Latent-Energy-Based NNs: An interpretable Neural Network architecture for model-order reduction of nonlinear statics in solid mechanics

Louen Pottier, Anders Thorin, Francisco Chinesta

Latent-Energy-Based NNs: An interpretable Neural Network architecture for model-order reduction of nonlinear statics in solid mechanics

Louen POTTIER[1,2], Anders THORIN[1], Francisco CHINESTA

[1] Université Paris-Saclay, CEA List, Palaiseau, France

[2] PIMM Laboratory, Arts et Metiers Institude of Technology, CNRS, Cnma, HESAM Université, Paris, France

ABSTRACT Nonlinear mechanical systems can exhibit non-uniqueness of the displacement field in response to a force field, which is related to the non-convexity of strain energy. This work proposes a Neural Network-based surrogate model capable of capturing this phenomenon while introducing an energy in a latent space of small dimension, that preserves the topology of the strain energy; this feature is a novelty with respect to the state of the art. It is exemplified on two mechanical systems of simple geometry, but challenging strong nonlinearities. The proposed architecture offers an additional advantage over existing ones: it can be used to infer both displacements from forces, or forces from displacements, without being trained in both ways.

KEYWORDS Nonlinear mechanics; hyperelasticity; finite strain; surrogate models; neural networks; model reduction

Highlights

- *A neural network capable of reducing nonlinear statics models in solid mechanics in finite strain theory.*
- *Invertible neural network: can be used in forward or reverse mode (force prediction from displacement field, or vice versa).*
- *Capable of dealing with multiple equilibria, corresponding to non-convex strain energy.*
- *Equilibrium stability is preserved in the low-dimensional latent space.*

1. Introduction

Because they are soft, elastomers or biological tissues often undergo large strains and are typically modeled with hyperelastic constitutive laws. They are a source of nonlinearity and lead to mechanical simulations that can be numerically costly.

Model reduction techniques have been developed for decades in order to reduce the computational cost: Principal Component Analysis (PCA), Proper Orthogonal Decomposition (POD) or more recently the Proper Generalized Decomposition (PGD) [6]. In this paper, we refer to such techniques as *classical model reduction methods*.

On the one hand, classical model reduction methods emanate from centuries of knowledge and offer physical interpretability. On the other hand, a more recent research topic is very promising: the use of Neural Networks (NNs) to speed up simulations, through dimensionality reduction. However, NNs often suffer from a lack of interpretability and are seen as "black boxes" compared to classical model reduction methods.

This work proposes a NN architecture that provides both efficiency and interpretability, in the framework of hyperelasticity. Of particular interest is the capability of the NN to deal with multiple solutions, which is related to the energy non-convexity.

Enforcing physical properties in NN architectures One common approach to address the lack of interpretability is to inject physical knowledge into neural networks, through penalty methods. For instance, Physics Informed Neural Networks [34] consist in adding terms to the loss function to encourage the respect of governing equations. Another approach is to directly integrate physical properties into the neural network architecture. In a structural dynamics context, several methods have been proposed for this purpose. Lagrangian Neural Networks (LNN) [7, 25] are used to learn a Lagrangian operator which is then used to compute the time evolution of the structure's displacement by using the Euler–Lagrange equations. A variant of this idea has been proposed with Hamiltonian Neural Networks (HNN) [9, 12, 35]. Instead of a Lagrangian, a Hamiltonian is learned which allows to enforce the absence of dissipation. Another approach consists in learning jointly a dynamical system and a Lyapunov function with two neural networks and then using the Lyapunov function to restrict the learned dynamics which guarantees non-negative dissipation [19, 26]. Such approaches address dynamics, which is not of direct interest in the present work.

In this paper, we target applications for hyperelastic structural mechanical systems in statics. Our approach can be seen as a special case of Lagrangian NN with no dynamic terms in the Lagrangian and with some subtleties and contributions: learning physics in a low-dimensional latent space, using an energy structure for the static terms of the Lagrangian, and using a neural network inversion method to predict both displacement from force and force from displacement.

Large strain hyperelasticity framework The present work targets deformable bodies (beams, shells or volumes) undergoing material nonlinearity (ie. constitutive law is nonlinear) and geometric nonlinearity (large deformation), in a static framework. The deformable body is assumed to be discretized in space so that it has a finite number of degrees-of-freedom (dofs) $n \in \mathbb{N}^*$. Let $u \in \mathbb{R}^n$ denote the displacement field within a structure composed of a hyperelastic material, ie. whose constitutive equation derives from a polyconvex potential [3, 4]. Let $\mathcal{U}$ denote the potential energy of the structure that depends only on u. Let $f \in \mathbb{R}^n$ denote the external forces and $\mathcal{W}(u, f) := u^\top f$ the corresponding work. Provided f is expressed in a fixed frame (independent of u), the following governing equation holds:

$$\frac{\partial \mathcal{U}}{\partial u} = f.$$

The physical interpretation is that the internal elastic forces are balanced with the external forces f.

The corresponding problem can be seen as a direct or an inverse problem, whether the displacement or the force is chosen as a (known) input.

DtF: Displacement u → System → Force f?

FtD: Force f → System → Displacement u?

Displacement to force (DtF) Finding the external force given the corresponding displacement field is direct and easy, it only requires computing the gradient of the strain energy. This external force is obviously unique.

Force to displacement (FtD): On the contrary, finding a displacement field corresponding to a given external force is not direct and the solution is in general not unique.

If the strain energy $\mathcal{U}$ is convex, the solution of the FtD problem is unique for all given f. Otherwise, for some f, $u := \mathcal{U}(u) - \mathcal{W}(u, f)$ can have multiple equilibrium points that can be stable, unstable or saddle points. In practice, uniqueness holds locally and the solution can be retrieved from some initial guess u_0 using a root-finding algorithm (such as Newton–Raphson).

Neural Networks for hyperelasticity In the context of hyperelasticity, polyconvexity of the strain potential has already been structurally imposed in NN architectures [18, 21, 38]. Though the idea in these articles is well-suited to learn constitutive laws or for homogenization, they are not designed to capture the non-convexity induced by geometric nonlinearities [3]. Such approaches can therefore not be applied to learn the mechanical problem in the general framework of geometric nonlinearities.

Several approaches have been applied to train NN to learn the response of hyperelastic structures. Originally, Physics-Informed NNs (PINNs) correspond to learning $u(x, y) = \mathrm{NN}(x, y)$ for some prescribed f that is used in the learning metric, in such a way that the static equilibrium is satisfied for all (x, y). This approach, exemplified in [1], requires training the NN every time f changes. It is therefore not suited for the general model reduction approach where the $u \leftrightarrow f$ relationship is sought.

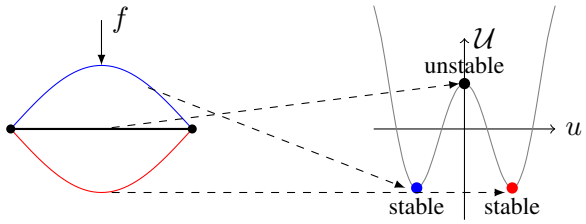

Figure 1: Example of a multistable static system: a prestressed clamped–clamped beam (left) and its strain energy (right). For $f = 0$, multiple FtD solutions coexist (two stable and one unstable): this illustrates the possible non-uniqueness of the FtD problem.

DtF predictions for hyperelastic systems have been carried out with U-nets [8] or FCNNs [36]. However, the use of model-order reduction is especially relevant for FtD problem, because it is more computationally expensive. Conversely, FtD predictions for hyperelastic systems have also been carried out with U-nets [27] or FCNNs [29], which are limited by construction to infer a unique solution, which restricts their applications to convex strain energy.

Addressing the non-uniqueness of the FtD problem In general, the FtD problem can have multiple solutions, see Fig. 1. A simple $u = \mathrm{NN}(f)$ structure can at best learn one FtD solution among the multiple ones, at worst learn a nonphysical weighting of multiple solutions (their average, for instance). For instance, in [1, 29], elongated structures with geometric non-linearities are involved leading to multiple solutions of the FtD problem. This non-uniqueness cannot be addressed with the approach therein. Instead, the following three approaches are able to address the non-uniqueness of the FtD problem:

1. retrieving the uniqueness in the neighborhood of a configuration u_0 by learning $u = \mathrm{NN}(f, u_0)$;
2. learning the DtF problem ie. $f = \mathrm{NN}(u)$ and using a root-finding algorithm initialized with u_0 to solve for u.
3. learning the set of FtD solutions ie. $\bigcup_i \{u_i\} = \mathrm{NN}(f)$.

Ideas one and two address the non-uniqueness of the problem by specifying the initial displacement u_0 from which we search a static solution u. The third idea provides the whole set of solutions.

Training the first family of architectures requires to have access to a (u, u_0, f) database. Depending on how the data was generated, we may only have access to a (u, f) database with no way to access to corresponding u_0. On the contrary, training the second family of architectures does not require u_0 because training is done in DtF mode for which the solution is unique, u_0 is only required in inference. Solutions of the FtD inference can then be obtained by reversing the trained NN, searching for some u such as $f = \mathrm{NN}(u)$ starting from u_0.

Some existing methods provide a closed-form reversibility of NNs [2, 11]. They rely on the bijectivity of the NN, which is restrictive in the present work because of the possibly multiple FtD solutions (cf. Fig. 1). In a more general framework, the NN can be reversed using a root-finding algorithm [16, 23] such as the Newton–Raphson procedure. The choice of the initialization u_0 of the root-finding algorithm will lead to different FtD solutions, which assert the non-uniqueness problem.

In [13, 41], an idea similar to the third family of architecture is proposed to solve inverse problems in the case of PDEs with several solutions. Their PINN was trained on a database of $(x, u(x))$ where $u(x)$ is the value in x of one or other solution of a parameterized PDE. They could retrieve the unknown parameter of the PDE (ie the equivalent of a single unknown value of f) and could train a PINN to predict every PDE solution simultaneously. Adapting one of such methods in the case of multiple values of f, using the PINN formalism or not, seems possible but non-trivial because the number of FtD solutions depends on f. Nevertheless, these examples show that it may also be possible to train the third family of architectures only having access to a (u, f) database even if the different solutions are not numbered. However, in inference, the third architecture does not easily allow to select the solution that would have been predicted by a physical solver starting from a given u_0. This can be restrictive in some applications.

Autoencoder structure Using an autoencoder structure to learn the underlying physics in a small dimensional so-called *latent space* has been proposed multiple times, see e.g. [14, 30]. Different encoder and decoder architectures can be used depending on the size of the mechanical fields to be learned and their discretization: fully-connected multilayer neural networks, convolutional NNs in the case of regular grid discretizations [15, 20, 27, 40], graph NNs for discretization on arbitrary meshes [22, 24, 28, 31], or even more classical reduction techniques such as POD [5, 37]. One originality of the present work is

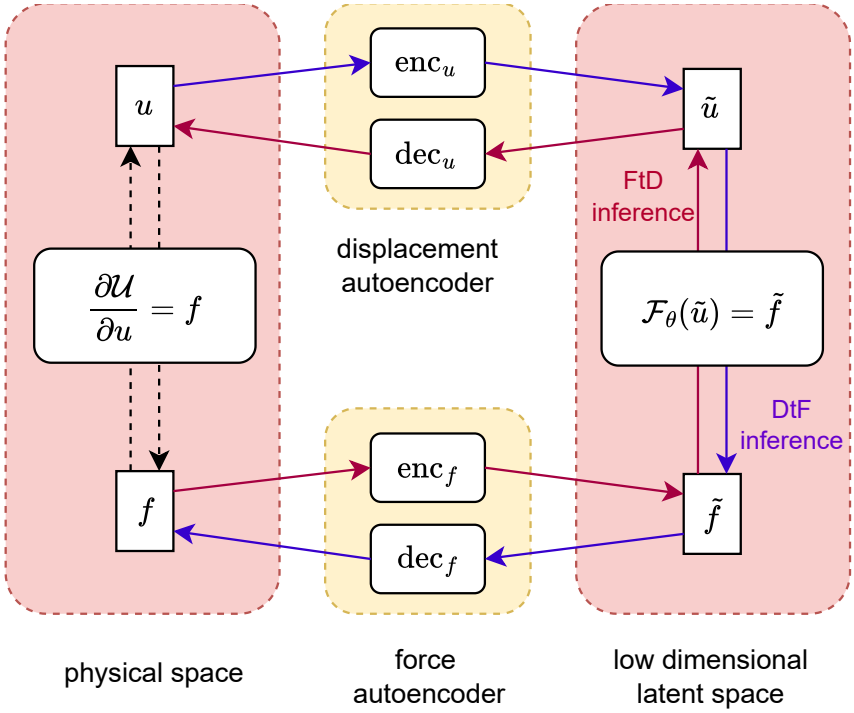

Figure 2: Architecture with double autoencoder, to learn highly nonlinear static equilibra. Of particular interest is the appropriate choice of the learnable relationship $\mathcal{F}_\theta(\tilde{u}) = \tilde{f}$ in the latent space.

that both known and unknown fields (displacement and force fields) are compressed using two different autoencoders.

The proposed architecture is exposed in more detail in Sec. 2. The test cases used to assess and compare the proposed NN are described in Sec. 3. Results are provided in the next section, Sec. 4.

2. Proposed architecture

2.1. Double autoencoder structure

Using the autoencoder-based neural network architecture depicted in Fig. 2 as a substitution model, we replace the search for the roots of $\partial_u \mathcal{U} - f$ with the search for the roots of a learnable latent model $\mathcal{F}_\theta(\tilde{u}) - \tilde{f}$ of much smaller dimensions; θ denotes the parameters of the model, for instance weights in a NN.

It can be used in both FtD and DtF modes, using a root-finding algorithm during FtD evaluation (see Sec. 1). The double autoencoder structure allows to reduce the size of the space in which this root-finding is performed, reducing the numerical cost and the risk of not converging.

DtF inference consists of the direct evaluation of $\mathrm{dec}_f(\mathcal{F}_\theta(\mathrm{enc}_u(u))$ whereas the FtD inference consists in Algo. 1.

Algorithm 1 FtD inference using Newton's method

Input: f, u_0
$\tilde{f} = \mathrm{enc}_f(f)$
$\tilde{u} = \mathrm{enc}_u(u_0)$
while $\|\mathcal{F}_\theta(\tilde{u}) - \tilde{f}\| > \varepsilon$ **do**
$\quad \tilde{u} \leftarrow \tilde{u} - \left(\frac{\partial \mathcal{F}_\theta(\tilde{u})}{\partial \tilde{u}}\right)^{-1} (\mathcal{F}_\theta(\tilde{u}) - \tilde{f})$
end while
$u = \mathrm{dec}_u(\tilde{u})$
Output : u

The choice of a suitable encoder and decoder architecture is problem-dependent and is a research topic in itself. Instead, this work focuses in transcribing the physical governing equations into a latent space. For that purpose, we have chosen encoders and decoders as simple fully-connected NNs for all the test cases and the architectures of the present contribution with no effort in optimizing their architectures.

This approach is conservative in the sense that results may only be better by improving the encoders and decoders.

2.2. Energy-based latent model

The learnable latent model structure can be chosen as general as represented in Fig. 2, ie. using an arbitrary NN to learn the relation between $\tilde{u}$ and $\tilde{f}$. One of the originality of our contribution is to propose the use of an energy structure in latent space inspired by the physical constitutive equation. Instead of choosing $\mathcal{F}_\theta$ as a NN, leading to a general $\mathrm{NN}(\tilde{u}) = \tilde{f}$ latent model, we choose $\mathcal{F}_\theta$ of the form $\mathcal{F}_\theta(\tilde{u}) = \partial_{\tilde{u}}\tilde{U}(\tilde{u})$ with $\tilde{U}$ a learnable energy function of $\tilde{u}$, so that the governing equation in the latent space $\partial_{\tilde{u}}\tilde{U} = \tilde{f}$ mimics the governing equation of the original problem $\partial_u \mathcal{U} = f$. For the roots to always exist, $\tilde{U}$ should be radially unbounded, which can be enforced by choosing $\tilde{U}(\tilde{u}) = \frac{1}{2}\tilde{u}^\top\tilde{u} + \mathrm{NN}(\tilde{u})$.

Computing the gradient of the latent energy with respect to $\tilde{u}$ is required in both DtF and FtD inference. Choosing the learnable energy $\tilde{U}(\tilde{u})$ as a Radial Basis Neural Network [10] plus a quadratic term to ensure that the energy is radially unbounded, makes its gradient with respect to $\tilde{u}$, ie. $\mathcal{F}_\theta$, simple enough to be computed analytically which allows a time-efficient implementation. Additionally, radial basis functions are more interpretable than LEBFC, whose weights are impossible to apprehend, contrary to radial basis functions.

Employing a latent energy structure allows us to solve the FtD problem by minimizing $\tilde{U}(\tilde{u}) - \tilde{u}^\top\tilde{f}$ wrt. $\tilde{u}$, which always converges to a local minimum. By contrast, Newton's algorithm can converge to a saddle point, see Fig. 6. In this case, the FtD inference is given by the Algorithm 2.

Algorithm 2 FtD inference using latent energy minimization

Input: f, u_0 (and learning rate α)
$\tilde{f} = \mathrm{enc}_f(f)$
$\tilde{u} = \mathrm{enc}_u(u_0)$
while $\|\frac{\partial\tilde{U}}{\partial\tilde{u}}(\tilde{u}) - \tilde{f}\| > \varepsilon$ **do**
  $\tilde{u} \leftarrow \tilde{u} - \alpha\dfrac{\partial}{\partial\tilde{u}}(\tilde{U}(\tilde{u}) - \tilde{u}^\top\tilde{f})$
end while
$u = \mathrm{dec}_u(\tilde{u})$
Output : u

3. Experiments

To validate the proposed approach, four different architectures are trained on several test cases described below. All the architectures have the same double autoencoder structure; they differ only by the learnable latent models, ie. by the choice of $\mathcal{F}_\theta$ in Fig. 2.

3.1. Compared latent models

The four different latent models compared are detailed below.

1. **LFC-f** $\tilde{u} = \mathrm{FCNN}(\tilde{f})$. This Latent Fully Connected (LFC) is direct in FtD mode, which erroneously prevents the existence of multiple FtD solutions. This architecture should therefore not be able to learn test cases with nonconvex potential energy. Besides, it requires a root-finding algorithm in DtF mode, leading to potential non-unique DtF solution, which is physically absurd. The advantage of this architecture is its very low FtD inference time, explained by the fact that it is not iterative contrary to the three other architectures below. It is analogous to the direct FtD NN architectures of [27, 29] (see Introduction).

2. **LFC-u** $\tilde{f} = \mathrm{FCNN}(\tilde{u})$. This latent model is direct in DtF mode and indirect in FtD mode, but has no energy structure. It should be able to learn test cases with nonconvex potential energy. It is analogous to the direct DtF NN architectures used in [8, 36] (see Introduction). FtD inference is achieved by a Newton–Raphson procedure in the latent space as described in algorithm 1.

3. **LEBFC** $\tilde{f} = \frac{\partial}{\partial\tilde{u}}\big(\frac{1}{2}\tilde{u}^\top\tilde{u} + \mathrm{FCNN}(\tilde{u})\big)$. This Latent Energy-Based (LEB) Fully Connected (FC) model, which is an original contribution of this work, is direct in DtF mode and is derived from a learnable latent energy, which is a quadratic function plus a fully-connected NN, as described in subsection 2.2. FtD inference uses latent energy minimization.

4. **LEBRB** $\tilde{f} = \frac{\partial}{\partial\tilde{u}}\big(\frac{1}{2}\tilde{u}^\top\tilde{u} + \mathrm{RBNN}(\tilde{u})\big)$. The only difference between LEBRB and LEBFC is the use of Radial Basis NN to learn the latent energy instead of FCNN. The terminology **LEBNN** encompasses

both. LEBRB is less general but allows time-efficient inference (see 2.2).

In the next subsections, we describe the test cases used in the comparison.

3.2. Test cases

This may be misleading but despite their geometric simplicity, the chosen examples are challenging due to their strong non-convex nonlinearities. On the contrary, learning a hyperelastic behavior on a complex geometry (in terms of number of degrees of freedom or curvature) may be very simple if the nonlinearity is small or large but convex. For instance, the easiest test case to learn in [39] is the one with the largest number of dofs, because it has the weakest nonlinearity.

3.2.1. Two-spring examples

The first test case is a 2-degree-of-freedom (dof) system: a mass connected to two springs, see Fig. 3. Although the springs' behavior laws are linear, the fact that the springs can rotate

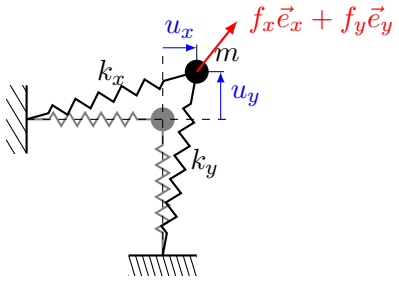

Figure 3: Two springs test case used to assess and compare the proposed architecture. Springs are linear, but the geometry is nonlinear (large displacements).

around their attachment points by large angles gives rise to geometric non-linearity. The strain energy of the structure can be shown to be:

$$\mathcal{U}(u_x, u_y) = \frac{1}{2}k_x(\sqrt{{u_y}^2 + (1+u_x)^2} - 1)^2 + \frac{1}{2}k_y(\sqrt{{u_x}^2 + (1+u_y)^2} - 1)^2. \tag{3.1}$$

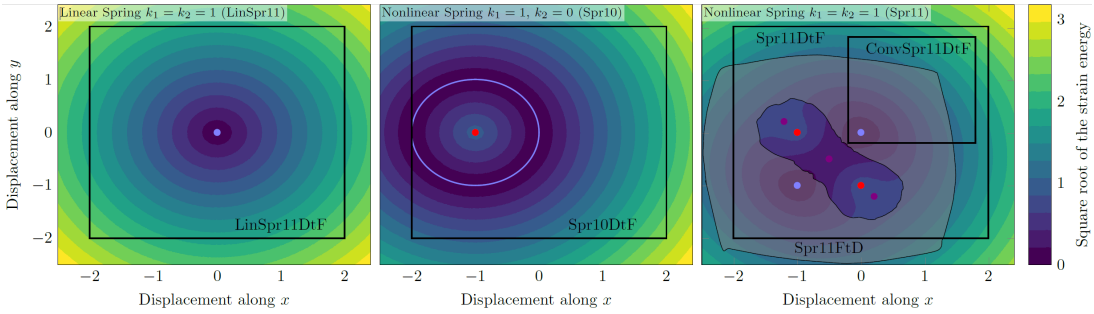

Figure 4: Strain energy for each spring test case. Local minimum: •. Local maximum: •. Saddle point: •. The envelope of the displacements of each dataset is depicted by the black curves/rectangles.

The convex properties of $\mathcal{U}$ depend on the values of k_x and k_y. To illustrate the capability of the NN architectures to learn various behaviors, we define the following subcases by varying the values for k_x, k_y and the range of u_x, u_y or f_x, f_y.

LinSpr11DtF dataset $k_x = k_y = 1$. $\nabla\mathcal{U}$ is linearized, leading to a quadratic strain energy:

$$\mathcal{U}(u_x, u_y) = \frac{1}{2}k_x u_x^2 + \frac{1}{2}k_y u_y^2. \tag{3.2}$$

u_x and u_y are sampled within $[-2, 2]$. The inverse problem (FtD) has a unique solution.

ConvSpr11DtF dataset $k_x = k_y = 1$. $\nabla\mathcal{U}$ is not linearized and are sampled from a subset ($u_x, u_y \in [-0.2, 1.8]$) on which the strain energy is convex. On this subset, the inverse problem (FtD) has a unique solution.

Spr11DtF dataset $k_x = k_y = 1$. $\nabla\mathcal{U}$ is not linearized and are sampled from a subset ($u_x, u_y \in [-2, 2]$) on which the strain energy is non-convex. On this larger subset, there are multiple FtD solutions. Up to 2 stable equilibria, 2 unstable equilibria and 3 saddle points coexist.

Spr11FtD dataset $k_x = k_y = 1$. $\nabla\mathcal{U}$ is not linearized and the force is sampled in a subset ($f_x, f_y \in [-2, 2]$) leading to displacements for which the strain energy is non-convex. There are multiple FtD solutions, but the dataset contains only stable equilibria because of FtD data generation. Up to 2 stable equilibria coexist.

Spr10DtF dataset $k_x = 1$, $k_y = 0$. $\nabla\mathcal{U}$ is not linearized, one of the springs has zero stiffness. The displacement is sampled from a subset ($u_x, u_y \in [-2, 2]$) on which the strain energy is non-convex. There are multiple FtD solutions, one stable equilibrium and one unstable equilibrium for a nonzero force, and a continuous circle of stable equilibria for $f = 0$, because the mass can rotate freely around the fixation point of the remaining spring.

For these test cases, data generation has been done either in DtF (sampling 1000 random u and computing $f = \frac{\partial \mathcal{U}}{\partial u}$) or in FtD (sampling 1000 random f and computing u with a root-finding algorithm). The FtD data generation leads to a dataset containing only stable equilibria whereas the DtF dataset also contain unstable equilibria (if they exist), see Fig. 4.

3.2.2. Hyperelastic beam

The second mechanical system is a hyperelastic fixed–free beam undergoing in the large transformation framework. The constitutive law is a St-Venant–Kirchhoff (stresses are proportional to strains); the non-linearity comes from the Green–Lagrange strain tensor. The beam was discretized in space with 10 elements in a 2-dimensional space, with a a clamped node, corresponding to 18 degrees of freedom. The discretized strain energy is a function of these 18 dofs and thus can not be visualized easily.

The beam parameters are the following: 10 nodes of which 1 is clamped, in a 2D plane (18 dofs), length $1\,\mathrm{m}$, width and thickness $0.01\,\mathrm{m}$, density 7.8 , Young modulus $210\,\mathrm{GPa}$.

Since multiple static equilibra can coexist for one force f, data could not be generated with a direct static solver. Instead, we used an in-house finite-element dynamic solver and applied a random oscillating force history at the free end of the beam, and then maintained it constant as f. With little damping ($\xi = 2\,\%$), the oscillations would stop and we would obtain a stable pair (f, u) solution of the static equilibrium equation. Using a dynamic solver was a way to perform FtD data generation, by approaching different stable solutions dynamically, hence overcoming the non-uniqueness of the solutions.As for the previous spring system, the maximum loading amplitude ($25\,\mathrm{kN}$) has been adjusted so that the non-linearity induced by the large displacement is sufficient to obtain non-unique FtD solution, see Fig. 5. The force f is chosen randomly within this range. The database includes 3778 (f, u) solutions. The corresponding (f, u) database is available in [32].

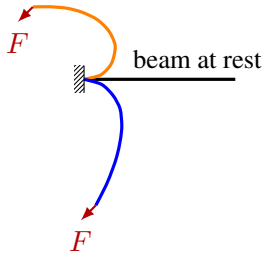

Figure 5: Test case of a clamped–free hyperelastic beam loaded at its tip. Two stable equilibria for the same given force are depicted.

3.3. Methodology

For all architectures, the training is performed in the direct mode, ie. without root-finding algorithm: in DtF mode, except for the LFC-f (in FtD mode). Latent models and autoencoders are trained simultaneously by minimizing the sum of the prediction error and the two reconstruction errors. Each error is weighted such that they have an equal contribution at the beginning of the training, and not adjusted during training.

For each test case, the dataset was randomly divided into $80\,\%$ for the training set and $20\,\%$ for the validation set. For each architecture and each test case, 10 trainings were done. The Adam optimizer [17] with default parameters and a learning rate chosen using a LRFinder algorithm [33].

3.4. DtF validation metrics

Given a displacement u (either for the spring-mass system or the beam), the corresponding reference force f_{obj} is compared with the NNs DtF predictions. Mean Absolute Error is computed between the predictions and the objective for every (u, f_{obj}) of the validation dataset.

The LFC-f architecture is not direct in DtF mode; the inverse problem is solved with a Newton method using a zero initial force f_0.

3.5. FtD validation metrics

Because the FtD solution is non-unique, an initial displacement u_0 has to be selected to compare the objective and the predictions. For the spring test cases, we built a (f, u_0, u) dataset by selecting random f and u_0 and computing the corresponding FtD solution from the governing equation. For the beam test case, u_0 was chosen using the (f, u) validation dataset as $u_0 = u/2$, which ensures that u and u_0 are in the same convex subset of the potential energy. Mean Absolute Error between predicted and objective displacements is then computed from these given f and u_0.

For the nonconvex cases, models can predict accurate stable equilibria that do not correspond to the given u_0: the root-finding algorithm may converge to a local minimum that is not the one reached by the mechanical system. This phenomenon is evaluated by computing the ratio of the number of predicted displacements in the right convex subset with the total number of cases: the metric is referred to as "success percentage" in Tab. 1.

4. Results

4.1. Comparisons of latent model structures

For the convex potential energies (test cases LinSpr11 and ConvSpr11), all four models learnt easily the system behavior with very high accuracy, see Tab. 1. In the case of convex potential energy, ie. always unique FtD solutions, there is little point in using an architecture requiring a root-finding algorithm in FtD inference. Direct FtD NN architecture such as LFC-f or architectures of [27, 29] are sufficient to reduce FtD inference in such simple cases.

On Spr11DtF and Spr10DtF ie. the two non-convex test cases with a DtF data generation, latent energy-based architectures outperform the two other architectures in terms of FtD evaluation metric, see Tab. 1. For such test cases, there are unstable equilibria and saddle points inside the dataset. FtD inference with LEBNN by energy minimization has a high success rate, better than LFC-u, in particular because Newton's method can converge to saddle points, see Fig. 6.

Error histograms in Fig. 11 show that the vast majority of samples lead to low errors (MAE < 0.05 for $u \in [-2, 2]$) but few samples have an error of the same error of magnitude as u. Such errors correspond to a FtD prediction being in the neighborhood of the wrong local minima, or close to a saddle point for LFC-u. Histograms show that this phenomenon is rare with LEB architectures.

LFC-u is not as inaccurate for the Spr11FtD and the beam. Indeed, such cases were generated by solving FtD problems, so that there are no unstable equilibria and saddle points in the training set to deteriorate the convergence of FtD inference with Newton. LFC-f is unable to learn multiple stable equilibria, leading to low accuracy.

Differences between LEBFC and LEBRB are insignificant, despite the lower inference time of LEBRB.

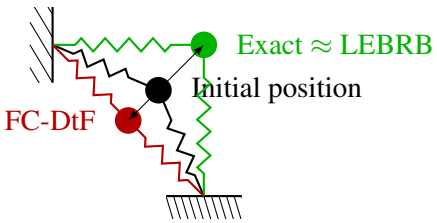

Figure 6: From an initial position (in black) in the vicinity of a saddle point of the potential energy of the system: FC-DtF converges to the (unstable) saddle point, while the LEBRB converges to the closest stable equilibrium by minimizing the latent energy.

4.2. Interpretability of LEBNN

The great advantage of LEBNN architectures is that they offer the possibility to visualize the latent energy, learnt only from a (u, f) database, ie. with no supervision of the energy. Fig. 8 proves that the latter embeds the complexity of the energy of the initial problem.

The latent energy has the right numbers of minima, maxima and saddle points in the zone covered by the training dataset, see Fig 9. The physical potential energy can be accurately reconstructed with LEBNN from the latent energy using enc_u. For the spring test case, this assertion can be directly verified,

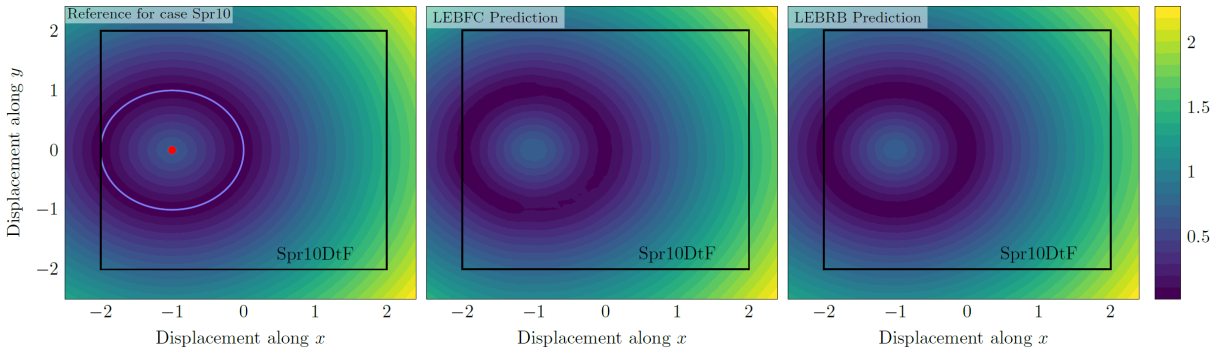

Figure 7: Contour plot of the strain energy for the Spr10 test case. Reference $\mathcal{U}(u_x, u_y)$ (left). Unsupervised prediction $\tilde{U}(\text{enc}_u(u_x, u_y))$ with LEBFC (center) and LEBRB (right). While being trained only on (u, f) pairs, both LEBNN architectures reconstruct the appropriate energy using only (u, f) pairs. Local minima: ○. Local maximum: ●. The envelope of the displacements of Spr10 dataset is depicted by the black rectangles.

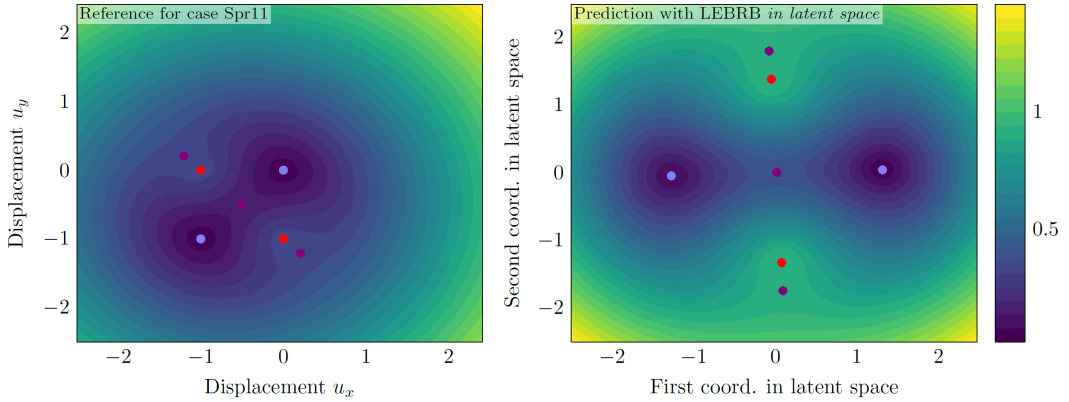

Figure 8: Contour plot of the strain energy for the Spr11 test case. Reference $\mathcal{U}(u_x, u_y)$ (left). Unsupervised prediction $\tilde{U}(h_1, h_2)$ with LEBRB (right). Energy in the physical and latent spaces have the same topology. In particular, they have the same equilibria.

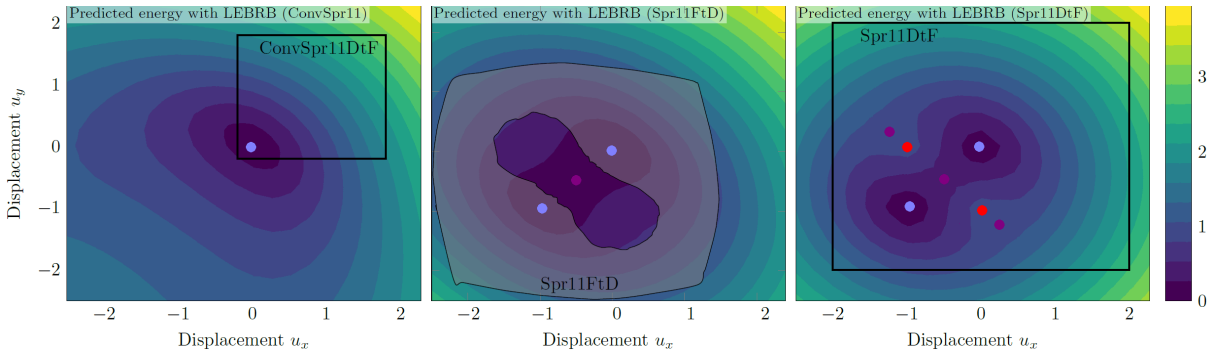

Figure 9: Contour plot of the strain energy learnt with the LEBRB for each dataset of Spr11, to be compared with levelsets in Fig. 8 (left). The energy in the physical space and in the latent space have the same topology. In particular, they have the same equilibria. Local minimum: ●. Local maximum: ●. Saddle point: ●.

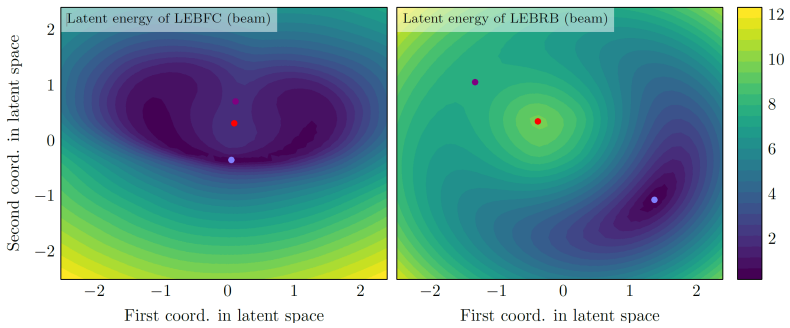

Figure 10: Contour plot of the strain energy in the latent space, for LEBFC (left) and LEBRB (right). Both have a single local minimum corresponding to the non-deformed beam and crescent-shaped level curves: for some $\tilde{f}$, there are 3 static equilibrium $\tilde{u}$ (two stable and one unstable), see e.g. Fig. 5.

see Fig. 7. For the beam, $u \in \mathbb{R}^{18}$ so that $\mathcal{U}(u)$ cannot be plotted. However, LEBNN allows plotting of its latent energy, which helps understand the system behavior: all the features of the system can be retrieved from the latent energy: single local minimum, crescent-shaped level sets, see Fig. 10.

The following provides an insight on the reason why LEBNNs work in FtD mode while trained only in DtF. The static problems FtD and DtF are both determined by the potential $\mathcal{U}$. Compressing this potential while preserving an energy structure in the latent space provides the ability to infer both displacements from forces and forces from displacements without being trained in both ways. This illustrates the sound architecture (latent energy) and an appropriate choice of latent space dimension: had it been chosen too large, the NN would have not been invertible. Chosen too small, the inversion would be inaccurate. The influence of the dimension of the latent space is discussed in Subsec. 4.4.

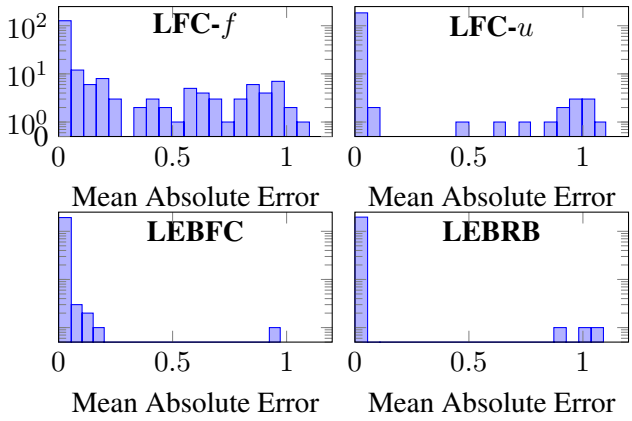

Figure 11: Error histograms for the four architectures in semi-log scale (Spr11DtF).

	LFC-f	LFC-u	LEBFC	LEBRB
LinSpr11				
- DtF MAE	3.1 ± 1.1	1.9 ± 0.4	1.0 ± 0.3	0.9 ± 0.2
- FtD MAE	3.2 ± 0.7	1.6 ± 0.5	1.4 ± 0.3	1.1 ± 0.3
- % success	100 ± 0	100 ± 0	100 ± 0	100 ± 0
ConvSpr11				
- DtF MAE	4.7 ± 1.7	2.5 ± 0.5	1.5 ± 0.2	1.6 ± 0.6
- FtD MAE	3.2 ± 0.6	1.7 ± 0.4	2.2 ± 0.5	2.6 ± 1.3
- % success	100 ± 0	100 ± 0	100 ± 0	100 ± 0
Spr11DtF				
- DtF MAE	$56. \pm 21.$	5.1 ± 0.4	2.4 ± 0.3	1.3 ± 0.3
- FtD MAE	$39. \pm 3.$	$15. \pm 5.$	5.7 ± 1.1	3.0 ± 1.5
- % success	83.7 ± 2.0	93.9 ± 1.3	98.6 ± 0.5	98.8 ± 0.6
Spr11FtD				
- DtF MAE	14.9 ± 1.2	13.8 ± 0.6	15.0 ± 1.6	10.3 ± 0.4
- FtD MAE	4.5 ± 0.7	4.6 ± 0.4	3.5 ± 1.0	3.7 ± 0.7
- % success	98.6 ± 0.2	99.1 ± 0.2	99.3 ± 0.4	99.1 ± 0.2
Spr10DtF				
- DtF MAE	13.7 ± 1.7	3.9 ± 0.3	2.5 ± 0.5	1.5 ± 0.2
- FtD MAE	26.1 ± 0.7	18.3 ± 2.6	5.3 ± 1.3	1.9 ± 0.6
- % success	90.8 ± 0.3	94.8 ± 0.7	99.4 ± 0.3	99.9 ± 0.1
Beam				
- DtF MAE	$62. \pm 14.$	7.9 ± 0.3	7.4 ± 0.7	5.5 ± 0.5
- FtD MAE	70.8 ± 0.2	17.9 ± 3.1	17.9 ± 1.9	14.9 ± 1.2
- % success	86.9 ± 0.1	97.8 ± 0.7	97.6 ± 0.4	97.9 ± 0.3

Table 1: Errors metrics evaluated for each dataset and architecture. Metrics were averaged on 10 trainings. Uncertainty is estimated by standard deviation. All MAEs were multiplied by 100. Metrics are defined in Sec. 3.

4.3. Chosen hyperparameters The following tables provide the NNs' architectures used for the above results (cf Tab. 2 and 3). The hyperparameters of the two autoencoders (number of layers, hidden layer size, activation functions) are the same for all four architectures and were not tuned. Using identity functions for enc_f and dec_f makes it possible to directly superimpose the levelsets of $\mathcal{U}$ and $\tilde{\mathcal{U}}$ in Figs. 7 and 9, but it is not compulsory, see for instance next paragraph. Corresponding codes are provided online [32].

enc_u	Lin(64,2) o ELU o Lin(64,64) o ELU o Lin(2,64)
dec_u	Lin(64,2) o ELU o Lin(64,64) o ELU o Lin(2,64)
enc_f	id
dec_f	id
case 1 (LFC-f)	$u = RN(f)$ with RN=Lin(64,2) o ELU o Lin(64,64) o ELU o Lin(2,64)
case 2 (LFC-u)	$f = RN(u)$ with RN=Lin(64,2) o ELU o Lin(64,64) o ELU o Lin(2,64)
case 3 (LEBFC)	$f = \frac{d}{du}(1/2u^\top u + RN(u))$ with RN = ELU o Lin(64,1) o ELU o Lin(64,64) o ELU o Lin(2,64)
case 4 (LEBRB)	$f = \frac{d}{du}(1/2u^\top u + RN(u))$ with RN = ELU o Lin(64,1) o RBF(2,64)

Table 2: Springs test case: architectures used to generate the results. $RBF(2, 64)$ denotes the radial basis functions defined as: $RBF(2, 64)(\tilde{u}) = \sum_{i=1}^{64} \sqrt{1 + (\frac{||\tilde{u}-c_i||}{s_i})^2}$.

enc_u	Lin(64,2) o ELU o Lin(64,64) o ELU o Lin(18,64)
dec_u	Lin(64,18) o ELU o Lin(64,64) o ELU o Lin(2,64)
enc_f	$\mathbb{R}^{18} \ni f \longmapsto \tilde{f} = [f_{17}, f_{18}] \in \mathbb{R}^2$
dec_f	$\mathbb{R}^2 \ni \tilde{f} \longmapsto f = [0, \dots, 0, \tilde{f}_1, \tilde{f}_2] \in \mathbb{R}^{18}$
case 1 (LFC-f)	same as in Tab. 2
case 2 (LFC-u)	same as in Tab. 2
case 3 (LEBFC)	same as in Tab. 2
case 4 (LEBRB)	same as in Tab. 2

Table 3: Beam test case: architectures used to generate the results.

4.4. Influence of the latent space dimension Fig. 12 depicts the FtD error as a function of DtF error for each architecture and different latent dimensions: 1, 2 and 3. Note that in this part, enc_f and dec_f were adapted from Tabs. 2 and 3 with an additional linear layer to match the latent space dimension. This linear layer was also added in latent dimension 2 for a fair comparison with latent dimensions 1 and 3. The errors are averaged on ten simulations for the springs test case and four simulations for the beam. The errors are much lower for the latent space of dimension 2 and the Latent-Energy Based architectures. For the beam case, despite displacement fields of dimension 18, they belong by construction to a 2D manifold: the FtD generation was carried out by varying the two components of an external force. Therefore, the resulting displacement fields can in theory be encoded in a 2-dimensional latent space.

A too small latent space dimension induces a loss of information and therefore large errors in DtF and FtD. That can be seen in Fig. 12, blue points. A too large latent space dimension does not induce any error in the direct sense, but its inversion with Newton's method leads to absurd results because it goes outside the manifold. This can be seen in Fig. 12: the architectures are much more accurate in the mode they have been trained. This gives a practical method to set the dimension of the latent space: it should match the number of parameters used to generate the database.

4.5. Influence of database size To get an insight on the influence of the dataset size, the four architectures were trained with a small dataset of only 40 points randomly chosen, and 10 additional points for the validation set. The batch size for the training is 8. Figure 13 shows that despite the small size of the dataset, LEBRB is able to learn the energy with a reasonable accuracy. Tab. 4 compares the FtD and DtF for the different architectures and assess them on an additional 1000 points (test set). It shows that the LEBNN architectures are much more accurate, illustrating the relevance of introducing a latent energy.

	LFC-f	LFC-u	LEBFC	LEBRB
Spr11DtF				
Training set (40 samples)				
- DtF MAE	212.6 ± 155.7	2.3 ± 0.5	4.8 ± 0.77	0.61 ± 0.31
- FtD MAE	35.8 ± 2.5	40.0 ± 18.8	12.4 ± 0.22	9.5 ± 2.7
- % success	87.5 ± 2.5	87.5 ± 2.5	97.5 ± 2.5	97.5 ± 2.5
Test set (1000 samples)				
- DtF MAE	163 ± 40.7	33 ± 1.8	16.7 ± 0.53	18.2 ± 0.02
- FtD MAE	83 ± 4.45	180 ± 121	18.2 ± 2.3	22.5 ± 1.01
- % success	76.7 ± 1.5	76.9 ± 5.05	96.4 ± 1.2	95.7 ± 0.25

Table 4: Errors metrics evaluated for each architecture on test case Spr11. Metrics were averaged on 10 trainings. All MAEs were multiplied by 100. Metrics are defined in Sec. 3.

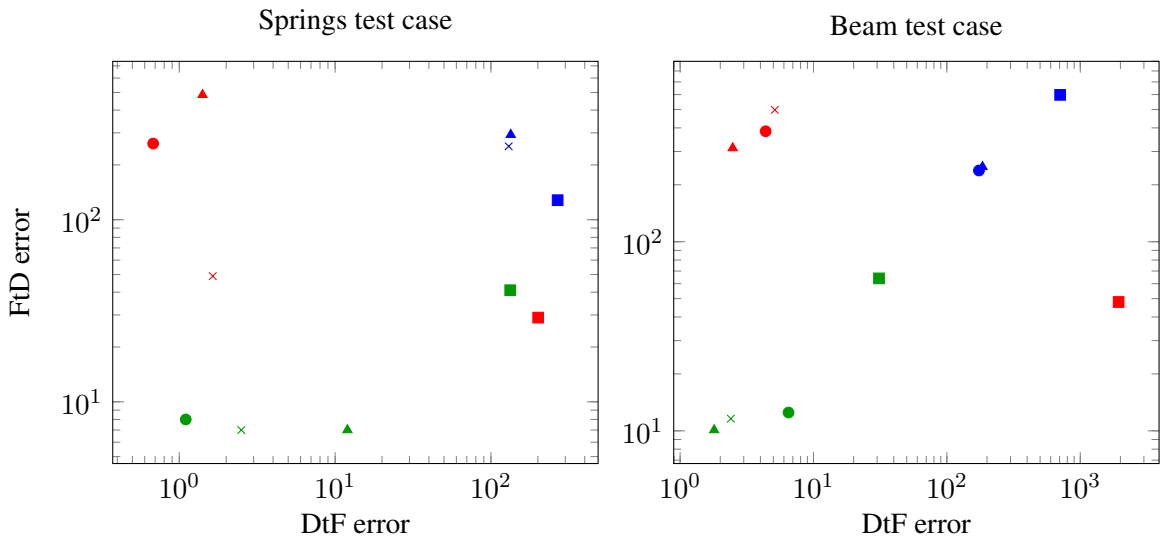

Figure 12: Errors in both modes (FtD and DtF) for the four architectures (■ LFC-f, ▲ LFC-u, ● LEBRB, × LEBFC) and different latent space dimensions: dimension 1, dimension 2, dimension 3.

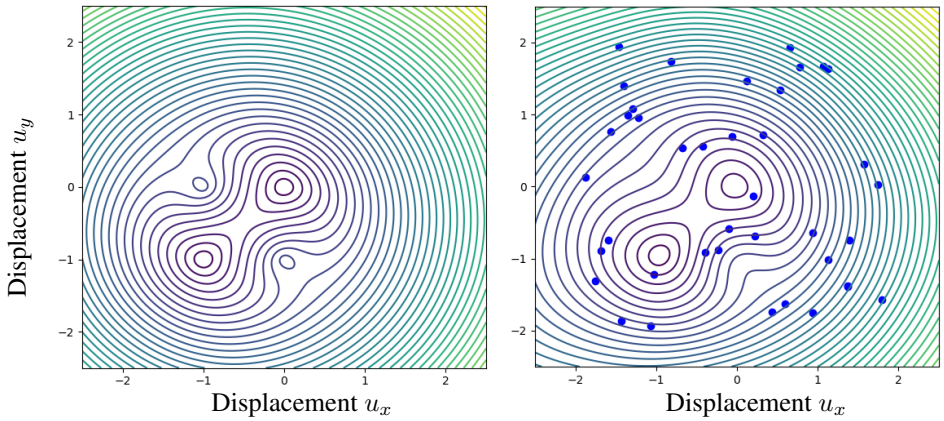

Figure 13: Reference energy (left) and predicted energy with LEBRB trained with only 40 samples (depicted in blue).

4.6. Comments on computation times

Fair cost comparison between finite elements and neural networks is difficult, because each of the methods depends on a lot of parameters: GPU/CPU, number of degrees of freedom, implementation, etc. Moreover, overhead costs are possibly significant for datasets as small as those considered here.

Algorithmic complexity is also case-dependent because of the structure of the involved matrices, the numbering of the nodes, etc. However, for finite elements, it is between at best $\mathcal{O}(n)$ for beams (band-diagonal structure independent of n) and at most $\mathcal{O}(n^3)$ (cost of a LU decomposition), with typical values between $\mathcal{O}(n^{2.3})$ and $\mathcal{O}(n^3)$. The complexity of the NN inference is also subtle, depending on whether the dimension of the latent space is fixed or not. For a given problem, the dimension of the latent space is expected to be insensitive to mesh refinement: it depends on the physical phenomenon rather than on the mesh refinement. Hence the inference of LEBNN in DtF mode scales as $\mathcal{O}(n)$, without the training phase. Another obstacle to fair comparison is the number of iterations in the Newton–Raphson's algorithm, which may depend on n and the proximity with an unstable equilibrium.

With all these limitations in mind, as an indication for the reader, computing a static equilibrium of a cantilever beam with 18 dofs using an optimized Finite Element dynamic code runs in about $5\,\mathrm{s}$ while the NN runs in FtD in about $5\,\mathrm{ms}$, with no effort in optimizing either simulation method.

5. Conclusion

Two NNs representative of already existing architectures (LFC-f and LFC-u) and two flavors of Latent-Energy Based NNs (LEBNN) have been compared on several test cases exhibiting various equilibrium typologies in the context of large strain hyperelasticity.

The different test cases include typologies where multiple solutions u to PDEs depending on a

parameter f can coexist and for which no existing architecture and training methods are suitable. In particular, a standard PINN architecture is incompatible with the multiplicity of FtD solutions and not suitable for model reduction with varying boundary condition f.

Two modes were considered: predicting forces in response to imposed displacements on a mechanical system, and predicting displacements in response to forces.

LEBNN is the only architecture leading to accurate results in both modes (despite being trained only in DtF mode). This result is even more significative when the size of the dataset is small, which is of practical interest (generating database can be costly). Additionally, LEBNN allows the visualization of the underlying energy of the system in a low-dimensional latent space, providing an intuitive understanding of its mechanical behavior and indicating that it has actually compressed sound physical information.

This work illustrates that including physical information within the NN architecture itself can be a fruitful approach. The counterpart of the non-uniqueness in the "force to displacement mode" is that LEBNN requires a Newton's method. As for any NN, it also requires a choice of hyperparameters. In particular, the dimension of the latent space must correspond to the physical intrinsic dimension for LEBNN to be relevant. For simple convex problems, there is no interest in using LEBNN: training two NNs LFC-f and LFC-u is sufficient to learn both FtD and DtF and there is no need for a Newton's method.

The end objective is to extend the idea of a latent-energy based NNs to dynamical systems in solid mechanics: for strongly nonlinear dynamical systems, $\mathcal{T}$ and $\mathcal{U}$ in the Lagrangian $\mathcal{L} = \mathcal{T} - \mathcal{U}$ play a very different role: the kinetic energy $\mathcal{T}$ may depend on u but is always convex (quadratic) in $\dot{u}$, while $\mathcal{U}$ often encompasses most of the nonlinearities. The present work, which precisely consists in learning $\mathcal{U}$, can therefore be seen as a possibly fruitful approach to learning the behavior of nonlinear dynamical systems, for instance with Lagrangian NNs, Hamiltonian NNs or NeuralODE.

6. References

[1] ABUEIDDA, Diab W., KORIC, Seid, GULERYUZ, Erman, SOBH, Nahil A. *Enhanced physics-informed neural networks for hyperelasticity*. International Journal for Numerical Methods in Engineering 124(7):1585–1601, 2022. [10.1002/nme.7176].

[2] ARDIZZONE, Lynton et al. *Analyzing inverse problems with invertible neural networks*. 2019. arXiv: 1808.04730 [cs.LG].

[3] BALL, John M. *Convexity conditions and existence theorems in nonlinear elasticity*. English. Archive for Rational Mechanics and Analysis 63(4):337–403, 1976. [10.1007/BF00279992].

[4] BONET, Javier, GIL, Antonio J., ORTIGOSA, Rogelio. *A computational framework for polyconvex large strain elasticity*, 2015. [10.1016/j.cma.2014.10.002].

[5] BUKKA, Sandeep, GUPTA, Rachit, MAGEE, Allan, JAIMAN, Rajeev. *Assessment of unsteady flow predictions using hybrid deep learning based reduced order models*. 2020.

[6] CHINESTA, Francisco, LADEVEZE, Pierre, CUETO, Elias. *A short review on model order reduction based on proper generalized decomposition*. ARCHIVES OF COMPUTATIONAL METHODS IN ENGINEERING 18:395–404, 2011. [10.1007/s11831-011-9064-7].

[7] CRANMER, Miles, GREYDANUS, Sam, HOYER, Stephan, BATTAGLIA, Peter, SPERGEL, David, HO, Shirley. *Lagrangian neural networks*. 2020. arXiv: 2003.04630 [cs.LG].

[8] DE BARRIE, Daniel, PANDYA, Manjari, PANDYA, Harit, HANHEIDE, Marc, ELGENEIDY, Khaled. *A deep learning method for vision based force prediction of a soft fin ray gripper using simulation data*. Frontiers in Robotics and AI 8, 2021. [10.3389/frobt.2021.631371].

[9] FENG, Yuan, WANG, Hexiang, YANG, Han, WANG, Fangbo. *Time-continuous energy-conservation neural network for structural dynamics analysis*. 2020. [10.48550/ARXIV.2012.14334].

[10] GHOSH, J., NAG, A. "An overview of radial basis function networks." *Radial Basis Function Networks 2: New Advances in Design*. Edited by Robert J. Howlett, Lakhmi C. Jain. Heidelberg: Physica-Verlag HD, 2001: 1–36. [ISBN: 978-3-7908-1826-0]. [10.1007/978-3-7908-1826-0_1].

[11] GOMEZ, Aidan N., REN, Mengye, URTASUN, Raquel, GROSSE, Roger B. *The reversible residual network: backpropagation without storing activations*. 2017. arXiv: 1707.04585 [cs.CV].

[12] GREYDANUS, Sam, DZAMBA, Misko, YOSINSKI, Jason. *Hamiltonian neural networks*. 2019. arXiv: 1906.01563 [cs.NE].

[13] GU, Yiqi, WANG, Chunmei, YANG, Haizhao. *Structure probing neural network deflation*. Journal of Computational Physics 434:110231, 2021. [10.1016/j.jcp.2021.110231].

[14] KIM, Byungsoo, AZEVEDO, Vinicius C., THUEREY, Nils, KIM, Theodore, GROSS, Markus, SOLENTHALER, Barbara. *Deep fluids: a generative network for parameterized fluid simulations*. Computer Graphics Forum 38(2):59–70, 2019. [10.1111/cgf.13619].

[15] KIM, Byungsoo, AZEVEDO, Vinicius C., THUEREY, Nils, KIM, Theodore, GROSS, Markus, SOLENTHALER, Barbara. *Deep Fluids: A Generative Network for Parameterized Fluid Simulations*. Computer Graphics Forum 38(2):59–70, 2019. arXiv: 1806.02071.

[16] KINDERMANN, J, LINDEN, A. *Inversion of neural networks by gradient descent*. Parallel Computing 14(3):277–286, 1990. [https://doi.org/10.1016/0167-8191(90)90081-J].

[17] KINGMA, Diederik P., BA, Jimmy. *Adam: a method for stochastic optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[18] KLEIN, Dominik K., FERNÁNDEZ, Mauricio, MARTIN, Robert J., NEFF, Patrizio, WEEGER, Oliver. *Polyconvex anisotropic hyperelasticity with neural networks*. Journal of the Mechanics and Physics of Solids 159:104703, 2022. [10.1016/j.jmps.2021.104703].

[19] LAWRENCE, Nathan P., LOEWEN, Philip D., FORBES, Michael G., BACKSTRÖM, Johan U., GOPALUNI, R. Bhushan. *Almost surely stable deep dynamics*, 2021. [10.48550/ARXIV.2103.14722].

[20] LEE, Kookjin, CARLBERG, Kevin T. *Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders*. Journal of Computational Physics 404:108973, 2020. arXiv: 1812.08373.

[21] LINDEN, Lennart, KLEIN, Dominik K., KALINA, Karl A., BRUMMUND, Jörg, WEEGER, Oliver, KÄSTNER, Markus. *Neural networks meet hyperelasticity: a guide to enforcing physics*. Journal of the Mechanics and Physics of Solids 179:105363, 2023. [10.1016/j.jmps.2023.105363].

[22] LINO, Mario, CANTWELL, Chris D., BHARATH, Anil A., FOTIADIS, Stathi. *Simulating continuum mechanics with multi-scale graph neural networks*. CoRR abs/2106.04900, 2021. arXiv: 2106.04900.

[23] LIU, Ruoshi, MAO, Chengzhi, TENDULKAR, Purva, WANG, Hao, VONDRICK, Carl. *Landscape learning for neural network inversion*. 2022. arXiv: 2206.09027 [cs.CV].

[24] LIU, Wenzhuo, YAGOUBI, Mouadh, SCHOENAUER, Marc. "Multi-resolution Graph Neural Networks for PDE Approximation." *Artificial Neural Networks and Machine Learning – ICANN 2021*. Volume 12893. Lecture Notes in Computer Science. Springer International Publishing, 2021:151–163. [10.1007/978-3-030-86365-4_13].

[25] LUTTER, Michael, RITTER, Christian, PETERS, Jan. *Deep lagrangian networks: using physics as model prior for deep learning*. 2019. arXiv: 1907.04490 [cs.LG].

[26] MANEK, Gaurav, KOLTER, J. Zico. *Learning stable deep dynamics models*. 2020. [10.48550/ARXIV.2001.06116].

[27] MENDIZABAL, Andrea, MÁRQUEZ-NEILA, Pablo, COTIN, Stéphane. *Simulation of hyperelastic materials in real-time using deep learning*. Medical Image Analysis 59:101569, 2019. [10.1016/j.media.2019.101569].

[28] MEYER, Lucas, POTTIER, Louen, RIBES, Alejandro, RAFFIN, Bruno. *Deep surrogate for direct time fluid dynamics*, 2021. [10.48550/ARXIV.2112.10296].

[29] ODOT, Alban, HAFERSSAS, Ryadh, COTIN, Stéphane. *Deepphysics: a physics aware deep learning framework for real-time simulation*. CoRR abs/2109.09491, 2021. arXiv: 2109.09491.

[30] PALIARD, Chloe, THUEREY, Nils, UM, Kiwon. *Exploring physical latent spaces for deep learning*. 2022. arXiv: 2211.11298 [cs.LG].

[31] PFAFF, Tobias, FORTUNATO, Meire, SANCHEZ-GONZALEZ, Alvaro, BATTAGLIA, Peter W. *Learning mesh-based simulation with graph networks*. 2021. arXiv: 2010.03409 [cs.LG].

[32] POTTIER, Louen, THORIN, Anders, CHINESTA, Francisco. *Database of force–displacement pairs with multiple static equilibria*. https://hal.science/hal-04737657. 2024.

[33] *Pytorch learning rate finder*. https://github.com/davidtvs/pytorch-lr-finder?tab=readme-ov-file. 2020.

[34] RAISSI, M., PERDIKARIS, P., KARNIADAKIS, G. E. *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*. Journal of Computational Physics 378:686–707, 2019. [10.1016/j.jcp.2018.10.045].

[35] SOSANYA, Andrew, GREYDANUS, Sam. *Dissipative hamiltonian neural networks: learning dissipative and conservative dynamics separately*. 2022. [10.48550/ARXIV.2201.10085].

[36] SUTO, K., SAKAI, Yusuke, TANIMICHI, K., OHSHIMA, T. "Surrogate model of elastic large-deformation behaviors of compliant mechanism using co-rotational beam element." 2022. [10.23967/wccm-apcom.2022.109].

[37] SWISCHUK, Renee, MAININI, Laura, PEHERSTORFER, Benjamin, WILLCOX, Karen. *Projection-based model reduction: formulations for physics-based machine learning*. Computers & Fluids 179:704–717, 2019. [https://doi.org/10.1016/j.compfluid.2018.07.021].

[38] TAC, Vahidullah, COSTABAL, Francisco, BUGANZA TEPOLE, Adrian. *Automatically polyconvex strain energy functions using neural ordinary differential equations*. 2021.

[39] THANGAMUTHU, Abishek, KUMAR, Gunjan, BISHNOI, Suresh, BHATTOO, Ravinder, KRISHNAN, N M Anoop, RANU, Sayan. *Unravelling the performance of physics-informed graph neural networks for dynamical systems*. 2023. arXiv: 2211.05520 [cs.LG].

[40] THUEREY, Nils, WEISSENOW, Konstantin, PRANTL, Lukas, HU, Xiangyu. *Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows*. AIAA Journal 58(1):25–36, 2020. [10.2514/1.j058291].

[41] ZHENG, Haoyang, HUANG, Yao, HUANG, Ziyang, HAO, Wenrui, LIN, Guang. *Hompinns: homotopy physics-informed neural networks for solving the inverse problems of nonlinear differential equations with multiple solutions*. Journal of Computational Physics 500:112751, 2024. [10.1016/j.jcp.2023.112751].