforce the physical constraints or accelerate the computation. Besides, if the unknown function is complicated, deeper neural networks are preferred since they have been demonstrated numerically to be more expressive.

3.2. Training Process

Most of the neural network training processes [8, 9, 10, 11, 12, 13, 14] are direct constitutive law fitting, which rely on the cleaned input and output data of the modeling term $M(u(x), x)$, like strain vs stress data. But for complex materials or phenomena, the measurement or computation of high fidelity comprehensive input-output data of $M(u(x), x)$ would be challenging. In the present work, the neural network model is trained by an end-to-end approach, namely using data $u(x)$ and the associated load conditions, measured by full-field measurement techniques such as digital image correlation, grid method [20], particle image velocimetry [21], and magnetic resonance velocimetry [22, 23] or generated by high-fidelity numerical simulations. We assume the data set contains pairs of $(u_i, F_i)$, $i = 1, \ldots, N$, or only the external force and boundary conditions, which are enough to assemble the external force term $F_i$. For most of engineering applications, the data are limited, which are obtained by either high-fidelity simulations or experiments. Therefore, the present training process should be suitable and effective with a small data set. Thanks to the enormous richness of constitutive information contained in these data, the training process takes about $O(10)$ data pairs in the applications.
By substituting the unknown constitutive law \( \mathcal{M}(u(x), x) \) in Eq. (1) with the neural network approximation \( \mathcal{M}_\theta(u, x) \), which takes discretized displacement vector \( u \in \mathbb{R}^n \) or its associated strain field as the input, and outputs the stress field, we can formulate the loss function as

\[
L(\theta) = \sum_{i=1}^{N} (P(u_i, \mathcal{M}_\theta(u_i, x)) - F_i)^2
\]

(4)

Then \( L(\theta) \) is minimized to obtain an optimal parameter estimator \( \hat{\theta} \in \Theta \).

The optimization of the loss function Eq. (4) to determining the weights \( \hat{\theta} \) can be done by gradient descent methods, specifically the Limited-memory BFGS method in the present work. Modern frameworks such as TensorFlow we adopt in the paper provide us a way of computing the gradients \( \nabla_\theta L(\theta) \) using reverse mode automatic differentiation (AD). It applies symbolic differentiation at the elementary operation level. In AD, all numerical computations are ultimately compositions of a finite set of elementary operations for which derivatives are known, and combining the derivatives of the constituent operations through the chain rule gives the derivation of the overall composition. AD has forward modes and reverse modes. A thorough investigation of their properties is beyond the scope of this paper. In a nutshell, researchers only need to focus on the forward simulation. The differentiation and optimization parts are taken care of by the software. The traditional solver and the neural network are combined to fulfill the end-to-end training process.

Moreover, each evaluation of the loss function Eq. (4) and its gradient do not require solving the linear or nonlinear system, where an expensive Newton’s solver may be required in the latter case. Hence it is efficient even when the optimization needs thousands of steps to converge, which is the general case for large scale neural network optimization.

3.3. Prediction Process

The prediction process is straightforward, the Newton’s method with the load stepping is applied to solve Eq. (2). In the view of the fact that neural network models can efficiently deliver the prediction values and their derivatives with respect to any input variables. Therefore, traditional solvers are applied with minor changes in the model term query.

Although in our applications, \( \mathcal{M}(u(x), x) \) is assumed to depend only on \( u(x) \), specifically its gradient, i.e. the strain field, and so is \( \mathcal{M}_\theta(u, x) \), for simplicity and consistency, we keep using \( \mathcal{M}_\theta(u, x) \). Therefore we can compute the Jacobian of \( P \) in Eq. (2) as

\[
D_u P(u, \mathcal{M}_\theta(u, x)) = \nabla_u P(u, \mathcal{M}_\theta(u, x)) + \nabla_{\mathcal{M}_\theta} P(u, \mathcal{M}_\theta(u, x)) \nabla_u \mathcal{M}_\theta(u, x)
\]

(5)

where \( D_u \) designates the partial derivative with respect to the displacement field \( u \). \( \nabla_u \mathcal{M}_\theta(u, x) \) is obtained via automatic differentiation. The Jacobian can be used for the Newton’s solver.

The workflow described in Sections 3.1 to 3.3, from training the neural network model to predicting the material behaviours, is visualized in Fig. 1.

Figure 1: Workflow for the predictive modeling and its uncertainty quantification
3.4. Workflow of Refactoring Existing Codes for Inverse Problems

The workflow for the FEM-neural network approach to learn the constitutive law is shown in Fig. 2. On the left hand side, we show a typical forward FEM assembly workflow, where data nodes (dashed boxes), including constitutive law model, boundary and initial conditions, and even the displacement field of the physics system, and operation nodes (solid boxes), including the FEM discretization and local assembly, are numerical. On the right hand side, we show the current framework for solving the inverse problem. We identify the parts of operations that involve the unknowns, i.e. the “variable constitutive law”. If the constitutive law is assumed to be linear, the variable constitutive law is simply represented by several scalar variables. For nonlinear cases, a neural network is applied to approximate the nonlinear constitutive law, which is parameterized by its coefficient variables. The corresponding numerical operations involving these variables are now changed to “tensor operator”, which can be viewed as symbolic computation since the variables are not numerical: they are assigned values at runtime and are continuously adjusted during optimization.

In the perspective of the implementation, all we have to do is to overloading the numerical operations with the tensor operations and therefore the forward simulation codes are reusable. This is exactly what we have done in the numerical examples. We need to point out that by cleverly overload the numerical operators using TensorFlow operations, the refactoring process can be quite straightforward and simple. We demonstrate in Section 3.3 how the forward simulation codes are refactored for inverse problems with no more than 20 lines of modification in total. In theory, the simplicity can be achieved in any language with operator overloading and multiple dispatch features.

In the optimization phase, given the observations, we can form the “loss function” (See Eq. (4)), based on the discrepancy between observations and predictions by the variable constitutive law (see Fig. 3). At this point, we can compute the gradients of the loss function by “back-propagating” the intermediate gradients to the variable constitutive law. We can then update it using the information from the gradients. Note in the back propagation, we use automatic differentiation instead of symbolic differentiation, which is handled automatically by TensorFlow.

![Workflow for the forward simulation (left) and the inverse problem (right).](image-url)

Figure 2: Workflow for the forward simulation (left) and the inverse problem (right).
3.5. Implementation Details

In this section, we provide a demo to illustrate how an existing finite element code can be refactored for inverse problems. We leverage the extensive support of multiple dispatch, metaprogramming and operator overloading of the Julia language. We overloaded almost all numerical operators in Julia by TensorFlow operations.

Due to the limit of space, we omit the unchanged part of the original finite element code (which consists of more than 2000 lines), which are shown as “...”. The original codes are shown on the left hand side while the modified codes are shown on the right hand side of Fig. 4. We have modified five lines of code and deleted several lines irrelevant for inverse problems (such as computing Jacobian for the Newton’s iteration). The main idea is that we replace the variable part (tensors in Fig. 2) using the keyword Variable.

The optimization is almost effortless if we leverage the deep learning frameworks developed within the machine learning. To show the simplicity, we list the formulation of loss function together with the optimization procedure in Fig. 5. It is remarkable how the time-consuming optimization procedure can be implemented in an elegant and simple way.
function AssemblyNonlinear(d, current_pressure, nDim, nNodes, nDoF, nEquations, nNodesElement, nElements, nEdgesElement, Coord, ID, IEN, LM, EBC, g, NBC, Params)
    K = zeros(Float64, nEquations, nEquations)
    F = zeros(Float64, nEquations)
    P = zeros(Float64, nEquations)
    # precompute lambda1 and lambda2
    # ...
    global stress_P = ConstitutiveLaw(lambdas)
    # Assemble K and F
    for e = 1:nElements
        # ... # Obtain the tangent operator and internal load vector
        k_e, p_e, f_e = AxisMembranePressureArrays(e, d_e, current_pressure, Coord, IEN, NBC, g, nNodesElement, nDoF, LM, Params);
        # ...
        K[PI,PI] = K[PI,PI] + k_e[I,I];
        P[PI] = P[PI] + p_e[I];
        # ...
    end
    return K, P, F
end

function tf_AxisMembranePressureArrays(e, d_e, current_pressure, stress_P, Coord, IEN, nNodesElement, nDoF, LM, Params)
    k_e = zeros(Float64, nNodesElement * nDoF, nNodesElement * nDoF);
    p_e = zeros(Float64, nNodesElement * nDoF);
    # ...
    return k_e, p_e, f_e
end

function tf_AssemblyNonlinear(d, current_pressure, tf_Constitutive_Law, nDim, nNodes, nElements, nNodesElement, nDoF, nEquations, nEdgesElement, Coord, ID, IEN, LM, EBC, g, NBC, Params)
    F = zeros(Float64, nEquations)
    P = Variable(zeros(Float64, nEquations), trainable=false)
    # precompute lambda1 and lambda2
    # ...
    global stress_P = tf_Constitutive_Law(lambdas)
    # Assemble K and F
    for e = 1:nElements
        # ... # Obtain the tangent operator and internal load vector
        p_e, f_e = tf_AxisMembranePressureArrays(e, d_e, current_pressure, stress_P, Coord, IEN, nNodesElement, nDoF, LM);
        # ...
        P = scatter_add(P, PI, p_e[I])
        # ...
    end
    return P, F
end

Figure 4: Comparison of the original FEM codes and the refactored codes for inverse problems.
# Inverse Problem: There are 17 observations -- Uval and Pval

loss = constant(0.0)

for i = 1:17
    P, F = tf_AssemblyNonlinear(Uval[:,i], Pval[i], x->nn(x), nDim, nNodes, nElements, nDoF, nEquations, nEdgesElement, Coord, ID, IEN, LM, EBC, g, NBC, membrane_params)
    global loss += sum((P-F)^2)
end

# Optimization only requires three lines of codes
sess = Session(); init(sess)
opt = ScipyOptimizerInterface(loss, method="L-BFGS-B",options=Dict("maxiter"=> 20000, "ftol"=>1e-12, "gtol"=>1e-12))
ScipyOptimizerMinimize(sess, opt, fetches=[loss])

Figure 5: Julia codes for formulation of the loss function and optimization.

4. Applicability and Accuracy Analysis

In this section, the conditions under which the aforementioned learning procedure is effective and the error bound of the predictive model learned by neural networks are discussed for a model problem. Consider the 1D variable coefficient Poisson equation,

\[-\partial_x(\kappa(x)\partial_x u(x)) = f(x), \quad 0 < x < 1\]

\[u(0) = u_0, \quad u(1) = u_1\]  \hspace{1cm} (6)

Here \(u_0, u_1\) are two numbers, \(f(x)\) is the source function and the coefficient \(\kappa(x)\) is approximated by a neural network \(\kappa_\theta(x)\) parameterized by \(\theta\). Therefore, the approximated linear constitutive law in (2) has the following physics format,

\[M_\theta(u, x) = \kappa_\theta(x)\partial_x u\]

We assume that \(f(x)\) and \(\kappa(x)\) have sufficient regularity so that \(u(x)\) is also smooth.

The variational formulation of Eq. (6) is discretized by the FEM on a uniform domain partition \(T^h = \{0 = x_0 < x_1 < \cdots < x_N = 1\}\) with \(h = x_j - x_{j-1}, j = 2, 2, \cdots, N_e, \) Let \(C(P_1(T^h))\) denote the continuous piecewise linear function space on \(T^h\), a subspace of Sobolev space \(H^1(0, 1)\). The finite element formulation of Eq. (6) is given by: Find \(u^h \in S^h = \{u | u \in C(P_1(T^h)), u(0) = u_0, u(1) = u_1\}\) such that:

\[a(u^h, w^h) = \int_0^1 \kappa(x)u^h(x)w^h(x)dx = \int_0^1 fw^hdx = (f, w^h)\]  \hspace{1cm} (7)

holds for \(\forall w^h \in V^h = \{w | w \in C(P_1(T^h)), w(0) = 0, w(1) = 0\}\). Applying one point Gaussian quadrature rule in each element, the bilinear operator Eq. (7) is discretized as

\[a^h(u^h, w^h) = h \sum_{j=1}^{N_e} \kappa(x_{j-1/2}) \frac{w^h(x_j) - w^h(x_{j-1})}{h} \frac{u^h(x_j) - u^h(x_{j-1})}{h} = a(u^h, w^h) + O(h^2)\]  \hspace{1cm} (8)

In the case that \(w^h\) are local linear basis functions, the summation in Eq. (8) has at most two non-vanishing summands and the local error can be improved to \(O(h^3)\)

\[a^h(u^h, w^h) = a(u^h, w^h) + O(h^3)\]  \hspace{1cm} (9)
For the training process, we collect $N$ data pairs either from simulation or from experiment data, $(u_1, f_1), (u_2, f_2), \ldots, (u_N, f_N)$). The parameters $\theta$ for the neural network are updated by minimizing the loss function $L(\theta)$

$$L(\theta) = \sum_{i=1}^{N} ||P_i - F_i||^2$$

(10)

here $P_i = \{a^h(u_i, \phi_1), a^h(u_i, \phi_2), \ldots, a^h(u_i, \phi_{N-1})\}$ and $F_i = \{(f_i, \phi_1), (f_i, \phi_2), \ldots, (f_i, \phi_{N-1})\}$ are assembled by the FEM. And $\phi_i$ is the hat function at node $i$. $a^h$ is the discretized bilinear operator in Eq. (9) equipped with the neural network constitutive law. Assume that we are able to minimize the loss to $O(\epsilon_0)$, i.e.,

$$L(\theta) = \sum_{i=1}^{N} ||P_i - F_i||^2 = O(\epsilon_0)$$

(11)

Therefore, the optimization error of each component of Eq. (11) satisfies

$$a^h(u_i, \phi_j) - (f_i, \phi_j) = O(\epsilon_1), \quad \forall j = 1, 2, \ldots, N_e - 1 \text{ and } i = 1, 2, \ldots, N$$

(12)

On the average, $O(\epsilon_1) = O(\sqrt{\epsilon_0 / N(N_e - 1)})$, but for the worst case, $O(\epsilon_1) = O(\sqrt{\epsilon_0 / h})$.

Plugging the data into Eq. (7), and combining with Eq. (9) lead to

$$(a^h(u_i, \phi_j) = a(u_i, \phi_j) + O(h^3) = (f_i, \phi_j) + O(h^3), \quad \forall j = 1, 2, \ldots, N_e - 1 \text{ and } i = 1, 2, \ldots, N$$

(13)

Subtracting Eq. (12) from Eq. (13), we obtain

$$(a^h - a^h)(u_i, \phi_j) = O(\epsilon_1 + h^3)$$

(14)

Bringing Eq. (9) and the definition of the hat function into (14) leads to

$$(\kappa(x_{j-\frac{1}{2}}) - \kappa\theta(x_{j-\frac{1}{2}})) \frac{u_i(x_j) - u_i(x_{j-1})}{h} - (\kappa(x_{j+\frac{1}{2}}) - \kappa\theta(x_{j+\frac{1}{2}})) \frac{u_i(x_{j+1}) - u_i(x_j)}{h} = O(\epsilon_1 + h^3)$$

(15)

Consider another data pair $(u_k, f_k)$, we can obtain the same estimation as Eq. (15)

$$(\kappa(x_{j-\frac{1}{2}}) - \kappa\theta(x_{j-\frac{1}{2}})) \frac{u_k(x_j) - u_k(x_{j-1})}{h} - (\kappa(x_{j+\frac{1}{2}}) - \kappa\theta(x_{j+\frac{1}{2}})) \frac{u_k(x_{j+1}) - u_k(x_j)}{h} = O(\epsilon_1 + h^3)$$

(16)

Combining Eq. (15) and Eq. (16) leads to

$$\begin{bmatrix}
  u_i(x_j) - u_i(x_{j-1}) & u_i(x_{j+1}) - u_i(x_j) \\
  u_k(x_j) - u_k(x_{j-1}) & u_k(x_{j+1}) - u_k(x_j)
\end{bmatrix}
\begin{bmatrix}
  \kappa(x_{j-\frac{1}{2}}) - \kappa\theta(x_{j-\frac{1}{2}}) \\
  \kappa(x_{j+\frac{1}{2}}) - \kappa\theta(x_{j+\frac{1}{2}})
\end{bmatrix} = O(\epsilon_1 + h^3)$$

(17)

Through Taylor expansion of the data $u_i$ and $u_k$, we have

$$\det \begin{bmatrix}
  u_i(x_j) - u_i(x_{j-1}) & u_i(x_{j+1}) - u_i(x_j) \\
  u_k(x_j) - u_k(x_{j-1}) & u_k(x_{j+1}) - u_k(x_j)
\end{bmatrix} = h (u'_i(x_j)u''_k(x_j) - u'_k(x_j)u''_i(x_j)) + O(h^2)$$

(18)

The error bound of the neural network model at each Gaussian point solving by Eq. (17) is given as

$$||\kappa(x_{j-\frac{1}{2}}) - \kappa\theta(x_{j-\frac{1}{2}})|| \cdot |u'_i(x_j)u''_k(x_j) - u'_k(x_j)u''_i(x_j)| \leq O \left( \frac{\epsilon_1 + h^2}{h} \right),$$

$$||\kappa(x_{j+\frac{1}{2}}) - \kappa\theta(x_{j+\frac{1}{2}})|| \cdot |u'_i(x_j)u''_k(x_j) - u'_k(x_j)u''_i(x_j)| \leq O \left( \frac{\epsilon_1 + h^2}{h} \right)$$

(19)

When both $\kappa$ and $\kappa\theta$ are smooth enough, through interpolation of Eq. (19), we can obtain the error bound on the interior point $x_{j-\frac{1}{2}} \leq x \leq x_{j+\frac{1}{2}}$ as follows

$$||\kappa(x) - \kappa\theta(x)|| \cdot |u'_i(x_j)u''_k(x_j) - u'_k(x_j)u''_i(x_j)| \leq O \left( \frac{\epsilon_1 + h^2}{h} \right)$$
in other words,

\[
\|\kappa(x) - \kappa_\theta(x)\| \leq \mathcal{O}\left(\frac{\epsilon_1}{\gamma_j h} + \frac{h^2}{\gamma_j}\right), \quad \gamma_j = \sup_{i,k=1,2,\ldots,N} |u'_i(x_j)u''_k(x_j) - u'_k(x_j)u''_i(x_j)|
\] (20)

The error bound in Eq. (20) reveals a quantitative relationship between optimization, discretization and data. The error term consists of the optimization error \(\mathcal{O}(\frac{\epsilon_1}{h})\) and the discretization error \(\mathcal{O}(h^2)\). The errors are magnified by the reciprocal of the correlation of the data \(1/\gamma_j\). If there are sufficient data, we are able to have a lower bound for the correlation term \(\gamma_j\). In the limit case \(h \rightarrow 0\) and the optimization error \(\epsilon_1\) is much smaller than \(h\), i.e. \(\epsilon_1 = o(h)\), we obtain the convergence of \(\kappa_\theta(x)\) to the true coefficient \(\kappa(x)\).

For optimization error dominant cases, the lesson is that increasing mesh resolution may not improve model learning, namely, fully-resolved meshes are not necessary for problems with coarse-grained models.

In sum, the present model is applicable and effective, when the following conditions are simultaneously satisfied

• The neural network is consistent, namely with correct input features, the optimization error should tend to zero.
• Both the underlying model \(\kappa\) and the predicted model \(\kappa_\theta\) are smooth enough to have bounded derivatives.
• The data \(u_i\) and \(u_k\) should not be too correlated such that \(|u'_i(x_j)u''_k(x_j) - u'_k(x_j)u''_i(x_j)|\) is small or vanishes. However, the issue can be resolved, when the data set is large enough so that there exist sufficient non-correlated observations.

5. Uncertainty Quantification (UQ)

Despite the error bound derived in Section 4, the aforementioned neural network model contains uncertainties, due to the optimization error, incomplete input features of the neural network, data noise, and the homogenization error. Estimating the uncertainties of the FEM-neural network framework is critical for predictive modeling. There exists lots of prior work to quantify system uncertainties, such as Monte Carlo methods, which rely on repeated random forward sampling, polynomial chaos methods [57, 58], which determine the evolution of input uncertainty in a dynamical system through orthogonal polynomials, and Bayesian procedures [59, 60], which infer the posterior distribution of unknowns from existent data. And more recently neural network related techniques such as Dropout [61] and DNN-based surrogate [62] are applied to quantify the uncertainties in the neural networks. In this section, we propose a UQ method specifically for quantifying the homogenization error, which is similar with the neural network error due to the incompleteness of its input features. Because the heterogeneity information, i.e. the coordinate \(x\), is not incorporated in the neural network model in the present paper.

Solving the discretized governing equation (Eq. (2)), we have

\[
\hat{\mathbf{u}} = \mathbf{u}(\hat{\theta}, p) \quad (21)
\]

where \(p\) denotes the force load parameter. The approximated constitutive law model parameter \(\hat{\theta} \in \mathbb{R}^n\) is learned from data by minimizing Eq. (4). We have assumed a homogenized model, i.e., the constitutive relation is assumed to be constant in the computational domain. However, in reality, at each element or each Gaussian point, the constitutive relations are slightly different due to the heterogeneity of the material. Therefore, the true solution is

\[
\mathbf{u} = \mathbf{u}(\theta_1, \theta_2, \ldots, \theta_g, p) \quad (22)
\]

where \(g\) denotes the total number of the Gaussian quadrature points over the whole computational domain, and \(\theta_i \in \mathbb{R}^m\) is the parameter associated to the constitutive model at the \(i\)-th Gaussian point. The discrepancy between Eq. (22) and Eq. (21) is defined as the uncertainty derived from homogenization errors.
Gaussian point. They are assumed to be independent and identical distributed (i.i.d.) random variables. The parameter is approximated in the one-dimensional subspace, use
\[ \lambda \]
\[ \Sigma \]
here
\[ \theta \]
the random variable \( \Delta \) needs large amount of data. Based on the idea of active subspace methods proposed by Constantine et al. [63], \( u \) here of a linear manifold spanned by the gradients of quantities of interest (QoIs) \( W \) is constructed by recovering an orthogonal projection matrix denoted here by \( W \) here.

Bringing Eq. (27) into Eq. (23) leads to
\[ \Delta u = u(\theta, p) - u(\theta_1, \theta_2, \cdots \theta_g, p) \approx \frac{\partial u}{\partial(\theta_1, \theta_2, \cdots \theta_g)} [\Delta \theta_1, \Delta \theta_2, \cdots \Delta \theta_g]^T \] (23)

here \( \Delta \theta_i = \theta - \theta_i \in \mathbb{R}^m, i = 1, 2, \cdots, g \), represent the constitutive law model form uncertainties on each Gaussian point. They are assumed to be independent and identical distributed (i.i.d.) random variables. And we assume the error has no bias, namely \( E[\Delta \theta] = 0 \). Its variance \( \Sigma_\theta \) is estimated from the training data by solving the following least square (LSQ) problem,

\[
\min_{\Sigma_\theta \succeq 0} \sum_{i=1}^{n} \sum_{j=1}^{N} \left( (\Delta u^j_i)^2 - \frac{\partial u^j_i}{\partial(\theta_1, \theta_2, \cdots \theta_g)} \Sigma_{\theta} [\Delta \theta_1, \Delta \theta_2, \cdots \Delta \theta_g]^T [\Delta \theta_1, \Delta \theta_2, \cdots \Delta \theta_g] \left( \frac{\partial u^j_i}{\partial(\theta_1, \theta_2, \cdots \theta_g)} \right)^T \right)^2
\]

\[
= \sum_{i=1}^{n} \sum_{j=1}^{N} \left( (\Delta u^j_i)^2 - \frac{\partial u^j_i}{\partial(\theta_1, \theta_2, \cdots \theta_g)} \text{diag}(\Sigma_{\theta}, \Sigma_{\theta}, \cdots \Sigma_{\theta}) \left( \frac{\partial u^j_i}{\partial(\theta_1, \theta_2, \cdots \theta_g)} \right)^T \right)^2
\]

here \( u^j_i \) is the \( i \)-th component of \( u^j_{\text{obs}} \), i.e., \( j \)-th observation, and \( \Delta u^j_i \) is the \( i \)-th component of
\[ u(\theta, p) - u^j_{\text{obs}} \] (25)

However, in most cases, the parameter number is large, the estimation of the variance matrix \( \Sigma_\theta \in \mathbb{R}^{m \times m} \) needs large amount of data. Based on the idea of active subspace methods proposed by Constantine et al. [63], the random variable \( \Delta \theta \) is restricted to a low-dimensional subspace of \( \mathbb{R}^m \), represented by an associated matrix denoted here by \( W \in \mathbb{R}^{m \times k} \), whose dimension \( k \) is order of magnitude smaller than \( m \). The subspace is constructed by recovering an orthogonal projection matrix \( W \) obtained through the eigendecomposition of a linear manifold spanned by the gradients of quantities of interest (QoIs)

\[
\text{span}\{\nabla_{\theta} J^1(u(\theta, p)), \nabla_{\theta} J^2(u(\theta, p)), \cdots, \nabla_{\theta} J^k(u(\theta, p))\}
\]

(26)

here \( J^i, i = 1, 2, \cdots, m \) are QoIs. The random difference at each Gaussian location is modeled as
\[ \Delta \theta_i = W \lambda_i, \quad i = 1, 2, \cdots, g \] (27)

The vector of reduced coordinates \( \lambda_i \in \mathbb{R}^k \) is a zero-mean i.i.d. random variable with a variance matrix \( \Sigma_\lambda \). Bringing Eq. (27) into Eq. (23) leads to
\[ \Delta u^j_i \approx \frac{\partial u^j_i}{\partial(\lambda_1, \lambda_2, \cdots \lambda_g)} [\lambda_1, \lambda_2, \cdots \lambda_g]^T \] (28)

here \[ \frac{\partial u^j_i}{\partial(\lambda_1, \lambda_2, \cdots \lambda_g)} = \frac{\partial u^j_i}{\partial(\theta_1, \theta_2, \cdots \theta_g)} \text{diag}(W^T, W^T, \cdots, W^T) \] (29)

The variance matrix \( \Sigma_\lambda \in \mathbb{R}^{k \times k} \) is approximated by solving the following least square problem
\[
\min_{\Sigma_\lambda \succeq 0} \sum_{i=1}^{n} \sum_{j=1}^{N} \left( (\Delta u^j_i)^2 - \frac{\partial u^j_i}{\partial(\lambda_1, \lambda_2, \cdots \lambda_g)} \text{diag}(\Sigma_\lambda, \Sigma_\lambda, \cdots, \Sigma_\lambda) \left( \frac{\partial u^j_i}{\partial(\lambda_1, \lambda_2, \cdots \lambda_g)} \right)^T \right)^2
\]

(30)

In the present framework, the low-dimensional subspace is chosen to be one-dimensional (\( m = 1 \)) and we use \( \lambda \in \mathbb{R} \) to denote the reduced coordinate. The only QoI, \( J \), is taken to be the maximum principal stress. The parameter is approximated in the one-dimensional subspace,
\[ \Delta \theta = \lambda \frac{\nabla_{\theta} J}{\| \nabla_{\theta} J \|} \] (31)
The normalization is necessary since \( J \) can be quite different in scales for different external loads.

Equation (30) can be further simplified as

\[
\min_{\Sigma_{\lambda} \geq 0} \sum_{i=1}^{n} \sum_{j=1}^{N} \left( (\Delta u_{ij})^{2} - \Sigma_{\lambda} c^{i} \right)^{2}
\]

(32)

here \( c^{i} = \sum_{j=1}^{N} \frac{\partial u_{ij}}{\partial (\lambda_{1}, \lambda_{2}, \cdots, \lambda_{g})} \left( \frac{\partial u_{ij}}{\partial (\lambda_{1}, \lambda_{2}, \cdots, \lambda_{g})} \right)^{T} \). And normalizing the inner summation of Eq. (32) by \( \frac{1}{(c_{i})^{2}} \), when \( c^{i} > 0 \), can improve the least square estimation.

Based on the Chebyshev’s inequality, the LSQ estimated variance \( \hat{\Sigma}_{\lambda} \) satisfies

\[
P(|\Sigma_{\lambda} - \hat{\Sigma}_{\lambda}| \geq \epsilon) \leq \frac{\text{Var}(\Sigma_{\lambda})}{\epsilon^2 N n}
\]

(33)

Therefore, when \( N n \) is large enough, the estimation \( \hat{\Sigma}_{\lambda} \) obtained by the LSQ converges to \( \Sigma_{\lambda} \) with high probability.

During the prediction process, Monte Carlo sampling method is applied to compute the confidence interval

\[
u_{\text{pred}}(p) \approx u(\hat{\theta}, p) + \frac{\partial u}{\partial (\lambda_{1}, \lambda_{2}, \cdots, \lambda_{g})} [\lambda_{1}, \lambda_{2}, \cdots, \lambda_{g}]^{T}
\]

(34)

where \( \lambda_{i} \) is generated as a Gaussian random variable, \( \mathcal{N}(0, \Sigma_{\lambda}) \). It is worth mentioning that the sampling process does not require repeated solving the forward problem, which is efficient for large sampling.

6. Applications

In this section, we present numerical results from solid mechanics for the proposed “small-data”-driven predictive modeling procedure: a multi-scale fiber reinforced plate problem and a highly nonlinear rubbery membrane problem.

• The first problem serves as a proof-of-concept example for the end-to-end approach, where we calibrate the linear fourth-order stiffness tensor in the constitutive relation. It can also be viewed as a demonstration of guess-then-fit approach: we first guess that the material is subject to linear constitutive law and then we fit the parameters from data.

• The second problem tackles the nonlinear constitutive relation with neural networks. And the strength of the neural network approach is explored in a thorough comparison with the piecewise linear response surface (PLRS) approach. We show that in this case, PLRS is either overfitting with large degrees of freedoms or under-fitting with small degrees of freedoms and is susceptible to noise. Meanwhile, neural networks generalizes well and is quite robust to noise.

In both problems, the training data and test data of the displacement field are generated numerically, the underlying constitutive models are chosen to be uniform or non-uniform, i.e., containing random noise. The predicted results and the corresponding confidence interval on the test data are reported.
6.1. Fiber Reinforced Plate

Consider first a thin linear elastic fiber reinforced rectangular plate \([0, L] \times [-c, c]\) with width \(L = 100\) and height \(2c = 20\). The plate is supported on the left edge \(x = 0\), a pinned support at the center point and vertical roller supports at both corners, and subjected to a distributed load along the right edge \(x = L\). (See Fig. 6). The distributed load on the right edge is

\[
F = \left( -\frac{3PL}{2c^2}, \frac{3P(1 - (y/c)^2)}{4c} \right), \quad y \in [-c, c]
\]

(35)

here \(P\) is a load strength parameter. Both the matrix and the reinforcing fibre are made of homogeneous and isotropic elastic materials for which the Young’s moduli and Poisson ratios are listed in Table 1. This is a multiscale composite material problem, generally, to resolve each fiber is computationally unaffordable. Therefore, the homogenized constitutive law will be applied. Using mathematical homogenization, the governing linear elastostatics equations with plane stress assumptions are expressed in terms of the (Cauchy) stress components \(\sigma_{ij}\),

\[
\sigma_{ij,j} + b_i = 0 \text{ in } \Omega
\]

\[
u_i = \bar{u}_i \text{ on } \Gamma_u
\]

\[
n_j\sigma_{ji} = \bar{t}_i \text{ on } \Gamma_t
\]

(36)

here \(u_i\) is the displacement, \(\Omega\), \(\Gamma_u\), and \(\Gamma_t\) are the computational domain, the displacement boundary, and the traction boundary. Summation convention is employed for repeated indices. The strain tensor is

\[
\varepsilon_{mn} = \frac{1}{2} \left( \frac{\partial u_n}{\partial x_m} + \frac{\partial u_m}{\partial x_n} \right)
\]

(37)

The linear constitutive relation between strain and stress is written as

\[
\sigma_{ij} = \overline{C}_{ijmn}\varepsilon_{mn}
\]

(38)

here \(\overline{C}_{ijmn}\) are the homogenized constitutive tensor components.
The computational domain is discretized by $24 \times 12$ quad elements with linear shape functions. The solution has the finite element approximation $u^h = v^h + \bar{u}^h$. With any test function $w^h$, the integration form of Eq. (36) is written as

$$\int_\Omega \varepsilon(w^h)^T \mathbf{C} \varepsilon(v^h) d\Omega = \int_\Omega w^h_b b_i d\Omega + \int_\Omega w^h_i \bar{t}_i d\Omega + \int_\Omega \varepsilon(w^h) \mathbf{C} \varepsilon(\bar{u}^h) d\Omega$$

Integrating Eq. (39) with the 2 point Gaussian quadrature in each direction, the fully discretized governing equation becomes

$$K(\mathbf{C})v^h - F = 0$$

here $K$ is the stiff matrix depends on the homogenized constitutive tensor $\mathbf{C}$, $v^h$ is the solution vector, and $F$ is the external force vector.

**Figure 7:** Schematic of the fine-scale unit-cell problems with fiber volume fraction 1/9 (left) and 1/4 (right).

Several solution data pairs $(v_k, P_k), k = 1, ..., N$ are collected by varying the load strength $P$ in Eq. (35) applied on the right edge. The constitutive law used to generate data is obtained through the homogenization procedure discussed in [17, 19]. For each pair $mn$, a fine-scale unit-cell problem (See Fig. 7) resolving microscale features on $\Theta$ is constructed,

$$C_{ijkl}(\varepsilon^f_{kl} + I_{klmn})_{,j} = 0 \text{ in } \Theta$$

$$u^f_i(y) = u^f_i(y + Y) \text{ on } \partial \Theta$$

$$u^f_i(y) = 0 \text{ on } \partial \Theta_{\text{vert}}$$

with the superposition of a background strain $I_{klmn} = (\delta_{mk} \delta_{nl} + \delta_{mk} \delta_{ml})/2$ and the correction displacement $u^f$. The correction displacement $u^f$ is assumed to be periodic in all directions, and zero at all corners $\partial \Theta_{\text{vert}}$ of the unit cell. Here $C_{ijkl}$ denotes the constitutive tensor and $\varepsilon^f$ denotes the strain associated to the correction displacement $u^f$ in the unit-cell problem. By solving the fine-scale unit-cell problem Eq. (40), the homogenized constitutive tensor component $\mathbf{C}_{ijmn}$ is given as

$$\bar{C}_{ijmn} = \frac{1}{\Theta} \int_\Theta \sigma^f_{ij} d\Theta$$

where $\sigma^f_{ij} = C_{ijkl}(\varepsilon^f_{kl} + I_{klmn})$.

For the linear constitutive law, it is unnecessary to use a neural network for approximation. Instead, we only need to learn the entries of a symmetric matrix. However, the algorithm remains the same and automatic differentiation is the workhorse for the optimization. In all the experiments below, we minimize the loss function Eq. (4) using the L-BFGS-B optimizer. The maximum iteration is 5000 and the tolerance for the gradients norm and the relative change in the objective function is $10^{-12}$.

For the UQ analysis, the only QoI is chosen to be the maximum principal stress

$$\sigma_1 = \frac{\sigma_{11} + \sigma_{22}}{2} + \sqrt{\left(\frac{\sigma_{11} - \sigma_{22}}{2}\right)^2 + \sigma_{12}^2}$$

Its gradient forms the basis of the reduced subspace for the parameter $\bar{C}$.
### 6.1.1. Uniform Distributed Fiber

For this case, the volume fraction of the fiber is assumed to be constant of $1/9$ for the whole plate. The fine-scale unit cell problem Eq. (40) is solved, according to [19], to build the homogenized constitutive tensor component $C_{ijmn}$ as follows

$$C = \begin{bmatrix}
1491.24 & 701.024 & 0 \\
701.024 & 1450.24 & 0 \end{bmatrix}$$

(43)

One data point ($N = 1$) is generated with load strength $P = 20$, with the linear homogenized multiscale constitutive law. For the inverse problem, the loss function Eq. (4) is minimized to reach an optimal value of $10^{-12}$ within 50 steps. The following constitutive tensor is obtained

$$C_\theta = \begin{bmatrix}
1491.24 & 701.024 & -1.12339 \times 10^{-8} \\
701.024 & 1450.24 & -1.3416 \times 10^{-8} \\
-1.12339 \times 10^{-8} & -1.3416 \times 10^{-8} & 362.941
\end{bmatrix}$$

(45)

The constitutive law is recovered exactly from the proposed learning process for the linear case.

### 6.1.2. Nonuniform Distributed Fiber

For this case, the volume fraction of the fiber is assumed to be nonuniform, between $1/9$ and $1/4$. Consider two fiber volume fraction distributions depicted in Fig. 8, the fibers volume fraction decreases from the left to the right (Fig. 8-top) as follows,

$$\frac{1}{9} x^2 + \frac{1}{4} \left(1 - \frac{x}{2L}\right)$$

(46)

and the fibers are denser at the center of the plate and gradually become sparse along the radius direction (Fig. 8-bottom), as follows

$$\frac{1}{9} \sqrt{(x-L/2)^2 + y^2} + \frac{1}{4} \left(1 - \sqrt{(x-L/2)^2 + y^2} \right)$$

(47)

The homogenized constitutive tensor component $\mathcal{C}_{ijmn}$ obtained by the fine-scale unit cell corresponding to volume fraction $1/4$ (see Figure 7-right) is

$$\mathcal{C}' = \begin{bmatrix}
1695.92 & 747.42 & 0 \\
747.42 & 1633.96 & 0 \\
0 & 0 & 405.76
\end{bmatrix}$$

(48)

The homogenized constitutive tensor at each element for volume fraction between $1/9$ and $1/4$ is linearly interpolated between $\mathcal{C}$ and $\mathcal{C}'$. Therefore, at each element, the constitutive laws are different. Our data-driven algorithm is able to homogenize the material based on the global response, and compute the homogenized constitutive tensor. The learned constitutive laws for these two fiber volume fraction distributions from the observation with load strength $P = 20$ are

$$C_{\theta_1} = \begin{bmatrix}
1582.58 & 698.793 & 1.24528 \\
698.793 & 1512.1 & 2.80921 \\
1.24528 & 2.80921 & 377.979
\end{bmatrix} \quad \text{and} \quad C_{\theta_2} = \begin{bmatrix}
1673.94 & 738.872 & 2.59714 \\
738.872 & 1578.87 & 6.06215 \\
2.59714 & 6.06215 & 399.123
\end{bmatrix}$$

(50)

These learned constitutive laws $C_{\theta_1}$ and $C_{\theta_2}$ are averages between $\mathcal{C}$ and $\mathcal{C}'$ that minimize Eq. (4). For the UQ analysis, the variance of the reduced coordinate of the first case is $\Sigma_\lambda \approx 1.15 \times 10^5$, while in the second case, the variance is $\Sigma_\lambda \approx 4.21 \times 10^5$. 

17
Figure 8: The volume fraction distributions of fibers in the thin plane with non-uniform constitutive laws in Section 6.1.2.
We then apply the learned constitutive laws and its corresponding variance $\hat{\Sigma}_\lambda$ to predict displacements for various load strengths ($P = 25, 35, 45, 55, 65, 75$) and perform UQ analysis. The exact solution, the predicted solution, and the confidence intervals corresponding to the quantiles 0.95 and 0.05, constructed with 2000 samples generated by Eq. (34) are depicted in Figs. 9 and 10. The predicted result and the exact solution are in good agreement, the confidence regions of the constitutive law fluctuations contain exact solutions.

Figure 9: Predicted shape of thin plate with a non-uniform fiber distribution (See Equation (46)) with learned constitutive laws, subjected to different external loads. The grey mesh is the undeformed configuration, the blue mesh is the exact deformed configuration. The red mesh is the predicted deformed configuration. The green and the pink regions correspond to the quantiles 0.95 and 0.05 results.
Figure 10: Predicted shape of thin plate with a non-uniform fiber distribution (See Equation (17)) with learned constitutive laws, subjected to different external loads. The grey mesh is the undeformed configuration, the blue mesh is the exact deformed configuration. The red mesh is the predicted deformed configuration. The green and the pink regions correspond to the quantiles 0.95 and 0.05 results.
6.2. Rubber Membrane

![Figure 11: Schematic of the axisymmetric rubber membrane: the initial undeformed configuration (dash line) and the current deformed configuration (solid line).](image)

Finally, the rubber membrane with both material and geometric nonlinearities is considered. The circular membrane of radius $L = 1$ is initially flat, and then stretched and inflated under the non-conservative (follower) pressure load (See Fig. 11). Due to the axisymmetry, the model problem is reduced to a 1D problem, described in the ($r, z$) plane. A point in the undeformed configuration has coordinate $(R, Z) \in [0, 1] \times \{0\}$ with line element $dS$, and in the deformed configuration has coordinate $(r, z)$ with line element $ds$. The displacement vector is defined as

$$u_r = r - R, \quad u_z = z - Z$$

Three principal stretch ratios of the membrane at the point are given by

$$\lambda_1 = \frac{ds}{dS}, \quad \lambda_2 = \frac{2\pi r}{2\pi R}, \quad \lambda_3 = \frac{t}{T}$$

Here $ds = \sqrt{dr^2 + dz^2}$, $dS = \sqrt{dR^2 + dZ^2}$, and $t$ and $T$ are the thickness of the membrane in the deformed configuration and the undeformed configuration. The rubber membrane is assumed to be incompressible,

$$dSdT = dsdt \implies \lambda_1 \lambda_2 \lambda_3 = 1$$

The corresponding first Piola-Kirchhoff stresses are $P_1$, $P_2$ and $P_3 = 0$. The weak form of the governing equations is written as

$$2\pi T \int_0^1 (P_1 \delta \lambda_1 + P_2 \delta \lambda_2) RdS - 2\pi p \int_0^r n \delta ur ds = 0 \quad \forall \delta u$$

Here the first term is the virtual internal work and the second term is the virtual external work corresponding to the non-conservative (follower) pressure load $p$. Due to the symmetry and pre-stretch of the membrane, the boundary conditions are

$$u_r(0, 0) = 0, \quad u_r(1, 0) = \bar{u}_r, \quad u_z(1, 0) = 0$$

The computational domain is discretized by 100 elements with linear shape functions. Integrating Eq. (53) with 3 point Gaussian quadrature in each element lead to the fully discretized governing equation

$$P(u, M) = F(u, p)$$

here $P$ and $F$ correspond to the discretization of the first and second term in Eq. (53), and $M$ represents the parameters $P_1$, $P_2$, which are unknowns and will be calibrated with observations. Here $M$ is the constitutive law related the principal stretches to the first Piola-Kirchhoff stresses.
Several solution data \((u_k, p_k)\) are collected by varying the pressure load \(p\). For generating the data, the rubber membrane is a Mooney-Rivlin hyperelastic incompressible material \([64]\) with an energy density function \(W\) as follows,

\[
W(\lambda_1, \lambda_2, \lambda_3) = \mu(\lambda_1^2 + \lambda_2^2 + \lambda_3^2 - 3) + \alpha(\lambda_1^2 \lambda_2^2 + \lambda_2^2 \lambda_3^2 + \lambda_3^2 \lambda_1^2 - 3)
\]

\(J = \lambda_1 \lambda_2 \lambda_3 = 1\) \hspace{1cm} (55)

Here \(\mu\) and \(\alpha\) are model parameters. The true constitutive law is given as

\[
P_1 = \frac{\partial W}{\partial \lambda_1} \quad \text{and} \quad P_2 = \frac{\partial W}{\partial \lambda_2}
\]

After nondimensionalizing the problem parameters, \(p' = \frac{p}{\mu T}\), \(\alpha' = \frac{\alpha}{\mu}\), \(P_1' = \frac{P_1}{\mu}\), and \(P_2' = \frac{P_2}{\mu}\), the integration form Eq. (53) becomes

\[
\int_0^1 (P_1' \delta \lambda_1 + P_2' \delta \lambda_2) RdS - p' \int_0^{r'} n \delta u rs = 0 \quad \forall \delta u
\]

The constitutive relation is approximated by a neural network

\[
\mathcal{M}_\theta : (\lambda_1, \lambda_2) \rightarrow (P_1', P_2')
\]

In all the experiments below, we minimize the loss function Eq. (4) using the \textit{L-BFGS-B} optimizer. The maximum iteration is 20000 and the tolerance for the gradients norm and the relative change in the objective function is \(10^{-12}\).

For the UQ analysis, the only QoI is chosen to be the maximum principal stress, in this case the maximum principal stress is \(\text{max}\{P_1', P_2'\}\). Its gradient forms the basis of the reduced subspace for the parameter \(\theta\).

6.2.1. Uniform Constitutive Law

Consider the Mooney-Rivlin hyperelastic incompressible rubber membrane with parameter \(\alpha' = 0.1\), subjected to non-dimensional pressure loads \(p'\) of 0.0, 0.5, 1.0, ..., 7.5, 8.0. We collect 17 data points \((u_k, p_k)\) to train the FEM-neural network framework. The neural network consists of two hidden layers with 20 neurons in each layer; the activation function is \(\tanh\) since it is smooth. The neural network converges within 15000 iterations.

Figure 12 shows the calibrated and the exact constitutive laws. Since the present supervised learning framework is incapable of extrapolating, predicting the corresponding stresses for strains \(\lambda_1, \lambda_2\) that are outside of the information scope embedded in the 17 data points, we do not expect the neural network result to be accurate for all \((\lambda_1, \lambda_2)\). Therefore, in the figure, only a subset of strains \(\lambda_1, \lambda_2\) that appear in the inputs are depicted. Nevertheless, in the test below, we also demonstrate the potential for the neural network to extrapolate \(\lambda_1, \lambda_2\) out of the scope. For the first component of the stress \(P_1'\) with respect to the strain \((\lambda_1, \lambda_2)\), the calibrated constitutive law relation almost overlaps with the exact one. For the second component \(P_2'\), the calibrated result deviates a little from the exact one near \(\lambda_1 = 6, \lambda_2 = 1\). This is because the loss function, the residual force Eq. (4), is less sensitive to the second component compared with the first component. However, we have treated both components equally in the formulation of the loss function. The back propagation will update the first component more effectively than the second one.
Figure 12: The exact constitutive law (reference) and the surrogate constitutive law learned by the neural network approach for the rubber membrane in the uniform constitutive law case.

We then apply the learned constitutive law and its corresponding variance $\hat{\Sigma}_\lambda \approx 2.43 \times 10^{-6}$ to predict the displacements for various load strengths ($p' = 2.2, 4.2, 6.2, 8.2$) and perform UQ analysis. Figure 13 shows the exact displacements, the predicted displacements and the confidence intervals corresponding to the quantiles 0.95 and 0.05, constructed with 2000 samples generated by Eq. (34) in the $z$ and $r$ directions. We need to point out we are also able to extrapolate to predict the behavior of the rubber membrane subjected to a pressure load of 8.2, although the neural network has not seen any observations beyond pressure 8.0.

The exact displacements and the predicted displacements almost collapse on top of each other. As for the quantification of the homogenized error, the blue uncertainty region is almost vanished, which means the predicted value is of high reliability. This is consistent with the fact that there is no noise in the underlying constitutive model.

6.2.2. Non-uniform Constitutive Law

Due to the thickness and material variations of the rubber membrane, the constitutive law can be noisy. The noise is incorporated in the test problem by varying the parameter $\alpha'$ in the energy density function Eq. (55) at different elements. The $\alpha'$ used to generate data is randomly sampled, which represents the noise in the constitutive law due to the thickness and material variations of the rubber membrane. The $\alpha'(R)$ curve is depicted in Fig. 14, which is assumed to be continuous, and about 10% variation is included. We generate 17 data with the same non-dimensional pressures as in the previous uniform constitutive law case. The displacements of the underlying model are solved with heterogeneous constitutive laws. In addition, we adopt the same neural network structure, optimizer and training iterations as that in the uniform case.

Figure 15 shows the calibrated relation and the reference constitutive relation corresponding to $\alpha' = 0.1$. Note in this case, the homogenized constitutive relation (reference) is not necessary the true relation but only serves as a reference. What is more relevant is how we can predict the behaviour of the rubber membrane subjected to the external pressure load. Since the training data cover the range $p \in [0, 8.0]$, the test pressures are chosen to be $p' = 2.2, 4.2, \text{ and } 6.2$, which avoids extrapolations. The predicted displacements and exact displacements, obtained with the heterogeneous constitutive law are shown in Fig. 16. Good agreement can be observed. The estimated variance of the reduced coordinate is $\hat{\Sigma}_\lambda \approx 2.23 \times 10^{-6}$. The confidence intervals corresponding to the quantiles 0.95 and 0.05 is constructed with 2000 samples generated by Eq. (54). The exact solution lies in the uncertainty region, which demonstrates the effectiveness of the presented framework.

It is also interesting to visualize the uncertainties for the maximum stress, we have used as the QoI. Figure 17 shows the maximum stresses together with its $3\delta$ confidence interval for both the uniform constitutive law case and non-uniform constitutive law case. We can clearly see that the exact maximum stresses lie in
Figure 13: The uniform constitutive law case: the predicted (dash red line) and the exact displacements (blue line) in the \( z \) and \( r \) directions (left to right) of the rubber membrane, subjected to pressure loads 2.2, 4.2, 6.2, and 8.2 (top to bottom), and their corresponding confidence intervals (blue region) corresponding to the quantiles 0.95 and 0.05.
the confidence interval in all cases. For the non-uniform constitutive law case, we have larger uncertainty error bounds, which is consistent with our intuition since the uniform constitutive law case does not have homogenization errors.

Figure 15: The exact constitutive law (reference) and the surrogate constitutive law learned by the neural network approach for the rubber membrane in the non-uniform constitutive law case.
Figure 16: The non-uniform constitutive law case: the predicted (dashed red line) and the exact displacements (blue line) in the \( z \) and \( r \) directions (left to right) of the rubber membrane, subjected to pressure loads 2.2, 4.2, and 6.2 (top to bottom), and their corresponding confidence intervals (blue region) corresponding to the quantiles 0.95 and 0.05.
6.3. Comparison with Piecewise Linear Response Surface

The constitutive relation Eq. (56) can also be approximated by a response surface. In this section, the proposed neural network approach is compared with the response surface approach, in which the response surface is parameterized by piecewise linear basis functions. In practice, it is impossible to adopt a point-to-point response surface fitting since the stress data is not directly available from the final output. Although in our case the strain distribution from the training data can be obtained, it can be arbitrarily nonuniform and will change on a case-by-case basis. For uniform grids and piecewise linear response surface, either the grid size \( h \) is large and we have large errors, or \( h \) is small and we overfit. In both cases, the approximation properties are poor. Adaptive triangulation (e.g., Delaunay triangulation) is possible but difficult to implement in a robust manner. Besides, global basis methods such as radial basis functions suffer from numerical difficulties due to ill-conditioned coefficient matrices. However, we show that neural networks lend us a universal method for approximating the constitutive law without any prior information on the data distribution. In addition, it shows robustness in terms of noise and generalizes better than fine-tuned piecewise linear response surface approaches. Consequently, the neural network approach has the potential to achieve state-of-the-art results using an end-to-end data-driven modeling.

For comparison, we use \texttt{L-BFGS-B} optimizer for both cases. For the piecewise linear response surface (PLRS-\( h \)), we use a uniform triangulation on \([0, 20]^2\) with mesh size \( h = 0.4, 1.0, 2.0 \) where the superscript 2 corresponds to the dimension of the input strain variables. The parametric domain and the mesh size are chosen based on the Fig. 18 which depicts all the principal stretch pairs \((\lambda_1, \lambda_2)\) post-processed from the training set for both uniform constitutive law and non-uniform constitutive law cases. The neural network (NN) is a standard fully-connected deep nets with two 20-neuron hidden layers. The number of parameters is shown in Table 2.

| Model     | PLRS-0.4 | PLRS-1.0 | PLRS-2.0 | NN   |
|-----------|----------|----------|----------|------|
| Degrees of freedom | 5000     | 400      | 100      | 520  |

Table 2: Number of parameters for different surrogate functions.

The optimization is terminated when the objective function is called 15000 times. The training data consists of 17 samples as mentioned before. The test data consists of three samples subjected to non-dimensional pressure loads of \( p = 2.2, 4.2, 6.2 \). Figure 19 shows the losses at each training iteration, including the losses evaluated on both the training set and the test set. PLRS-0.4 achieves the least training...
loss, thanks to the large number of local degrees of freedom in the response surface, which well fit the unknown functions on these training data in Fig. 18. However, PLRS-0.4 fails to predict or approximate the stresses associated to principal stretch pairs \((\lambda_1, \lambda_2)\) that do not appear in the training data. These stresses fail to be updated during the training process, therefore, the calibrated response surfaces are quite rough and highly oscillating, which is illustrated in Fig. 20-right. Certainly, the overfitting leads to poor performance on the test set. PLRS-1.0 and PLRS-2.0 reach plateaus rapidly, which indicates under-fitting, i.e. failing to capture the underlying trend of the data (See Fig. 20). Hence, they also lead to poor performance in the test set. And it is worth mentioning PLRS-1.0 reaches similar training error in the non-uniform case as NN, but struggles to generalize well on the test data.

NN achieves consistent losses on both the training set and the test set. And the losses are reduced by about four orders of magnitude during the training process. We believe NN is able to regularize the surrogate function and consistently interpolate on the missing input states, hence leads to smooth constitutive relations (See Fig. 12 and Fig. 15). And NN can be more efficient for high dimensional problems, compared with the response surface approach, for which the number of parameters depends on the input parameter dimensions exponentially.

Figure 18: The principal stretch pairs \((\lambda_1, \lambda_2)\) that appear in the training set for uniform and nonuniform case and the triangulation of the parametric domain with \(h = 2.0\).

Figure 19: The losses evaluated on both the training set and the test set at each training step with response surface approaches (PLRS-0.4, PLRS-1.0 and PLRS-2.0) and the neural network approach (NN) for the uniform constitutive law case (left) and the non-uniform constitutive law case (right).
Moreover, we also study the sensitivity to initial weights \cite{65} of these two approaches, since both approaches involve highly non-convex optimization problems. We start from several initial guesses, the i.i.d. Gaussian random variables with mean 0 and standard deviation 100, for both PLRS and NN approaches (We also tried standard deviation 10 and found no substantial difference.). Figure \ref{fig:21} depicts that PLRS is quite robust with respect to different initial guesses for all loss curves overlap well on each other. However, all these initial guesses lead to poor performance on the test set. Meanwhile, the NN approach shows good generalization property, i.e. consistent losses on both training set and test set, even in the non-uniform constitutive law case.

7. Conclusion

Data-driven approaches continue to gain popularity for constructing coarse-grained models, when data from high-fidelity simulations and high resolution experiments become richer. In this work, a general framework combining traditional FEM and the neural network for predictive modeling is presented. The proposed framework discretizes the physical system through FEM, but replaces the constitutive law, or any coarse-grained model part, with a black-box neural network. The neural network is trained based on global response
information, for example the displacement field, instead of point-to-point strain vs stress data. In view of
the fact that the global response data contains plenty of strain-stress informations and such data are eas-
er to measure from experiments. Furthermore, the applicability and accuracy of the present framework is
analyzed. When the data set contains comprehensive constitutive data, and the relation is smooth enough,
the error of constitutive law predicted by the neural network error is bounded by the optimization error
and the discretization error. The uncertainties due to heterogeneity of the material is quantified efficiently
in the FEM-neural network framework. And the framework is tested on a multi-scale fiber reinforced thin
plate problem and a highly nonlinear rubbery membrane problem. It is worth mentioning that in the rub-
bery membrane problem, the comparison of neural network as a surrogate model to the response surface
is presented, the strength of the neural network approach, i.e. its good regularization and generalization
properties, is highlighted.

An interesting area for future work would be to extend current framework to partially observed data.
Examples in the present work uses the whole displacement filed, however, part of the displacement field,
saying only the displacement field on the boundary of the 3D bulk, should be enough to train the neural
network. It would also be interesting to extend current framework for time-dependent physical systems.
In such cases, it may be possible to treat the forward propagation as a standalone operator and use the
adjoint state method to derive the gradient of the operator. In this way, the neural network approximation
to unknown functions is decoupled from the sophisticated numerical simulations. And more challenging
problems, like damage mechanics, will be explored in the future.

Acknowledgements

Daniel Z. Huang thanks Prof. Peter Pinsky for providing the model problems in our numerical ex-
periments. Kailai Xu thanks the Stanford Graduate Fellowship in Science & Engineering and the 2018
Schlumberger Innovation Fellowship for the financial support.
References

[1] David M Trujillo and Henry R Busby. *Practical inverse analysis in engineering*, volume 7. CRC press, 1997.

[2] Eric Dow and Qiqi Wang. Quantification of structural uncertainties in the k-w turbulence model. In *52nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 19th AIAA/ASME/AHS Adaptive Structures Conference 13t*, page 1762, 2011.

[3] Jian-Xun Wang, Jin-Long Wu, and Heng Xiao. Physics-informed machine learning approach for reconstructing reynolds stress modeling discrepancies based on dns data. *Physical Review Fluids*, 2(3):034603, 2017.

[4] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Machine learning of linear differential equations using gaussian processes. *Journal of Computational Physics*, 348:683–693, 2017.

[5] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.

[6] Michele Milano and Petros Koumoutsakos. Neural network modeling for near wall turbulent flow. *Journal of Computational Physics*, 182(1):1–26, 2002.

[7] Julia Ling, Andrew Kurzawski, and Jeremy Templeton. Reynolds averaged turbulence modeling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016.

[8] J Ghaboussi, JH Garrett Jr, and Xiping Wu. Knowledge-based modeling of material behavior with neural networks. *Journal of engineering mechanics*, 117(1):132–153, 1991.

[9] GW Ellis, C Yao, Rui Zhao, and Df Penumadu. Stress-strain modeling of sands using artificial neural networks. *Journal of geotechnical engineering*, 121(5):429–435, 1995.

[10] Yuelin Shen, K Chandrashekhara, WF Breig, and LR Oliver. Finite element analysis of v-ribbed belts using neural network based hyperelastic material model. *International Journal of Non-Linear Mechanics*, 40(6):875–890, 2005.

[11] BA Le, Julien Yvonnet, and Q-C He. Computational homogenization of nonlinear elastic materials using neural networks. *International Journal for Numerical Methods in Engineering*, 104(12):1061–1084, 2015.

[12] Julia Ling, Reese Jones, and Jeremy Templeton. Machine learning strategies for systems with invariance properties. *Journal of Computational Physics*, 318:22–35, 2016.

[13] Tomonari Furukawa and Genki Yagawa. Implicit constitutive modelling for viscoplasticity using neural networks. *International Journal for Numerical Methods in Engineering*, 43(2):195–219, 1998.

[14] Kun Wang and WaiChing Sun. A multiscale multi-permeability poroplasticity model linked by recursive homogenizations and deep learning. *Computer Methods in Applied Mechanics and Engineering*, 334:337–380, 2018.

[15] Zvi Hashin. Analysis of composite materials—a survey. *Journal of Applied Mechanics*, 50(3):481–505, 1983.

[16] CT Sun and RS Vaidya. Prediction of composite properties from a representative volume element. *Composites Science and Technology*, 56(2):171–179, 1996.

[17] Frédéric Feyel and Jean-Louis Chaboche. Fe2 multiscale approach for modelling the elastoviscoplastic behaviour of long fibre sic/ti composite materials. *Computer methods in applied mechanics and engineering*, 183(3-4):309–330, 2000.
[18] T Kanit, S Forest, Ia Galliet, Va Mounoury, and D Jeulin. Determination of the size of the representative volume element for random composites: statistical and numerical approach. *International Journal of solids and structures*, 40(13-14):3647–3679, 2003.

[19] Zheng Yuan and Jacob Fish. Toward realization of computational homogenization in practice. *International Journal for Numerical Methods in Engineering*, 73(3):361–380, 2008.

[20] Yves Surrel. Moiré and grid methods: a signal-processing approach. In *Interferometry’94: photomechanics*, volume 2342, pages 118–128. International Society for Optics and Photonics, 1994.

[21] Christian E Willert and Morteza Gharib. Digital particle image velocimetry. *Experiments in fluids*, 10(4):181–193, 1991.

[22] CJ Elkins, M Markl, N Pelc, and JK Eaton. 4d magnetic resonance velocimetry for mean velocity measurements in complex turbulent flows. *Experiments in Fluids*, 34(4):494–503, 2003.

[23] Christopher J Elkins and Marcus T Alley. Magnetic resonance velocimetry: applications of magnetic resonance imaging in the measurement of fluid motion. *Experiments in Fluids*, 43(6):823–858, 2007.

[24] Michel Grediac, Fabrice Pierron, Stéphane Avril, and Evelyne Toussaint. The virtual fields method for extracting constitutive parameters from full-field measurements: a review. *Strain*, 42(4):233–253, 2006.

[25] Stéphane Avril, Marc Bonnet, Anne-Sophie Bretelle, Michel Grédiac, François Hild, Patrick Ienny, Félix Latourte, Didier Lemosse, Stéphane Pagano, Emmanuel Pagnacco, et al. Overview of identification methods of mechanical parameters based on full-field measurements. *Experimental Mechanics*, 48(4):381, 2008.

[26] Giuseppe Geymonat, François Hild, and Stéphane Pagano. Identification of elastic parameters by displacement field measurement. *Comptes Rendus Mecanique*, 330(6):403–408, 2002.

[27] Xia-Ting Feng and Chengxiang Yang. Genetic evolution of nonlinear material constitutive models. *Computer Methods in Applied Mechanics and Engineering*, 190(45):5957–5973, 2001.

[28] Alexandre M Tartakovsky, Carlos Ortiz Marrero, D Tartakovsky, and David Barajas-Solano. Learning parameters and constitutive relationships with physics informed deep neural networks. *arXiv preprint arXiv:1808.03398*, 2018.

[29] John P Boyd. *Chebyshev and Fourier spectral methods*. Courier Corporation, 2001.

[30] Claudio Canuto, M Yousuff Hussaini, Alfio Quarteroni, and Thomas A Zang. *Spectral methods*. Springer, 2006.

[31] Claudio Canuto, M Yousuff Hussaini, Alfio Quarteroni, A Thomas Jr, et al. *Spectral methods in fluid dynamics*. Springer Science & Business Media, 2012.

[32] David A Field. Laplacian smoothing and delaunay triangulations. *Communications in applied numerical methods*, 4(6):709–712, 1988.

[33] Der-Tsai Lee and Bruce J Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Computer & Information Sciences*, 9(3):219–242, 1980.

[34] Jean-Paul Berrut and Lloyd N Trefethen. Barycentric lagrange interpolation. *SIAM review*, 46(3):501–517, 2004.

[35] John C Mason and David C Handscomb. *Chebyshev polynomials*. Chapman and Hall/CRC, 2002.

[36] A De Boer, MS Van der Schoot, and Hester Bijl. Mesh deformation based on radial basis function interpolation. *Computers & structures*, 85(11-14):784–795, 2007.
[37] Jooyoung Park and Irwin W Sandberg. Universal approximation using radial-basis-function networks. *Neural computation*, 3(2):246–257, 1991.

[38] Shmuel Rippa. An algorithm for selecting a good value for the parameter c in radial basis function interpolation. *Advances in Computational Mathematics*, 11(2-3):193–210, 1999.

[39] Robert Schaback. Error estimates and condition numbers for radial basis function interpolation. *Advances in Computational Mathematics*, 3(3):251–264, 1995.

[40] Cameron Thomas Mouat and Richard Keith Beatson. Rbf collocation. 2002.

[41] Marc G Genton and William Kleiber. Cross-covariance functions for multivariate geostatistics. *Statistical Science*, pages 147–163, 2015.

[42] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[43] Michael L Stein. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 2012.

[44] Wim Van Beers and Jack PC Kleijnen. Kriging interpolation in simulation: a survey. In *Proceedings of the 36th conference on Winter simulation*, pages 113–121. Winter Simulation Conference, 2004.

[45] Christopher KI Williams and Carl Edward Rasmussen. Gaussian processes for regression. In *Advances in neural information processing systems*, pages 514–520, 1996.

[46] J Bernardo, J Berger, A Dawid, A Smith, et al. Regression and classification using gaussian process priors. *Bayesian statistics*, 6:475, 1998.

[47] Mark N Gibbs. *Bayesian Gaussian processes for regression and classification*. PhD thesis, Citeseer, 1998.

[48] Christopher KI Williams and David Barber. Bayesian classification with gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

[49] George G Lorentz, Manfred von Golitschek, and Yuly Makovoz. *Constructive approximation: advanced problems*, volume 304. Springer Berlin, 1996.

[50] Hrushikesh N Mhaskar. Neural networks for optimal approximation of smooth and analytic functions. *Neural computation*, 8(1):164–177, 1996.

[51] Kailai Xu and Eric Darve. Calibrating Lévy process from observations based on neural networks and automatic differentiation with convergence proofs. *arXiv preprint arXiv:1812.08883*, 2018.

[52] Kailai Xu and Eric Darve. The neural network approach to inverse problems in differential equations. *arXiv e-prints*, page arXiv:1901.07758, January 2019.

[53] E Weinan, Jiequn Han, and Arnulf Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, 2017.

[54] Bernardo Llanas, Sagrario Lantarón, and Francisco J Sáinz. Constructive approximation of discontinuous functions by neural networks. *Neural Processing Letters*, 27(3):209–226, 2008.

[55] Trenton Kirchdoerfer and Michael Ortiz. Data-driven computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 304:81–101, 2016.

[56] Trenton Kirchdoerfer and Michael Ortiz. Data driven computing with noisy material data sets. *Computer Methods in Applied Mechanics and Engineering*, 326:622–641, 2017.
[57] Dongbin Xiu and George Em Karniadakis. The wiener–askey polynomial chaos for stochastic differential equations. *SIAM journal on scientific computing*, 24(2):619–644, 2002.

[58] Dongbin Xiu and George Em Karniadakis. Modeling uncertainty in flow simulations via generalized polynomial chaos. *Journal of computational physics*, 187(1):137–167, 2003.

[59] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.

[60] Marc C Kennedy and Anthony O’Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, 2001.

[61] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[62] Rohit K. Tripathy and Ilias Bilionis. Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification. *Journal of Computational Physics*, 375:565 – 588, 2018.

[63] Paul G Constantine, Eric Dow, and Qiqi Wang. Active subspace methods in theory and practice: applications to kriging surfaces. *SIAM Journal on Scientific Computing*, 36(4):A1500–A1524, 2014.

[64] Isaac Fried. Finite element computation of large rubber membrane deformations. *International Journal for Numerical Methods in Engineering*, 18(5):653–660, 1982.

[65] Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv e-prints*, page arXiv:1511.06422, Nov 2015.