Science Arts & Métiers (SAM)
is an open access repository that collects the work of Arts et Métiers Institute of Technology researchers and makes it freely available over the web where possible.

This is an author-deposited version published in: https://sam.ensam.eu
Handle ID: .http://hdl.handle.net/10985/19924

To cite this version :

Quercus HERNÁNDEZ, Alberto BADÍAS, David GONZÁLEZ, Elías CUETO, Francisco CHINESTA SORIA - Structure-preserving neural networks - Journal of Computational Physics - Vol. 426, p.1-16 - 2021

Any correspondence concerning this service should be sent to the repository Administrator : scienceouverte@ensam.eu

Structure-preserving neural networks

Quercus Hernández [a], Alberto Badías [a], David González [a], Francisco Chinesta [b], Elías Cueto [a,*]

[a] Aragon Institute of Engineering Research (I3A), Universidad de Zaragoza, Maria de Luna 3, E-50018 Zaragoza, Spain
[b] ESI Chair and PIMM Lab, ENSAM ParisTech, 155 Boulevard de l'Hôpital, 75013 Paris, France

Keywords:
Scientific machine learning
Neural networks
Structure preservation
GENERIC

A B S T R A C T

We develop a method to learn physical systems from data that employs feedforward neural networks and whose predictions comply with the first and second principles of thermodynamics. The method employs a minimum amount of data by enforcing the metriplectic structure of dissipative Hamiltonian systems in the form of the so-called General Equation for the Non-Equilibrium Reversible-Irreversible Coupling, GENERIC (Öttinger and Grmela (1997) [36]). The method does not need to enforce any kind of balance equation, and thus no previous knowledge on the nature of the system is needed. Conservation of energy and dissipation of entropy in the prediction of previously unseen situations arise as a natural by-product of the structure of the method.
Examples of the performance of the method are shown that comprise conservative as well as dissipative systems, discrete as well as continuous ones.

1. Introduction

With the irruption of the so-called fourth paradigm of science [17] a growing interest is detected on the machine learning of scientific laws. A plethora of methods have been developed that are able to produce more or less accurate predictions about the response of physical systems in previously unseen situations by employing techniques ranging from classical regression to the most sophisticated deep learning methods.

For instance, recent works in solid mechanics have substituted the constitutive equations with experimental data [1,25], while conserving the traditional approach on physical laws with high epistemic value (i.e., balance equations, equilibrium). Similar approaches have applied this concept to the unveiling (or correction) of plasticity models [21], while others created the new concept of constitutive manifold [20,22]. Other approaches are designed to unveil an explicit, closed form expression for the physical law governing the phenomenon at hand [3].

An interest is observed in the incorporation of the already existing scientific knowledge to these data-driven procedures. This interest is two-fold. Indeed, we prefer not to get rid of centuries of scientific knowledge and rely exclusively on powerful

☆ This project has been partially funded by the ESI Group through the ESI Chair at ENSAM ParisTech and through the project "Simulated Reality: an intelligence augmentation system based on hybrid twins and augmented reality" at the University of Zaragoza. The support of the Spanish Ministry of Economy and Competitiveness through grant number CICYT-DPI2017-85139-C2-1-R and by the Regional Government of Aragon, through the project T24_20R, and the European Social Fund, are also gratefully acknowledged.

* Corresponding author.
E-mail addresses: quercus@unizar.es (Q. Hernández), abadias@unizar.es (A. Badías), gonzal@unizar.es (D. González), francisco.chinesta@ensam.eu

machine learning strategies. Existing theories have proved to be useful in the prediction of physical phenomena and are still in the position of helping to produce very accurate predictions. This is the procedure followed in the so-called data-driven computational mechanics approach mentioned before. On the other hand, these theories help to keep the consumption of data to a minimum. Data are expensive to produce and to maintain. Already existing scientific knowledge could alleviate the amount of data needed to produce a successful prediction.

The mentioned works on data-driven computational mechanics usually rely on traditional machine learning algorithms, which are very precise and tested but usually computationally expensive. With the recent advances in data processing, computing resources and machine learning, neural networks have become a powerful tool to analyze traditionally hard problems such as image classification [26,45], speech recognition [12,18] or data compressing [42,46]. These new machine learning methods outperform many of the traditional ones, both in modeling capacity and computational time (once trained, certain neural networks can easily handle real time requirements). Recent works in the machine learning community [29, 31,34] have shown that neural networks are also versatile in constraint optimizations.

This is the approach followed by several authors in the context of physical simulations, which aim to solve a set of partial differential equations (PDEs) in complex dynamical systems. Physical problems must satisfy inherently certain conditions dictated by physics, often formulated as conservation laws, and can be imposed to a neural network using extra loss terms in the constrained optimization process [32].

Similar constraints are imposed in the so-called physically-informed neural networks approach [40,48]. This family of methods employs neural networks to solve highly nonlinear partial differential equations (PDEs) resulting in very accurate and numerically stable results. However, they rely on prior knowledge of the governing equations of the problem.

The authors have introduced the so-called thermodynamically consistent data-driven computational mechanics [7,9,10]. Unlike other existing works, this approach does not impose any particular balance equation to solve for. Instead, it relies on the imposition of the right thermodynamic structure of the resulting predictions, as dictated by the so-called GENERIC formalism [15]. As will be seen, this ensures conservation of energy and the right amount of entropy dissipation, thus giving rise to predictions satisfying the first and second principles of thermodynamics. These techniques, however, employ regression to unveil the thermodynamic structure of the problem at the sampling points. For previously unseen situations, they employ interpolation on the matrix manifold describing the system.

Recent works in symplectic networks [23] have by-passed those drawbacks by exploiting the mathematical properties of Hamiltonian systems, so no prior knowledge of the system is required. However, this technique only operates on conservative systems with no entropy generation.

The aim of this work is the development of a new structure-preserving neural network architecture capable of predicting the time evolution of a system based on experimental observations on the system, with no prior knowledge of its governing equations, to be valid for both conservative and dissipative systems. The key idea is to merge the proven computational power of neural networks in highly nonlinear physics with thermodynamic consistent data-driven algorithms. The resulting methodology, as will be seen, is a powerful neural network architecture, conceptually very simple—based on standard feed-forward methodologies—that exploits the right thermodynamic structure of the system as unveiled from experimental data, and that produces interpretable results [33].

The outline of the paper is as follows. A brief description of the problem setup is presented in Section 2. Next, in Section 3, the methodology is presented of both the GENERIC formalism and the feed-forward neural networks used to solve the stated problem. This technique is used in different physical systems of increasing complexity: a double thermo-elastic pendulum (Section 4) and a Couette flow in a viscoelastic fluid (Section 5). The paper is completed with a discussion in Section 6.

2. Problem statement

Weinan E seems to be the first author in interpreting the process of learning physical systems as the solution of a dynamical system [6]. Consider a system whose governing variables will be hereafter denoted by $\boldsymbol{z} \in \mathcal{M} \subseteq \mathbb{R}^n$, with $\mathcal{M}$ the state space of these variables, which is assumed to have the structure of a differentiable manifold in $\mathbb{R}^n$.

The problem of learning a given physical phenomenon can thus be seen as the one of finding an expression for the time evolution of their governing variables $\boldsymbol{z}$,

$$\dot{\boldsymbol{z}} = \frac{d\boldsymbol{z}}{dt} = F(\boldsymbol{x}, \boldsymbol{z}, t),\ \boldsymbol{x} \in \Omega \in \mathbb{R}^D,\ t \in \mathcal{I} = (0, T],\ \boldsymbol{z}(0) = \boldsymbol{z}_0, \tag{1}$$

where $\boldsymbol{x}$ and t refer to the space and time coordinates within a domain with $D = 2, 3$ dimensions. $F(\boldsymbol{x}, \boldsymbol{z}, t)$ is the function that gives, after a prescribed time horizon T, the flow map $\boldsymbol{z}_0 \to \boldsymbol{z}(\boldsymbol{z}_0, T)$.

While this problem can be seen as a general supervised learning problem (we fix both $\boldsymbol{z}_0$ and $\boldsymbol{z}$), when we have additional information about the physics being represented by the sought function F, it is legitimate to try to include it in the search procedure. W. E seems to have been the first in suggesting to impose a Hamiltonian structure on F if we know that energy is conserved, for instance [6]. Very recently, two different approaches follow this same rationale [2,23].

For conservative systems, therefore, imposing a Hamiltonian structure seems a very appealing way to obtain thermodynamics-aware results. However, when the system is dissipative, this method does not provide with valid results. Given

the importante of dissipative phenomena (viscous solids, fluid dynamics, ...) we explore the right thermodynamic structure to impose to the search methodology.

The goal of this paper is to develop a new method of solving Eq. (1) using state of the art deep learning tools, in order to predict the time evolution of the state variables of a given system. The solution is forced to fulfill the basic thermodynamic requirements of energy conservation and entropy inequality restrictions via the GENERIC formalism, presented in the next section.

3. Methodology

In this section we develop the appropriate thermodynamic structure for dissipative systems. Classical systems modeling can be done at a variety of scales. We could think of the most detailed (yet often impractical) scale of molecular dynamics, where energy conservation applies and the Hamiltonian paradigm can be imposed. However, the number of degrees of freedom and, noteworthy, the time scale, renders this approach of little interest for many applications. On the other side of the spectrum lies thermodynamics, where only conserved, invariant, quantities are described and thus there is no need for conservation principles. At any other (mesoscopic) scale, unresolved degrees of freedom give rise to the appearance of fluctuation in the results (or its equivalent, dissipation). At these scales, traditional modeling procedures imply expressing physical insights in the form of governing equations [13]. These equations are then validated from experimental observations.

Alternatively, thermodynamics can be thought of as a *meta-physics*, in the sense that it is actually a theory of theories [14]. It provides us with the right theoretic framework in which basic principles are met. And, in particular for any of these intermediate or mesoscopic scales, a so-called metriplectic structure emerges. The term metriplectic comes for the combination of symplectic and Riemannian (metric) geometry and emphasizes the fact that there are conservative as well as dissipative contributions to the general evolution of such a system. Once such a geometric structure is found for the system, we are in the position of fixing the framework in which our neural networks can look for the adequate prediction of the future states of the system. The particular metriplectic structure that we employ for such a task is known, as stated before, as GENERIC.

3.1. The GENERIC formalism

The "General Equation for Non-Equilibrium Reversible-Irreversible Coupling", GENERIC, formalism [15,36] establishes a mathematical framework in order to model the dynamics of a system. Furthermore, it is compatible with classical equilibrium thermodynamics [35], preserving the symmetries of the system as stated in Noether's theorem. It has served as the basis for the development of several consistent numerical integration algorithms that exploit these desirable properties [11,41].

The GENERIC structure for the evolution in Eq. (1) is obtained after finding two algebraic or differential operators

$$\boldsymbol{L} : T^*\mathcal{M} \to T\mathcal{M}, \quad \boldsymbol{M} : T^*\mathcal{M} \to T\mathcal{M},$$

where $T^*\mathcal{M}$ and $T\mathcal{M}$ represent, respectively, the cotangent and tangent bundles of $\mathcal{M}$. As in general Hamiltonian systems, there will be an energy potential, which we will denote hereafter by $E(\boldsymbol{z})$. In order to take into account the dissipative effects, a second potential (the so-called Massieu potential) is introduced in the formulation. It is, of course, the entropy potential of the GENERIC formulation, $S(\boldsymbol{z})$. With all these ingredients, we arrive at a description of the dynamics of the system of the type

$$\frac{d\boldsymbol{z}}{dt} = \boldsymbol{L}\frac{\partial E}{\partial \boldsymbol{z}} + \boldsymbol{M}\frac{\partial S}{\partial \boldsymbol{z}}. \tag{2}$$

As shown in Eq. (2), the time evolution of the system described by the nonlinear operator $F(\boldsymbol{x}, \boldsymbol{z}, t)$ presented in Eq. (1) is now split in two separated terms:

- **Reversible Term**: It accounts for all the reversible (non-dissipative) phenomena of the system. In the context of classical mechanics, this term is equivalent to Hamilton's equations of motion that relates the particle position and momentum. The operator $\boldsymbol{L}(\boldsymbol{z})$ is the Poisson matrix—it defines a Poisson bracket—and is required to be skew-symmetric (a cosymplectic matrix).
- **Non-Reversible Term**: The rest of the non-reversible (dissipative) phenomena of the system are modeled here. The operator $\boldsymbol{M}(\boldsymbol{z})$ is the friction matrix and is required to be symmetric and positive semi-definite.

The GENERIC formulation of the problem is completed with the following so-called degeneracy conditions

$$\boldsymbol{L}\frac{\partial S}{\partial \boldsymbol{z}} = \boldsymbol{M}\frac{\partial E}{\partial \boldsymbol{z}} = \boldsymbol{0}. \tag{3}$$

The first condition expresses the reversible nature of the $\boldsymbol{L}$ contribution to the dynamics whereas the second requirement expresses the conservation of the total energy by the $\boldsymbol{M}$ contribution. This means no other thing that the energy potential

does not contribute to the production of entropy and, conversely, that the entropy functional does not contribute to reversible dynamics. This mutual degeneracy requirement in addition to the already mentioned $\boldsymbol{L}$ and $\boldsymbol{M}$ matrix requirements ensure that:

$$\frac{\partial E}{\partial t} = \frac{\partial E}{\partial \boldsymbol{z}} \cdot \frac{\partial \boldsymbol{z}}{\partial t} = \frac{\partial E}{\partial \boldsymbol{z}} \left(\boldsymbol{L} \frac{\partial E}{\partial \boldsymbol{z}} + \boldsymbol{M} \frac{\partial S}{\partial \boldsymbol{z}} \right) = 0,$$

which expresses the conservation of energy in an isolated system, also known as the first law of thermodynamics. Applying the same reasoning to the entropy S:

$$\frac{\partial S}{\partial t} = \frac{\partial S}{\partial \boldsymbol{z}} \cdot \frac{\partial \boldsymbol{z}}{\partial t} = \frac{\partial S}{\partial \boldsymbol{z}} \left(\boldsymbol{L} \frac{\partial E}{\partial \boldsymbol{z}} + \boldsymbol{M} \frac{\partial S}{\partial \boldsymbol{z}} \right) = \frac{\partial S}{\partial \boldsymbol{z}} \boldsymbol{M} \frac{\partial S}{\partial \boldsymbol{z}} \geq 0,$$

which guarantees the entropy inequality, this is, the second law of thermodynamics.

3.2. Proposed integration algorithm

Once the learning procedure is accomplished, our neural network is expected to integrate the system dynamics in time, given previously unseen initial conditions. In order to numerically solve the GENERIC equation, we formulate the discretized version of Eq. (2) following previous works [11]:

$$\frac{\boldsymbol{z}_{n+1} - \boldsymbol{z}_n}{\Delta t} = \mathsf{L} \cdot \frac{\mathsf{D}E}{\mathsf{D}\boldsymbol{z}} + \mathsf{M} \cdot \frac{\mathsf{D}S}{\mathsf{D}\boldsymbol{z}}. \tag{4}$$

The time derivative of the original equation is discretized with a forward Euler scheme in time increments Δt, where $\boldsymbol{z}_{n+1} = \boldsymbol{z}_{t+\Delta t}$. L and M are the discretized versions of the Poisson and friction matrices. Last, $\frac{\mathsf{D}E}{\mathsf{D}\boldsymbol{z}}$ and $\frac{\mathsf{D}S}{\mathsf{D}\boldsymbol{z}}$ represent the discrete gradients, which can be approximated in a finite element sense as:

$$\frac{\mathsf{D}E}{\mathsf{D}\boldsymbol{z}} \simeq \boldsymbol{A}\boldsymbol{z}, \quad \frac{\mathsf{D}S}{\mathsf{D}\boldsymbol{z}} \simeq \boldsymbol{B}\boldsymbol{z}, \tag{5}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ represent the discrete matrix form of the gradient operators.

Finally, manipulating algebraically Eq. (4) with Eq. (5) and including the degeneracy conditions of Eq. (3), the proposed integration scheme for predicting the dynamics of a physical system is the following

$$\boldsymbol{z}_{n+1} = \boldsymbol{z}_n + \Delta t \left(\mathsf{L} \cdot \boldsymbol{A}\boldsymbol{z}_n + \mathsf{M} \cdot \boldsymbol{B}\boldsymbol{z}_n \right) \tag{6}$$

subject to:

$$\mathsf{L} \cdot \boldsymbol{B}\boldsymbol{z}_n = \boldsymbol{0},$$
$$\mathsf{M} \cdot \boldsymbol{A}\boldsymbol{z}_n = \boldsymbol{0},$$

ensuring the thermodynamical consistency of the resulting model.

To sum up, the main objective of this work is to compute the form of the $\boldsymbol{A}(\boldsymbol{z})$ and $\boldsymbol{B}(\boldsymbol{z})$ gradient operator matrices, subject to the degeneracy conditions, in order to integrate the initial system state variables $\boldsymbol{z}_0$ over certain time steps Δt of the time interval $\mathcal{I}$. Usually, the form of matrices $\boldsymbol{L}$ and $\boldsymbol{M}$ is known in advance, given the vast literature in the field. If necessary, these terms can also be computed [11].

3.3. Feed-forward neural networks

In the introduction we already mentioned the intrinsic power of neural networks in many fields. The main reason under the fact that neural networks are able to learn and reproduce such a variety of problems is that they are considered to be universal approximators [4,19], meaning that they are capable of approximating any measurable function to any desired degree of accuracy. The main limitation of this technique is the correct selection of the tuning parameters of the network, also called hyperparameters.

Another universal approximator are polynomials, as they can approximate any infinitely differentiable function as a Taylor power series expansion. The main difference is that neural networks rely on composition of functions rather than sum of power series:

$$\hat{\boldsymbol{y}} = (f^{[L]} \circ f^{[L-1]} \circ \ldots \circ f^{[l]} \circ \ldots \circ f^{[2]} \circ f^{[1]})(\boldsymbol{x}). \tag{7}$$

Eq. (7) shows that the desired output $\hat{\boldsymbol{y}}$ from a defined input $\boldsymbol{x}$ of a neural network is a composition of different functions $f^{[l]}$ as building blocks of the network in L total layers. The challenge is to select the best combination of functions in the correct order such that it approximates the solution of the studied problem.

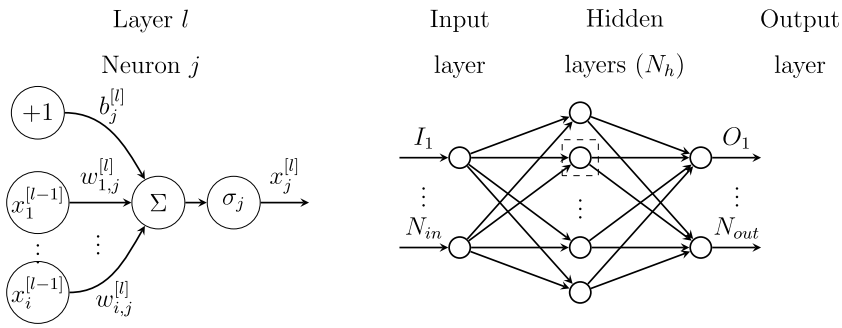

Fig. 1. Representation of a single neuron (left) as a part of a fully connected neural net (right).

The simplest building block of artificial deep neural network architectures is the neuron or perceptron (Fig. 1, left). Several neurons are stacked in a multilayer perceptron (MLP), which is mathematically defined as follows

$$\boldsymbol{x}^{[l]} = \sigma(\boldsymbol{w}^{[l]}\boldsymbol{x}^{[l-1]} + \boldsymbol{b}^{[l]}), \tag{8}$$

where l is the index of the current layer, $\boldsymbol{x}^{[l-1]}$ and $\boldsymbol{x}^{[l]}$ are the layer input and output vector respectively, $\boldsymbol{w}^{[l]}$ is the weight matrix of the last layer, $\boldsymbol{b}^{[l]}$ is the bias vector of the last layer and σ is the activation function. If no activation function is applied, the MLP is equivalent to a linear operator. However, σ is chosen to be a nonlinear function in order to increase the capacity of modeling more complex problems, which are commonly nonlinear. In classification problems, the traditional activation function is the logistic function (sigmoid) whereas in regression problems, Rectified Linear Unit (ReLU) [8] or hyperbolic tangent are commonly used.

In this work, we use a deep neural network architecture known as feed-forward neural network [44]. It consists of a several layer of multilayer perceptrons with no cyclic connections, as shown in Fig. 1 (right).

The input of the neural net is the vector state of a given time step $\boldsymbol{z}_n$, and the outputs are the concatenated GENERIC matrices $\boldsymbol{A}_n^{\text{net}}$ and $\boldsymbol{B}_n^{\text{net}}$: for a system with n state variables the number of inputs and outputs are $N_{\text{in}} = n$ and $N_{\text{out}} = 2n^2$. Then, using the GENERIC integration scheme, the state vector at the next time step $\boldsymbol{z}_{n+1}^{\text{net}}$ is obtained. This method is repeated for the whole simulation time T with a total of N_T snapshots.

The state variables of a general dynamical system may differ in several orders of magnitude from each other, due to their own physical nature or measurement units. Then, a pre-processing of the input data (scaling or normalization) can improve the model performance and stability.

The number of hidden layers N_h depends on the complexity of the problem. Increasing the net size raises the computational power of the net to model more complex phenomena. However, it slows the training process and could lead to data overfitting, limiting its generalization and extrapolation capacity. The size of the hidden layers is chosen to be the same as the output size of the net N_{out}.

The cost function for our neural network is composed of three different terms:

- **Data loss**: The main loss condition is the agreement between the network output and the real data. It is computed as the squared error sum, computed between the predicted state vector $\boldsymbol{z}_{n+1}^{\text{net}}$ and the ground truth solution $\boldsymbol{z}_{n+1}^{\text{GT}}$ for each time step.

$$\mathcal{L}_n^{\text{data}} = \|\boldsymbol{z}_{n+1}^{\text{GT}} - \boldsymbol{z}_{n+1}^{\text{net}}\|_2^2. \tag{9}$$

- **Fulfillment of the degeneracy conditions**: The cost function will also account for the degeneracy conditions in order to ensure the thermodynamic consistency of the solution, implemented as the sum of the squared elements of the degeneracy vectors for each time step,

$$\mathcal{L}_n^{\text{degen}} = \|\mathsf{L} \cdot \boldsymbol{B}_n^{\text{net}}\boldsymbol{z}_n^{\text{net}}\|_2^2 + \|\mathsf{M} \cdot \boldsymbol{A}_n^{\text{net}}\boldsymbol{z}_n^{\text{net}}\|_2^2. \tag{10}$$

  This term acts as a regularization of the loss function and, at the same time, is the responsible of ensuring thermodynamic consistency. So to speak, it is the cornerstone of our method.

- **Regularization**: In order to avoid overfitting, an extra L2 regularization term $\mathcal{L}^{\text{reg}}$ is added to the loss function, defined as the sum over the squared weight parameters of the network.

$$\mathcal{L}^{\text{reg}} = \sum_{l}^{L}\sum_{i}^{n^{[l]}}\sum_{j}^{n^{[l+1]}} (w_{i,j}^{[l]})^2. \tag{11}$$

The total cost function is computed as the sum squared error (SSE) of the data loss and degeneracy residual, in addition to the regularization term, at the end of the simulation time T for each train case. The regularization loss is highly

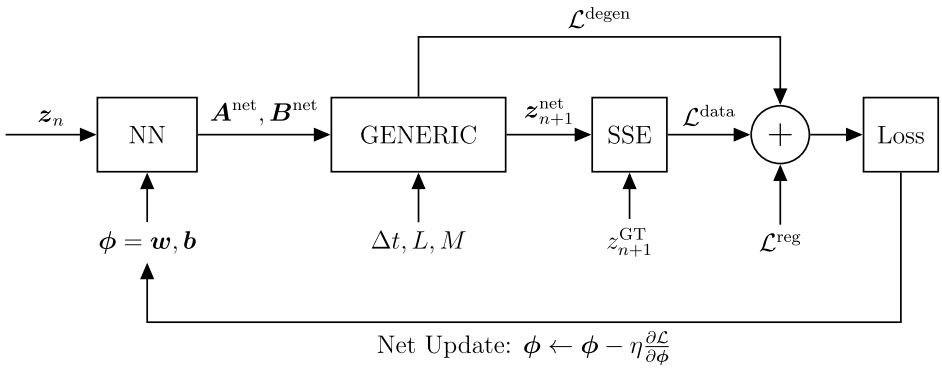

Fig. 2. Sketch of a structure-preserving neural network training algorithm.

dependent on the size of the network layers and has different scaling with respect to the other terms, so it is compensated with the regularization hyperparameter (weight decay) λ_r. An additional weight λ_d is added to the data loss term, which accounts for the relative scaling error with respect to the degeneracy conditions.

$$\mathcal{L} = \sum_{n=0}^{N_T} (\lambda_d \mathcal{L}_n^{\text{data}} + \mathcal{L}_n^{\text{degen}}) + \lambda_r \mathcal{L}^{\text{reg}}. \tag{12}$$

The usual backpropagation algorithm [37] is then used to calculate the gradient of the loss function for each net parameter (weight and bias vectors), which are updated with the gradient descent technique [43]. The process is then repeated for a maximum number of epochs n_{epoch}. The resulting training algorithm is sketched in Fig. 2.

The proposed methodology is tested with two different databases of nonlinear physical systems, split in a partition of train cases ($N_{\text{train}} = 80\%$ of the database) and test cases ($N_{\text{test}} = 20\%$ of the database). The net performance is evaluated with the mean squared error (MSE) of the state variables prediction, associated with the data loss term, Eq. (9), over all the time snapshots,

$$\text{MSE}^{\text{data}}(\boldsymbol{z}_i) = \frac{1}{N_T} \sum_{n=0}^{N_T} \left(\boldsymbol{z}_{i,n}^{\text{GT}} - \boldsymbol{z}_{i,n}^{\text{net}} \right)^2. \tag{13}$$

The same procedure is applied to the degeneracy constraint, associated with the degeneracy loss term, Eq. (10), over all the time snapshots,

$$\text{MSE}^{\text{degen}}(\boldsymbol{z}_i) = \frac{1}{N_T} \sum_{n=0}^{N_T} \left(\mathsf{L} \cdot \boldsymbol{B}_{i,n}^{\text{net}} \boldsymbol{z}_{i,n}^{\text{net}} + \mathsf{M} \cdot \boldsymbol{A}_{i,n}^{\text{net}} \boldsymbol{z}_{i,n}^{\text{net}} \right). \tag{14}$$

As a general error magnitude of the algorithm, the average MSE of both the train ($N = N_{\text{train}}$) and test trajectories ($N = N_{\text{test}}$) is also reported for both the data ($m = \text{data}$) and degeneracy ($m = \text{degen}$) constraints,

$$\overline{\text{MSE}}^m(\boldsymbol{z}) = \frac{1}{N} \sum_{i=1}^{N} \text{MSE}^m(\boldsymbol{z}_i). \tag{15}$$

Algorithm 1 and Algorithm 2 show a pseudocode of our proposed algorithm to both the training and test processes. The proposed method is fully implemented in PyTorch [38] and trained in an Intel Core i7-8665U CPU.

4. Validation examples: double thermo-elastic pendulum

4.1. Description

The first example is a double thermo-elastic pendulum (Fig. 3) consisting of two masses m_1 and m_2 connected by two springs of variable lengths λ_1 and λ_2 and natural lengths at rest λ_1^0 and λ_2^0.

The set of variables describing the double pendulum are here chosen to be

$$\mathcal{S} = \{\boldsymbol{z} = (\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{p}_1, \boldsymbol{p}_2, s_1, s_2) \in (\mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R} \times \mathbb{R}), \quad \boldsymbol{q}_1 \neq \boldsymbol{0},\ \boldsymbol{q}_1 \neq \boldsymbol{q}_2\}, \tag{16}$$

where $\boldsymbol{q}_i$, $\boldsymbol{p}_i$ and s_i are the position, linear momentum and entropy of each mass $i = 1, 2$.

Algorithm 1 Pseudocode for the train algorithm.

Load train database: $\boldsymbol{z}^{\mathrm{GT}}$ (train partition), Δt, L, M;
Define network architecture: N_{in}, $N_{\mathrm{out}} = 2N_{\mathrm{in}}^2$, N_h, σ_j;
Define hyperparameters: η, λ_d, λ_r;
Initialize $w_{i,j}$, b_j;
for *epoch* $\leftarrow 1, n_{\mathrm{epoch}}$ **do**
  for *train_case* $\leftarrow 1, N_{\mathrm{train}}$ **do**
    Initialize state vector: $\boldsymbol{z}_0^{\mathrm{net}} \leftarrow \boldsymbol{z}_0^{\mathrm{GT}}$;
    Initialize losses: $\mathcal{L}^{\mathrm{data}}, \mathcal{L}^{\mathrm{degen}} = 0$;
    for *snapshot* $\leftarrow 1, N_T$ **do**
      Forward propagation: $[\boldsymbol{A}_n^{\mathrm{net}}, \boldsymbol{B}_n^{\mathrm{net}}] \leftarrow \mathrm{Net}(\boldsymbol{z}_n^{\mathrm{GT}})$; ▷ Eq. (8)
      Time integration:
$\boldsymbol{z}_{n+1}^{\mathrm{net}} \leftarrow \boldsymbol{z}_n^{\mathrm{net}} + \Delta t\,(\mathsf{L} \cdot \boldsymbol{A}_n^{\mathrm{net}} \boldsymbol{z}_n^{\mathrm{net}} + \mathsf{M} \cdot \boldsymbol{B}_n^{\mathrm{net}} \boldsymbol{z}_n^{\mathrm{net}})$; ▷ Eq. (4)
      Update data loss: $\mathcal{L}^{\mathrm{data}} \leftarrow \mathcal{L}^{\mathrm{data}} + \mathcal{L}_n^{\mathrm{data}}$; ▷ Eq. (9)
      Update degeneracy loss: $\mathcal{L}^{\mathrm{degen}} \leftarrow \mathcal{L}^{\mathrm{degen}} + \mathcal{L}_n^{\mathrm{degen}}$; ▷ Eq. (10)
    end for
    SSE loss function: $L \leftarrow \lambda_d \mathcal{L}^{\mathrm{data}} + \mathcal{L}^{\mathrm{degen}} + \lambda_r \mathcal{L}^{\mathrm{reg}}$ ▷ Eq. (11), Eq. (12)
    Backward propagation;
    Optimizer step;
  end for
  Learning rate scheduler;
end for

Algorithm 2 Pseudocode for the test algorithm.

Load test database: $\boldsymbol{z}^{\mathrm{GT}}$ (test partition), Δt, L, M;
Load network parameters;
for *test_case* $\leftarrow 1, N_{test}$ **do**
  Initialize state vector: $\boldsymbol{z}_0^{\mathrm{net}} \leftarrow \boldsymbol{z}_0^{\mathrm{GT}}$;
  for *snapshot* $\leftarrow 1, N_T$ **do**
    Forward propagation: $[\boldsymbol{A}_n^{\mathrm{net}}, \boldsymbol{B}_n^{\mathrm{net}}] \leftarrow \mathrm{Net}(\boldsymbol{z}_n^{\mathrm{net}})$; ▷ Eq. (8)
    Time step integration:
$\boldsymbol{z}_{n+1}^{\mathrm{net}} \leftarrow \boldsymbol{z}_n^{\mathrm{net}} + \Delta t\,(\mathsf{L} \cdot \boldsymbol{A}_n^{\mathrm{net}} \boldsymbol{z}_n^{\mathrm{net}} + \mathsf{M} \cdot \boldsymbol{B}_n^{\mathrm{net}} \boldsymbol{z}_n^{\mathrm{net}})$; ▷ Eq. (4)
    Update state vector: $\boldsymbol{z}_n^{\mathrm{net}} \leftarrow \boldsymbol{z}_{n+1}^{\mathrm{net}}$;
    Update snapshot: $n \leftarrow n + 1$;
  end for
  Compute $\mathrm{MSE}^{\mathrm{data}}$, $\mathrm{MSE}^{\mathrm{degen}}$; ▷ Eq. (13), Eq. (14)
end for
Compute $\overline{\mathrm{MSE}}^{\mathrm{data}}$, $\overline{\mathrm{MSE}}^{\mathrm{degen}}$; ▷ Eq. (15)

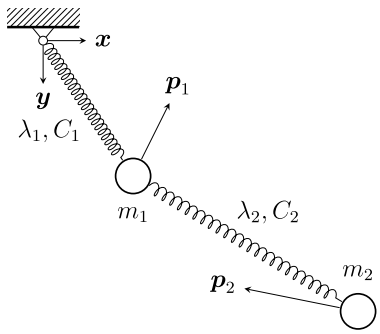

Fig. 3. Double thermo-elastic pendulum.

The lengths of the springs λ_1 and λ_2 are defined solely in terms of the positions as

$$\lambda_1 = \sqrt{\boldsymbol{q}_1 \cdot \boldsymbol{q}_1}, \quad \lambda_2 = \sqrt{(\boldsymbol{q}_2 - \boldsymbol{q}_1) \cdot (\boldsymbol{q}_2 - \boldsymbol{q}_1)}.$$

The total energy of the system can be expressed as the sum of the kinetic energy of the two masses K_i and the internal energy of the springs e_i for $i = 1, 2$,

$$E = E(\boldsymbol{z}) = \sum_i K_i(\boldsymbol{z}) + \sum_i e_i(\lambda_i, s_i),$$
$$K_i = \frac{1}{2m_i} |\boldsymbol{p}_i|^2. \tag{17}$$

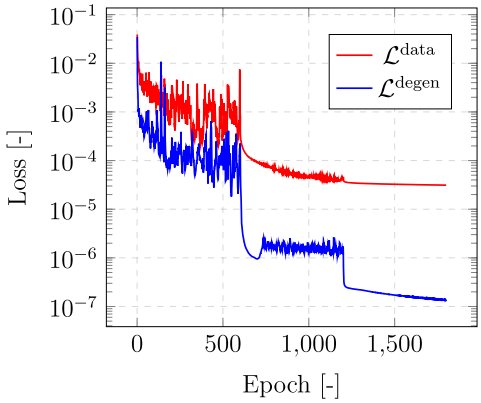

Fig. 4. Loss evolution of data and degeneracy constraints for each epoch of the structure-preserving neural network training process of the double pendulum example.

The total entropy of the double pendulum is the sum of the entropies of the two masses s_i,

$$S = S(\boldsymbol{z}) = s_1 + s_2. \tag{18}$$

This model includes thermal effects in the stretching of the springs due to the Gough-Joule effect. The absolute temperatures T_i at each spring are obtained through Eq. (19). These temperature changes induce a heat flux between both springs, being proportional to the temperature difference and a conductivity constant $\kappa > 0$,

$$T_i = \frac{\partial e_i}{\partial s_i}. \tag{19}$$

In this case, there is a clear contribution of both conservative Hamiltonian mechanics (mass movement) and non-Hamiltonian dissipative effects (heat flux), resulting in a non-zero Poisson matrix ($\mathsf{M} \neq \mathbf{0}$). Thus, the GENERIC matrices associated with this physical system are known to be [11]

$$\mathsf{L} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{-1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{-1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 \end{bmatrix}, \quad \mathsf{M} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 & -1/2 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -1/2 & 1 \end{bmatrix}. \tag{20}$$

4.2. Database and hyperparameters

The training database is generated with a thermodynamically consistent time-stepping algorithm [41] in MATLAB. The masses of the double pendulum are set to $m_1 = 1$ kg and $m_2 = 2$ kg, joint with springs of a natural length of $\lambda_1^0 = 2$ m and $\lambda_2^0 = 1$ m and thermal constant of $C_1 = 0.02$ J and $C_2 = 0.2$ J and conductivity constant of $\kappa = 0.5$. The simulation time of the movement is $T = 60$ s in time increments of $\Delta t = 0.3$ s ($N_T = 200$ snapshots).

The database consists of the state vector, Eq. (16), of 50 different trajectories with random initial conditions of position $\boldsymbol{q}_i$ and linear momentum $\boldsymbol{p}_i$ of both masses m_i ($i = 1, 2$) around a mean position and linear momentum of $\boldsymbol{q}_1 = [4.5,\ 4.5]^\top$ m, $\boldsymbol{p}_1 = [2,\ 4.5]^\top$ kg·m/s, and $\boldsymbol{q}_2 = [-0.5,\ 1.5]^\top$ m, $\boldsymbol{p}_2 = [1.4,\ -0.2]^\top$ kg·m/s respectively. Although the initial conditions of the simulations are similar, it results in a wide variety of the mass trajectories due to the chaotic behavior of the system. This database is split randomly in 40 train trajectories and 10 test trajectories. Thus, there is a total of 80.000 training snapshots and 20.000 test snapshots.

The net input and output size is $N_{\text{in}} = 10$ and $N_{\text{out}} = 2N_{\text{in}}^2 = 200$. The state vector is normalized based on the training set statistical mean and standard deviation. The number of hidden layers is $N_h = 5$ with ReLU activation functions and linear in the last layer. It is initialized according to the Kaiming method [16] with normal distribution and the optimizer used is Adam [24] with a weight decay of $\lambda_r = 10^{-5}$ and data loss weight of $\lambda_d = 10^2$. A multistep learning rate scheduler is used, starting in $\eta = 10^{-3}$ and decaying by a factor of $\gamma = 0.1$ in epochs 600 and 1200. The training process ends when a fixed number of epochs $n_{epoch} = 1800$ is reached.

The time evolution of the data $\mathcal{L}^{\text{data}}$ and degeneracy $\mathcal{L}^{\text{degen}}$ loss terms for each training epoch are shown in Fig. 4.

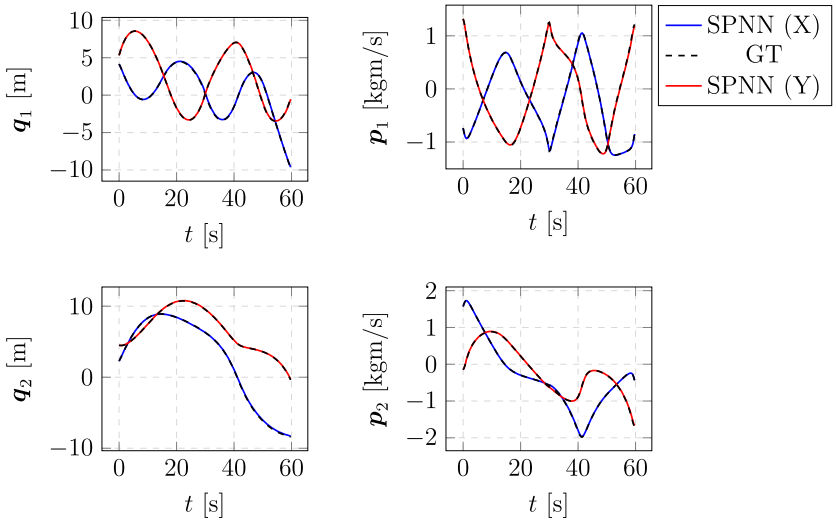

Fig. 5. Time evolution of the state variables in a test trajectory of a double thermo-elastic pendulum using a time-stepping solver (Ground Truth, GT) and the proposed GENERIC integration scheme (SPNN). Since every variable has a vectorial character, both components are depicted and labeled as X and Y, respectively.

Table 1
Mean squared error of the data loss ($\overline{\text{MSE}}^{\text{data}}$) and degeneracy loss ($\overline{\text{MSE}}^{\text{degen}}$) for all the state variables of the double pendulum.

State Variables		$\overline{\text{MSE}}^{\text{data}}$		$\overline{\text{MSE}}^{\text{degen}}$	
		Train	Test	Train	Test
$\boldsymbol{q}_1$ [m]	X	$1.95 \cdot 10^{-2}$	$3.87 \cdot 10^{-2}$	$3.56 \cdot 10^{-8}$	$4.43 \cdot 10^{-8}$
	Y	$2.72 \cdot 10^{-2}$	$8.21 \cdot 10^{-2}$	$4.74 \cdot 10^{-8}$	$5.77 \cdot 10^{-8}$
$\boldsymbol{q}_2$ [m]	X	$2.04 \cdot 10^{-2}$	$3.65 \cdot 10^{-2}$	$9.28 \cdot 10^{-8}$	$8.55 \cdot 10^{-8}$
	Y	$2.72 \cdot 10^{-2}$	$3.73 \cdot 10^{-2}$	$3.55 \cdot 10^{-8}$	$5.01 \cdot 10^{-8}$
$\boldsymbol{p}_1$ [kg·m/s]	X	$6.43 \cdot 10^{-4}$	$1.33 \cdot 10^{-4}$	$4.00 \cdot 10^{-8}$	$7.08 \cdot 10^{-8}$
	Y	$1.06 \cdot 10^{-3}$	$4.06 \cdot 10^{-3}$	$1.21 \cdot 10^{-7}$	$1.40 \cdot 10^{-7}$
$\boldsymbol{p}_2$ [kg·m/s]	X	$4.88 \cdot 10^{-4}$	$9.84 \cdot 10^{-4}$	$6.00 \cdot 10^{-8}$	$4.58 \cdot 10^{-8}$
	Y	$8.47 \cdot 10^{-4}$	$1.79 \cdot 10^{-4}$	$9.76 \cdot 10^{-8}$	$1.20 \cdot 10^{-7}$
s_1 [J/K]		$1.21 \cdot 10^{-5}$	$3.51 \cdot 10^{-5}$	$1.31 \cdot 10^{-7}$	$2.06 \cdot 10^{-7}$
s_2 [J/K]		$1.22 \cdot 10^{-5}$	$3.18 \cdot 10^{-5}$	$2.40 \cdot 10^{-7}$	$2.95 \cdot 10^{-7}$

Table 2
Mean squared error of the energy ($\overline{\text{MSE}}\,(E)$) and entropy ($\overline{\text{MSE}}\,(s)$) of the double pendulum.

Variable	Train	Test
E [J]	$7.99 \cdot 10^{-3}$	$8.86 \cdot 10^{-3}$
S [J/K]	$6.52 \cdot 10^{-8}$	$6.33 \cdot 10^{-8}$

4.3. Results

Fig. 5 shows the time evolution of the state variables (position, momentum and entropy) of each mass given by the solver and the neural net.

Table 1 shows the mean squared error of the data and degeneration loss terms for all the state variables of the double pendulum. The results are computed separately as the mean over all the train and test trajectories using Eq. (15).

Fig. 6 and Fig. 7 show the time evolution of the internal and kinetic energy (Eq. (17)) and the entropy (Eq. (18)) respectively for the two pendulum masses ($i = 1, 2$). The total energy is conserved and the total entropy satisfies the entropy inequality, fulfilling the first and second laws of thermodynamics respectively. The mean error for both train and test trajectories is reported in Table 2.

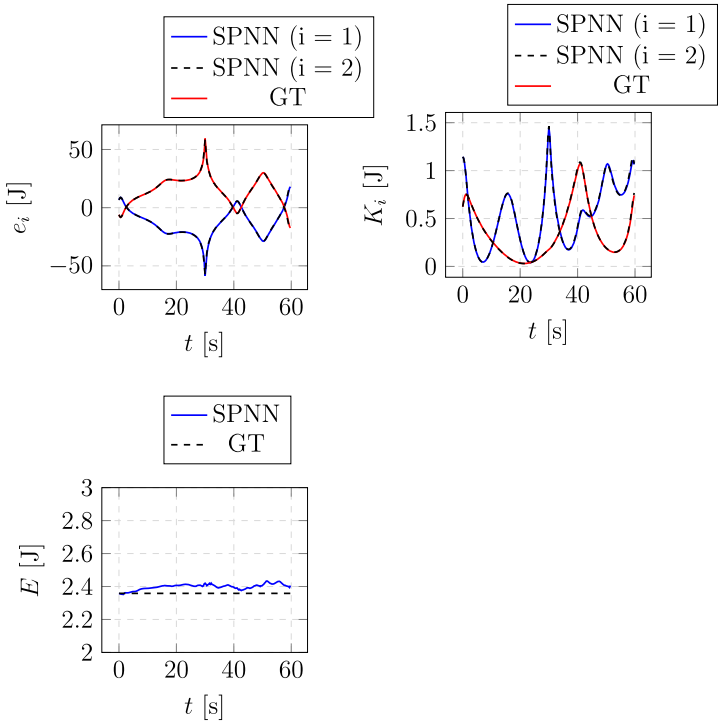

Fig. 6. Time evolution of the energy in a test trajectory of a double thermo-elastic pendulum using a time-stepping solver (Ground Truth, GT) and the proposed GENERIC integration scheme (Net).

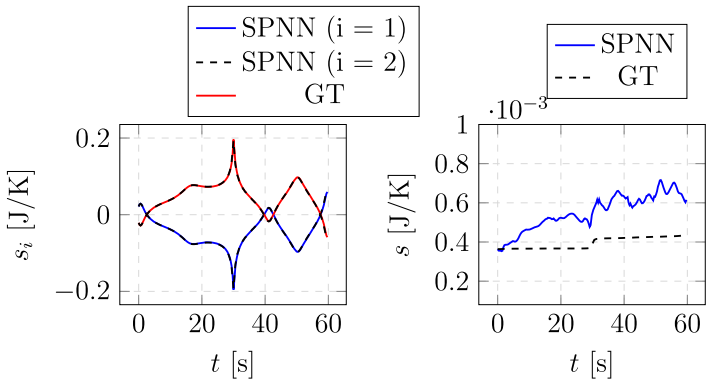

Fig. 7. Time evolution of the entropy in a test trajectory of a double thermo-elastic pendulum using a time-stepping solver (Ground Truth, GT) and the proposed GENERIC integration scheme (SPNN).

5. Couette flow of an Oldroyd-B fluid

5.1. Description

The second example is a shear (Couette) flow of an Oldroyd-B fluid model. This is a constitutive model for viscoelastic fluids, consisting of linear elastic dumbbells (representing polymer chains) immersed in a solvent.

The Oldroyd-B model arises in the modeling of flows of diluted polymeric solutions. This model can be obtained both from a purely macroscopic point of view as well as from a microscopic one, by modeling polymer chains as linear dumbbells diluted in a Newtonian substrate. Alternatively, it can also be obtained by considering the deviatoric part $\boldsymbol{T}$ of the stress tensor $\boldsymbol{\sigma}$ (the so-called extra-stress tensor), to be of the form

$$\boldsymbol{T} + \lambda_1 \overset{\nabla}{\boldsymbol{T}} = \eta_0 \left(\dot{\boldsymbol{\gamma}} + \lambda_2 \overset{\nabla}{\dot{\boldsymbol{\gamma}}} \right), \tag{21}$$

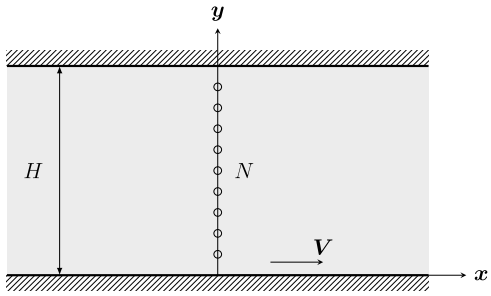

Fig. 8. Couette flow in an Oldroyd-B fluid.

where the triangle denotes the non-linear Oldroyd's upper-convected derivative [39]. Coefficients η_0, λ_1 and λ_2 are model parameters. It is standard to denote the strain rate tensor by $\dot{\boldsymbol{\gamma}} = (\nabla^s \boldsymbol{v}) = \boldsymbol{D}$.

Finally, the stress in the solvent (denoted by a subscript s) and polymer (denoted by a subscript p) are given by

$$\boldsymbol{T} = \eta_s \dot{\boldsymbol{\gamma}} + \boldsymbol{\tau},$$

so that

$$\boldsymbol{\tau} + \lambda_1 \overset{\triangledown}{\boldsymbol{\tau}} = \eta_p \dot{\boldsymbol{\gamma}},$$

which is the constitutive equation for the elastic stress.

Pseudo-experimental data are obtained by the CONNFFESSIT technique [27], based on the Fokker-Plank equation [28]. This equation is solved by converting it in its corresponding Itô stochastic differential equation,

$$\begin{aligned} dr_x &= \left(\frac{\partial \boldsymbol{v}}{\partial y} r_y - \frac{1}{2\mathrm{We}} r_x\right) dt + \frac{1}{\sqrt{\mathrm{We}}} dV_t, \\ dr_y &= -\frac{1}{2\mathrm{We}} r_y dt + \frac{1}{\sqrt{\mathrm{We}}} dW_t, \end{aligned} \tag{22}$$

where $\boldsymbol{v}$ is the flow velocity, $\boldsymbol{r} = [r_x,\ r_y]^\top$, $r_x = r_x(y,t)$ the position vector and assuming a Couette flow so that $r_y = r_y(t)$ depends only on time, we stand for the Weissenberg number and V_t, W_t are two independent one-dimensional Brownian motions. This equation is solved via Monte Carlo techniques, by replacing the mathematical expectation by the empirical mean.

The model relies on the microscopic description of the state of the dumbbells. Thus, it is particularly useful to base the microscopic description on the evolution of the conformation tensor $\boldsymbol{c} = \langle \boldsymbol{rr} \rangle$, this is, the second moment of the dumbbell end-to-end distance distribution function. This tensor is in general not experimentally measurable and plays the role of an internal variable. The expected xy stress component tensor will be given by

$$\tau = \frac{\epsilon}{\mathrm{We}} \frac{1}{K} \sum_{k=1}^{K} r_x r_y,$$

where K is the number of simulated dumbbells and $\epsilon = \frac{\nu_p}{\nu_p}$ is the ratio of the polymer to solvent viscosities.

The state variables selected for this problem are the position of the fluid on each node of the mesh, see Fig. 8, its velocity $\boldsymbol{v}$ in the x direction, internal energy e and the conformation tensor shear component τ,

$$\mathcal{S} = \{\boldsymbol{z} = (\boldsymbol{q}, \boldsymbol{v}, e, \tau) \in (\mathbb{R}^2 \times \mathbb{R} \times \mathbb{R} \times \mathbb{R})\}. \tag{23}$$

The GENERIC matrices associated with each node of this physical system are the following

$$\mathsf{L} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathsf{M} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{24}$$

In order to simulate a measurement of real captured data, Gaussian noise is added to the state vector, computed as a random variable following a normal distribution with zero mean and standard deviation proportional to the standard deviation of the database $\sigma_{\boldsymbol{z}}$ and noise level ν,

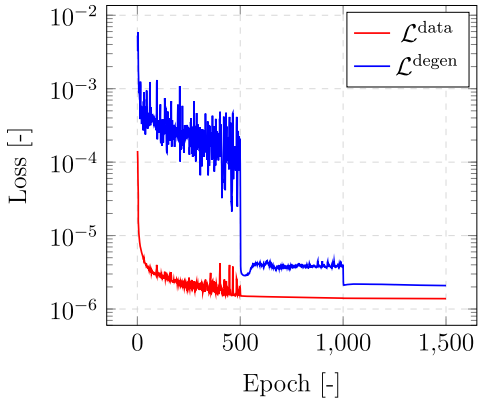

Fig. 9. Loss evolution of data and degeneracy constraints for each epoch of the neural network training process of the Couette flow example.

$$\boldsymbol{z}^{\text{GT}}_{noise} = \boldsymbol{z}^{\text{GT}} + \nu \cdot \sigma_{\boldsymbol{z}} \cdot \mathcal{N}(0, 1) \tag{25}$$

The results of both the noise-free and the noisy database are compared with two different network architectures:

- Unconstrained network: This architecture is the same as the proposed network but removing the degeneracy conditions of the energy and entropy, Eq. (10), in the loss function. These conditions ensure the thermodynamic consistency of the resulting integrator, so not including them affects negatively in the accuracy of the results, as will be seen.
- Black-Box network: In this case, no GENERIC architecture is imposed, acting as a black-box integrator trained to directly predict the state vector time evolution $\boldsymbol{z}_{t+1}$ from the previous time step $\boldsymbol{z}_t$. This naive approach is shown to be inappropriate, as no physical restrictions are given to the model.

5.2. Database and hyperparameters

The training database for this Oldroyd-B model is generated in MATLAB with a multiscale approach [28] in the dimensionless form. The fluid is discretized in the vertical direction with $N = 100$ elements (101 nodes) in a total height of $H = 1$. A total of 10,000 dumbells were considered at each nodal location in the model. The lid velocity is set to $V = 1$, the viscoelastic Weissenberg number $\text{We} = 1$ and Reynolds number of $\text{Re} = 0.1$. The simulation time of the movement is $T = 1$ in time increments of $\Delta t = 0.0067$ ($N_T = 150$ snapshots).

The database consisted of the state vector (Eq. (23)) of the 100 nodes trajectories (excluding the node at $h = H$, for which a no-slip condition $v = 0$ has been imposed). This database is split in 80 train trajectories and 20 test trajectories.

The net input and output size is $N_{\text{in}} = 5$ and $N_{\text{out}} = 2N^2_{\text{in}} = 50$. The number of hidden layers is $N_h = 5$ with ReLU activation functions and linear in the last layer. It is initialized according to the Kaiming method [16], with normal distribution and the optimizer used is Adam [24], with a weight decay of $\lambda_r = 10^{-5}$ and data loss weight of $\lambda_d = 10^3$. A multistep learning rate scheduler is used, starting in $\eta = 10^{-3}$ and decaying by a factor of $\gamma = 0.1$ in epochs 500 and 1000. The training process ends when a fixed number of epochs $n_{epoch} = 1500$ is reached. The same parameters are considered also for the noisy database network ($\nu = 1\%$) and the unconstrained network.

The black-box network training parameters are analogous to the structure-preserving network, except for the output size $N_{\text{out}} = N_{\text{in}} = 5$. Several network architectures were tested, and the lowest error is achieved with $N_h = 5$ hidden layers and 25 neurons each layer.

The time evolution of the data $\mathcal{L}^{\text{data}}$ and degeneracy $\mathcal{L}^{\text{degen}}$ loss terms for each training epoch are shown in Fig. 9.

5.3. Results

Fig. 10 shows the time evolution of the state variables (position q, velocity v, internal energy e and conformation tensor shear component τ) given by the solver and the neural net. There is a good agreement between both plots. Moreover, the proposed scheme is able to predict the time evolution of the flow for several snapshots beyond the training simulation time $T = 1$, as shown in the same figure.

Table 3 show the mean squared error of the data and degeneration loss terms for all the state variables of the Couette flow of an Oldroyd-B fluid. The results are computed separately as the mean over all the train and test trajectories using Eq. (15).

Fig. 11 shows a box plot of the data error (MSE^{data}) for the train and test sets in the four studied architectures. The results of the structure-preserving neural network outperform the other two approaches even with noisy training data. The error of the unconstrained neural network is greater than one order of magnitude than our approach, proving the importance of the degeneracy conditions in the GENERIC formulation. Last, the naive black-box approach shows the worst performance of the four networks, as no physical restriction is considered.

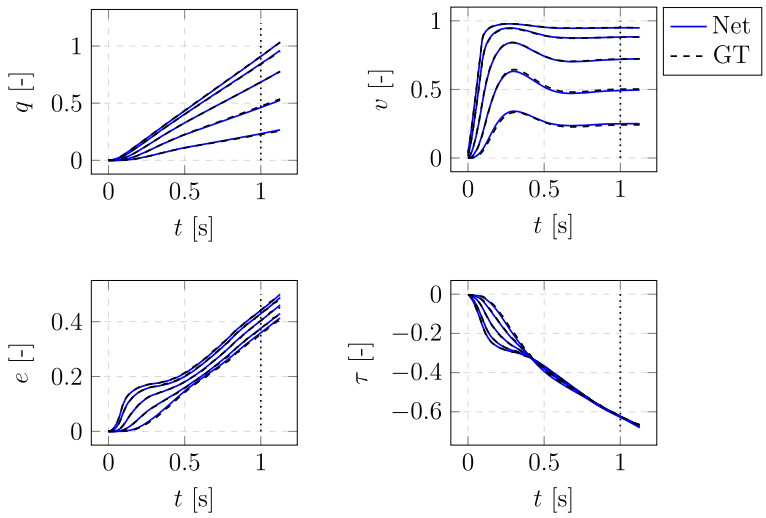

Fig. 10. Time evolution of the state variables in five test nodes of a Couette flow using a solver (Ground Truth, GT) and the proposed GENERIC integration scheme (Net). The dotted vertical line represents the simulation time $T = 1$ of the training dataset.

Table 3
Mean squared error of the data loss ($\overline{\text{MSE}}^{\text{data}}$) and degeneracy loss ($\overline{\text{MSE}}^{\text{degen}}$) for all the state variables of the Couette flow.

State Variables		$\overline{\text{MSE}}^{\text{data}}$		$\overline{\text{MSE}}^{\text{degen}}$	
		Train	Test	Train	Test
q [-]	X	$5.40 \cdot 10^{-6}$	$6.29 \cdot 10^{-6}$	$1.72 \cdot 10^{-7}$	$1.96 \cdot 10^{-7}$
	Y	0.00	0.00	0.00	0.00
v [-]		$3.23 \cdot 10^{-5}$	$4.75 \cdot 10^{-5}$	$1.19 \cdot 10^{-6}$	$1.48 \cdot 10^{-6}$
e [-]		$7.85 \cdot 10^{-6}$	$6.60 \cdot 10^{-6}$	$7.06 \cdot 10^{-7}$	$9.11 \cdot 10^{-7}$
τ [-]		$2.36 \cdot 10^{-5}$	$1.26 \cdot 10^{-5}$	$1.07 \cdot 10^{-6}$	$1.31 \cdot 10^{-6}$

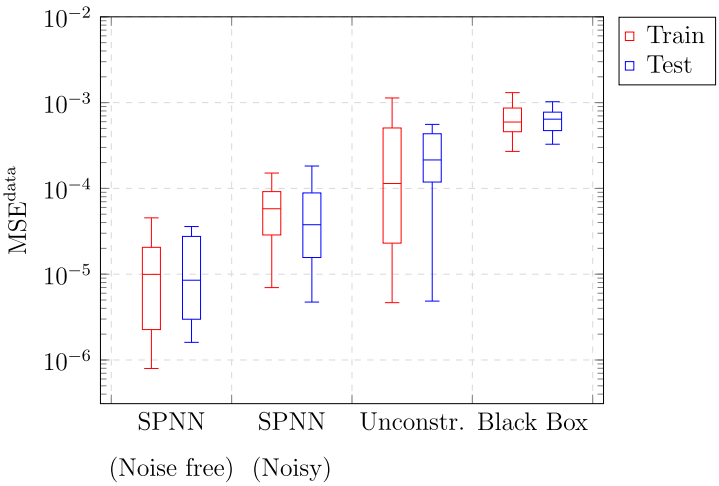

Fig. 11. Box plots for the data integration mean squared error (MSE^{data}) of the Couette flow in both train and test cases.

With respect to our previous work [11], that employed a piece-wise linear regression approach, these examples show similar levels of accuracy, but a much greater level of robustness. For instance, this same example was included in the mentioned reference. However, in that case, the problem had to be solved with the help of a reduced order model with only six degrees of freedom, due to the computational burden of the approach. In our former approach, the GENERIC structure was identified by piece-wise linear regression for each of the few global *modes* of the approximation. So to speak, in that case, we *learnt* the characteristics of the flow. Here, on the contrary, the net is able to find an approximation for any velocity value at the 101 nodes of the mesh—say, fluid particles—without any difficulty. In this case, we are *learning* the

Table 4
Computation training time of the proposed algorithm for the two reported examples in the noise free networks.

Example	Epoch Time	Total Time
Double Pendulum	2.44 s/epoch	73.18 min
Couette Flow	1.22 s/epoch	30.53 min

behavior of fluid particles. It will be interesting, however, to study to what extent the employ of variational autoencoders, as in Bertalan et al. [2], could help in solving more intricate models. Autoencoders help in determining the actual number of degrees of freedom needed to represent a given physical phenomenon.

6. Conclusions

In this work we have presented a new methodology to ensure thermodynamic consistency in the deep learning of physical phenomena. In contrast to existing methods, this methodology does not need to know in advance any information related to balance equations or the precise form of the PDE governing the phenomena at hand. The method is constructed on top of the right thermodynamic principles that ensure the fulfillment of the energy dissipation and entropy production. It is valid, therefore, for conservative as well as dissipative systems, thus overcoming previous approaches in the field.

When compared with our previous works in the field (see Gonzalez et al. [11]), the present methodology showed to be more robust, allowing us to find approximations for systems with orders of magnitude more degrees of freedom. This new approach is also less computationally demanding. For the double pendulum case, the snapshot optimization of the GENERIC matrices proposed in [11] has a measured performance of 10 min per trajectory, which add up to 400 minutes considering the 40 studied trajectories, whereas our new neural-network approach trains in only 73.18 minutes. The computational time of the other examples is shown in Table 4.

The reported results show good agreement between the network output and the synthetic ground truth solution, even with moderate noisy data. We have also shown the importance of including the degeneracy conditions of the GENERIC formulation to the neural network constraints, as it ensures the thermodynamical consistency of the integrator. The structure-preserving neural network outperforms other naive black-box approaches, since the physical constraints act as an inductive bias, facilitating the learning process. However, the error can be reduced using several techniques:

- **Database**: As a general method of increasing the precision of an Euler integration scheme, the time step Δt can be decreased so the total number of snapshots is increased. On the contrary, the database will be larger, slowing the training process. The same way, the database can be enriched with a wider variety of cases, improving the net predictive capabilities.
- **Integration Scheme**: A higher order Runge-Kutta integration scheme could be introduced in Eq. (4) in order to get higher solution accuracy [47]. However, it requires several forward passes through the neural net for each time step, incrementing the complexity of the integration scheme and the training process. Additionally, GENERIC-based integration schemes have showed very good performance even for first-order approaches. [41]
- **Net Architecture**: To increase the computational power of the net, more and larger hidden layers N_h can be added. However, this could lead to a more over-fitted solution which limit the prediction power and versatility of the net. It also increases the computational cost of both the training process and the testing of the net.
- **Training Hyperparameters**: The neural networks trained in this work could be optimized using several hyperparameter tuning methods such as random search, Bayesian optimization or gradient-based optimization to get a more efficient solution.

Several open questions remain as a future work. A more exhaustive analysis can be performed to evaluate the influence of noisy data to the integrator evolution, in order to add robustness to the method and even predict wider simulation times using incremental learning [5,30].

CRediT authorship contribution statement

- Quercus Hernandez: software, investigation, validation
- Alberto Badias: software, investigation, validation
- David González: data, investigation, validation
- Francisco Chinesta: methodology, validation
- Elias Cueto: conceptualization, methodology, investigation, writing

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This project has been partially funded by the ESI Group through the ESI Chair at ENSAM Arts et Metiers Institute of Technology, and through the project 2019-0060 “Simulated Reality” at the University of Zaragoza. The support of the Spanish Ministry of Economy and Competitiveness through grant number CICYT-DPI2017-85139-C2-1-R and by the Regional Government of Aragon, grant T24_20R, and the European Social Fund, are also gratefully acknowledged.

References

[1] Jacobo Ayensa-Jiménez, Mohamed H. Doweidar, Jose A. Sanz-Herrera, Manuel Doblaré, A new reliability-based data-driven approach for noisy experimental data with physical constraints, Comput. Methods Appl. Mech. Eng. 328 (2018) 752–774.
[2] Tom Bertalan, Felix Dietrich, Igor Mezić, Ioannis G. Kevrekidis, On learning Hamiltonian systems from data, Chaos: Interdiscip. J. Nonlinear Sci. 29 (12) (Dec 2019) 121107.
[3] Steven L. Brunton, Joshua L. Proctor, J. Nathan Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. (2016).
[4] Euntae Choi, Kyungmi Lee, Kiyoung Choi, Approximation by superpositions of a sigmoidal function, arXiv preprint, arXiv:1907.07872, 2019.
[5] George Cybenko, Autoencoder-based incremental class learning without retraining on old data, Math. Control Signals Syst. 2 (4) (1989) 303–314.
[6] Weinan E, A proposal on machine learning via dynamical systems, Commun. Math. Stat. 5 (1) (Mar 2017) 1–11.
[7] Chady Ghnatios, Iciar Alfaro, David González, Francisco Chinesta, Elias Cueto, Data-driven generic modeling of poroviscoelastic materials, Entropy 21 (12) (2019).
[8] Xavier Glorot, Antoine Bordes, Yoshua Bengio, Deep sparse rectifier neural networks, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS, 2011, pp. 315–323.
[9] D. González, F. Chinesta, E. Cueto, Consistent data-driven computational mechanics, AIP Conf. Proc. 1960 (1) (2018) 090005.
[10] David González, Francisco Chinesta, Elías Cueto, Learning corrections for hyperelastic models from data, Front. Mater. 6 (2019) 14.
[11] David González, Francisco Chinesta, Elías Cueto, Thermodynamically consistent data-driven computational mechanics, Contin. Mech. Thermodyn. 31 (1) (2019) 239–253.
[12] Alex Graves, Abdel-rahman Mohamed, Geoffrey Hinton, Speech recognition with deep recurrent neural networks, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2013, pp. 6645–6649.
[13] Miroslav Grmela, Generic guide to the multiscale dynamics and thermodynamics, Comput. Phys. Commun. 2 (3) (2018) 032001.
[14] Miroslav Grmela, Vaclav Klika, Michal Pavelka, Gradient and generic evolution towards reduced dynamics, 2019.
[15] Miroslav Grmela, Hans Christian Öttinger, Dynamics and thermodynamics of complex fluids. I. Development of a general formalism, Phys. Rev. E 56 (6) (1997) 6620.
[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Delving deep into rectifiers: surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE International Conference on Computer Vision, ICCV, 2015, pp. 1026–1034.
[17] Tony Hey, Stewart Tansley, Kristin Tolle, The Fourth Paradigm: Data-Intensive Scientific Discovery, Microsoft Research, October 2009.
[18] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, et al., Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups, IEEE Signal Process. Mag. 29 (6) (2012) 82–97.
[19] Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al., Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.
[20] Rubén Ibanez, Emmanuelle Abisset-Chavanne, Jose Vicente Aguado, David Gonzalez, Elias Cueto, Francisco Chinesta, A manifold learning approach to data-driven computational elasticity and inelasticity, Arch. Comput. Methods Eng. 25 (1) (2018) 47–57.
[21] Rubén Ibáñez, Emmanuelle Abisset-Chavanne, David González, Jean-Louis Duval, Elias Cueto, Francisco Chinesta, Hybrid constitutive modeling: data-driven learning of corrections to plasticity models, Int. J. Mater. Forming 12 (4) (2019) 717–725.
[22] Ruben Ibañez, Domenico Borzacchiello, Jose Vicente Aguado, Emmanuelle Abisset-Chavanne, Elías Cueto, Pierre Ladevèze, Francisco Chinesta, Data-driven non-linear elasticity: constitutive manifold construction and problem discretization, Comput. Mech. 60 (5) (2017) 813–826.
[23] Pengzhan Jin, Aiqing Zhu, George Em Karniadakis, Yifa Tang, Symplectic networks: intrinsic structure-preserving networks for identifying Hamiltonian systems, arXiv preprint, arXiv:2001.03750, 2020.
[24] Diederik P. Kingma, Jimmy Ba, Adam: a method for stochastic optimization, arXiv preprint, arXiv:1412.6980, 2014.
[25] Trenton Kirchdoerfer, Michael Ortiz, Data-driven computational mechanics, Comput. Methods Appl. Mech. Eng. 304 (2016) 81–101.
[26] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, Imagenet classification with deep convolutional neural networks, in: F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, vol. 25, Curran Associates, Inc., 2012, pp. 1097–1105.
[27] Manuel Laso, Hans Christian Öttinger, Calculation of viscoelastic flow using molecular models: the connffessit approach, J. Non-Newton. Fluid Mech. 47 (1993) 1–20.
[28] Claude Le Bris, Tony Lelievre, Multiscale modelling of complex fluids: a mathematical initiation, in: Multiscale Modeling and Simulation in Science, Springer, 2009, pp. 49–137.
[29] Jay Yoon Lee, Sanket Vaibhav Mehta, Michael Wick, Jean-Baptiste Tristan, Jaime Carbonell, Gradient-based inference for networks with output constraints, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2009.
[30] Zhizhong Li, Derek Hoiem, Learning without forgetting, in: IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.
[31] Pablo Márquez-Neila, Mathieu Salzmann, Pascal Fua, Imposing hard constraints on deep networks: promises and limitations, arXiv preprint, arXiv: 1706.02025, 2017.
[32] Jim Magiera, Deep Ray, Jan S. Hesthaven, Christian Rohde, Constraint-aware neural networks for Riemann problems, J. Comput. Phys. 409 (2020) 109345.
[33] W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, Bin Yu, Interpretable machine learning: definitions, methods, and applications, arXiv preprint, arXiv:1901.04592, 2019.
[34] Yatin Nandwani, Abhishek Pathak, Parag Singla, et al., A primal dual formulation for deep learning with constraints, in: Advances in Neural Information Processing Systems, 2019.

[35] Hans Christian Öttinger, Beyond Equilibrium Thermodynamics, John Wiley & Sons, 2005.
[36] Hans Christian Öttinger, Miroslav Grmela, Dynamics and thermodynamics of complex fluids. II. Illustrations of a general formalism, Phys. Rev. E 56 (6) (1997) 6633.
[37] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, Adam Lerer, Automatic differentiation in pytorch, in: Autodiff Workshop: The Future of Gradient-Based Machine Learning Software and Techniques, 2017.
[38] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, Soumith Chintala, Pytorch: an imperative style, high-performance deep learning library, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, vol. 32, Curran Associates, Inc., 2019, pp. 8026–8037.
[39] R.G. Owens, T.N. Phillips, Computational Rheology, Imperial College Press, 2002.
[40] Maziar Raissi, Paris Perdikaris, George E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[41] Ignacio Romero, Thermodynamically consistent time-stepping algorithms for non-linear thermomechanical systems, Int. J. Numer. Methods Eng. 79 (6) (2009) 706–732.
[42] Jonathan Romero, Jonathan P. Olson, Alan Aspuru-Guzik, Quantum autoencoders for efficient compression of quantum data, Quantum Sci. Technol. 2 (4) (2017) 045001.
[43] Sebastian Ruder, An overview of gradient descent optimization algorithms, arXiv preprint, arXiv:1609.04747, 2016.
[44] Jürgen Schmidhuber, Deep learning in neural networks: an overview, Neural Netw. 61 (2015) 85–117.
[45] Hoo-Chang Shin, Holger R. Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, Ronald M. Summers, Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning, IEEE Trans. Med. Imaging 35 (5) (2016) 1285–1298.
[46] Lucas Theis, Wenzhe Shi, Andrew Cunningham, Ferenc Huszár, Lossy image compression with compressive autoencoders, arXiv preprint, arXiv:1703.00395, 2017.
[47] Yi-Jen Wang, Chin-Teng Lin, Runge-Kutta neural network for identification of dynamical systems in high accuracy, IEEE Trans. Neural Netw. 9 (2) (1998) 294–307.
[48] Dongkun Zhang, Ling Guo, George Em Karniadakis, Learning in modal space: solving time-dependent stochastic pdes using physics-informed neural networks, SIAM J. Sci. Comput. 42 (2) (2020) A639–A665.