Response to Reviewers

In this document, we, the authors of the manuscript *Non-intrusive reduced-order modeling of nonlinear problems using neural networks*, reply to the referee report by two anonymous reviewers, to whom we are grateful for the time devoted to our paper and their valuable, constructive and pleasing comments. For the sake of clarity, we report the original comments in italic and we inline our responses, so that each issue raised by the reviewers is addressed separately. If revisions to the original manuscript are made, we mention the page, paragraph and line at which they occur. The modifications in the revised manuscript are highlighted with different colours for different reviewers; the colour which is associated with a reviewer is specified in the title of the corresponding section.

Reviewer #1

The paper presents an innovative procedure to compute the coefficients of the reduced order model constructed through Proper Orthogonal Decomposition using a set of high-fidelity snapshots. The paper represents a permanent value to the literature of Reduced Order Models. The non-intrusive method proposed in this study allows the reconstruction of the reduced model avoiding the complexity and the computational burden of a Galerkin procedure to compute the projection coefficients. The non-intrusive method presented in this paper relies to multi-layer perceptrons for the prediction of the ROM coefficients. Neural Networks constitute a valid alternative to conventional interpolation techniques which present limitations when applied to nonlinear matrix manifolds. This method is independent of the original size of the problem, hence it represents a significant computational effort reduction with respect to full order models. The theoretical background is clearly explained without being two lengthy and previous works are appropriately referenced. Numerical results supporting technical claims clearly presents the advantages of the proposed method over conventional Galerkin procedures. Reviewer recommends acceptance of the paper.

No response required.

Reviewer #2 (red)

The idea of employing neural networks to interpolate the projection coefficients in a non-intrusive POD reduced basis method is certainly novel and interesting. The paper is concise and clearly written, and the numerical examples provide a good demonstration of the proposed methodology. To further improve this work I would recommend providing the following clarifications.

1. *Much of the recent practical success of neural networks is attributed to advances in the way they are initialized, tuned, and optimized (see for e.g. LeCun, et al. "Efficient backprop." Neural networks: Tricks of the trade, 1998). For instance, (mostly empirical) techniques in network initialization, data normalization, and adaptations of the the Robbins-Monro approach to stochastic optimization have led to more robust neural network training procedures. The authors here give an exhaustive overview of the formulation of neural networks, but not much details on their practical tuning for the proposed methodology, other than that the Levenberg-Marquardt algorithm is used. Given that the neural network training procedure is known to define a non-convex optimization problem that returns non-identifiable (i.e., non unique) network weights, it would be very helpful if the authors could provide more details regarding how the networks are trained in practice, e.g., is the data normalized and how? how is the network initialized? what is the learning rate (i.e. step size or damping factor) for the parameter updates? etc. That would also greatly benefit anyone that would like to try reproduce your results.*

   Let $\mathbf{t_p} \in \mathbb{R}^{M_O}$ and $\mathbf{y_p} \in \mathbb{R}^{M_O}$ be respectively the desired output and the actual output corresponding to the input $\mathbf{p} \in P$, P being the training set, and define $\mathbf{e_p} := \mathbf{t_p} - \mathbf{y_p}$. Assuming a quadratic performance function E, i.e.,

$$E = E(\mathbf{w}) = \sum_{\mathbf{p} \in P} \left\| \mathbf{e_p}(\mathbf{w}) \right\|^2 ,$$

   and denoting by $J_\mathbf{p}$ the Jacobian of $\mathbf{e_p}$ with respect to the weights vector $\mathbf{w}$, each iteration of the Levenberg-Marquardt

(LM) algorithm entails the resolution of the linear system

$$\left[\sum_{\mathbf{p}\in P} J_{\mathbf{p}}(\mathbf{w})^T J_{\mathbf{p}}(\mathbf{w}) + \mu I\right] \Delta\mathbf{w} = -\sum_{\mathbf{p}\in P} J_{\mathbf{p}}(\mathbf{w})^T \mathbf{e}_{\mathbf{p}}(\mathbf{w}), \tag{1}$$

where $\Delta\mathbf{w}$ is the weight update, $\mu \geq 0$ is a bias parameter and I denotes the identity matrix of appropriate dimensions. Note that if $\mu = 0$ we recover Newton's method (see, e.g., [3]), while for $\mu \gg 1$ the search direction $\Delta\mathbf{w}$ approaches the antigradient of E, i.e., we recover the backpropagation algorithm (see, e.g., [5]). Hence, the LM algorithm can be seen as an interpolation between Newton's method and the steepest descent method, aiming to retain the advantages of both techniques.

The algorithm proceeds as follows. At each epoch t, we solve the linear system (1). If $\Delta\mathbf{w}$ leads to a reduction in the performance function E, i.e., $E(\mathbf{w}(t) + \Delta\mathbf{w}) < E(\mathbf{w}(t))$, the parameter μ is reduced by a factor $\beta > 1$. Conversely, if $E(\mathbf{w}(t) + \Delta\mathbf{w}) < E(\mathbf{w}(t))$, μ is multiplied by the same factor β. This reflects the idea that far from the actual minimum of the surface error E one should prefer the gradient method to Newton's method, since the latter may diverge. Yet, once in a neighborhood of the minimum, we switch to Newton's method to exploit its faster convergence [6]. As mentioned at page 9, Section 4.3, lines 328 – 330, we set $\mu = 0.001$ as starting point for μ and $\beta = 10$.

Further practical details regarding the weights initialization, the multiple restart approach and the early stopping technique which we implemented have been added at page 13, Section 6, lines 436 – 440. Moreover, it should be pointed out that no sort of normalization has been applied to training data in our simulations.

2. *It would be helpful if the authors could motivate the choice of neural networks a bit better, and provide a brief comparison with alternative nonlinear interpolation methods, especially keeping in mind the scenario of scarce training data-sets. For example, in a very similar setting (albeit not related to reduce basis methods) Parussini, et al. "Multi-fidelity Gaussian process regression for prediction of random fields." J. Comput. Phys. (2017) motivated the use of multi-fidelity regression using Gaussian processes for interpolating modal coefficients. Could you provide a short comment on why you believe that neural networks are the right tool for the given interpolation problem?*

   As mentioned in the Introduction (see page 2, lines 57 – 61), standard interpolation techniques generally fail in enforcing the nonlinear constraints typical of reduced bases manifolds, unless using a large amount of samples, i.e., snapshots. On the contrary, feedforward neural networks are well-known and widely-used for their capability in fitting (highly) nonlinear maps. Hence, one may reasonably expect that neural networks require less training data as compared to a traditional interpolation method. Moreover, the hope is that the size of the training set nicely scales with the dimension of the parameter domain. This would be extremely beneficial in real-life applications involving thousands of parameters, and should be further investigated in the future.

   In Section 6.1.2, we already motivate the use of neural networks by comparing the POD-NN method with a cubic splines-based non-intrusive reduced basis method, which is briefly introduced at page 18, lines 548 – 559. The results, visualized in Fig. 6.7 and commented at lines 560 – 571, support our choice.

3. *Most contemporary approaches for training neural networks rely on variants of gradient descent (e.g. Adam optimizer), or quasi-Newton optimizers like L-BFGS. How does the Levenberg-Marquardt compare to such approaches? Do you see any immediate shortcomings or advantages of one approach versus the other?*

   Figure 6.2 at page 15, Section 6.1.1, has been enriched with the two rightmost plots, comparing the LM algorithm against two other famous weight-updating routines: the scaled conjugate gradient (SCG) [7] and the BFGS quasi-Newton method [2]. The comparison regards the test error (*centre*) and the average training time (*right*) for different numbers of training samples. Results are commented within the same page at lines 502 – 512. In a nutshell: LM turns out to be much more accurate than both SCG and BFGS, entailing a reasonable time overhead with respect to its competitors. Yet, it is well recognized that the performance of LM quickly degrades as the number of free parameters in the network exceeds a few hundreads units (see, e.g., [4]). Hence, the analysis should be repeated in the case so-large networks are employed.

4. *Some of the terminology used to define the neural network seem to be outdated. For example, terms like "input pattern", "activation pattern", "teaching input", "training pattern" are not well aligned with the current neural network*

literature. For instance one would simply refer to "input data", "activation", "training inputs", etc. I think this introduces unnecessary jargon, and the presentation can be simplified in benefit of the reader.

Following the Reviewer's suggestions, we have applied the following modifications:

- replace "activation pattern" with "input vector" (page 7, Section 4.1, line 266; page 8, Section 4.2, line 275);
- replace "input pattern" with either "input vector" (page 9, Section 4.3, lines 325 and 326) or "training input" (page 9, Section 4.3, line 307; page 10, Section 5, line 368);
- replace "(training) patterns" with "training data" (page 9, Section 4.3, lines 298 and 302; page 11, Section 5, line 387; page 15, Section 6.1.1, line 487; page 18, Section 6.1.2, line 562);
- replace "teaching input" with either "desired output" (page 9, Section 4.3, lines 307 and 325; page 10, Section 5, line 368) or "output vector" (page 10, Section 5, line 368);
- replace "teaching inputs" with "teaching data" (page 10, Section 5, line 369);
- replace "learning patterns" with "learning data" (page 17, Section 6.1.2, lines 541 and 542).

5. *Although Cybenko's classical results justify the use of networks with no more than two hidden layers (at least in theory), much of the recent resurgence of neural networks is attributed to the performance and efficiency demonstrated by deeper architectures. Could you provide a comment on any advantages vs disadvantages between deep/narrow architectures versus the shallow/wide network architectures used here.*

   Relying upon the Cybenko's results, we limit ourselves to three-layers feedforward neural networks mainly to ease the research of an optimal network architecture. Indeed, by fixing the number of hidden layers, each one populated with H neurons, we can conduct a convergence analysis with respect to solely two free parameters: the amount of training samples N_{tr} and H itself. This also allows for a clear and effective visualization of the results (see, e.g., Fig. 6.1 (*left*) or Fig. 6.6). However, the more complex the underlying differential problem, the larger the number of neurons we may need, and so the wider the network. Since the number of free parameters (i.e., connections) of a fully connected perceptron is linear in the number of layers but quadratic in the number of neurons per layer, then it might be convenient to adopt deep-and-narrow rather than shallow-and-wide networks. Further, the former are known to reduce the risk of overfitting, as each layer learns at a different level of abstraction. This, coupled with the increasing computational power available, motivates the large attention deserved by deep neural networks in the last years. On the other hand, the deeper the network, the slower backpropagation learns at weights close to the input layer [5]. Hence, extending the automatic routine outlined at page 11, one may proceed as follows:

   - start with a three-layers configuration and increase the number of neurons and training samples;
   - when widening the network does not further improve its performance, re-arrange the hidden neurons in multiple layers;
   - keeping the number of neurons per layer fixed, add new layers until a desired level of accuracy is reached, or resources are exhausted.

6. *Arguably the online run time is key in this setting. However, for completeness, could you provide a brief comparison of the offline computational cost associated with the FE, POD-G and POD-NN approaches.*

   As pointed out in the comment, the online run time is a key factor in the context of reduced order modeling (ROM). Ideally, it should be independent of the size of the original problem - as in the case of the POD-NN method. Conversely, the computational cost associated with the offline phase depends not only on the full order model, but also on the degree of predictive accuracy which the reduced order model should ensure. As an example, at page 17, Section 6.1.2, lines 535 – 537, we have reported the offline run time for the POD-G and POD-NN methods applied to the two-dimensional Poisson problem considered in the paper. It turns out that training the neural networks extends the offline phase by roughly a factor 10, as compared to the POD-G algorithm. Yet, POD-NN processes any only query 10^3 times faster than POD-G does.

7. *In terms of future work, do you foresee such methods becoming applicable beyond steady-state problems (i.e. problems with intermittent dynamics)? What are the main limitations there?*

As said in the Conclusion (page 25, line 704), the extension of the proposed method to time-dependent parameterized problems should be investigated in the future. In this respect, let us remark that the generation of the reduced space is totally decoupled from the training and evaluation of the neural networks (page 26, Conclusion, lines 706 – 707). For instance, in the case of Hamiltonian systems, a reduced basis could be extracted via the structure-preserving greedy approach proposed by Afkham & Hesthaven [1]. Further, ROMs built via the POD-NN algoruthm seem to have better stability properties, as compared to traditional projection-based ROMs. This speculation is based on the results regarding the lid-driven cavity problem, where the POD-NN ROM turns out not to require any stabilizing technique. Hence, we believe that our method could be applied to a broad class of PDE problems.

References

[1] Afkham, B. M., Hesthaven, J. S. (2017). *Structure preserving model reduction of dissipative Hamiltonian systems*. Submitted to Journal of Scientific Computing.

[2] Gill, P. E., Murray, W., Wright, M. H. (1981). *Practical optimization*. Academic Press.

[3] Hagan, M. T., Menhaj, M. B. (1994). *Training feedforward networks with the Marquardt algorithm*. IEEE Transactions on Neural Networks, 5(6):989-993.

[4] Hagan, M. T., Demuth, H. B., Beale, M. H. (1996). *Neural Network Design*. Boston, MA: PWS Publishing.

[5] Kriesel, D. (2007). *A Brief Introduction to Neural Networks*. Retrieved from `http://www.dkriesel.com/en/science/neural_networks`.

[6] Marquardt, D. W. (1963). *An algorithm for least-squares estimation of nonlinear parameters*. Journal of the Society for Industrial and Applied Mathematics, 11(2):431-441.

[7] Møller, M. D. (1993). *A scaled conjugate gradient algorithm for fast supervised learning*. Neural Networks, 6:525-533.