POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Physics-informed machine learning methods for reduced-order modeling

Laurea Magistrale in Aeronautical Engineering - Ingegneria Aeronautica

Author: Riccardo Tomada

Advisor: Prof. Edie Miglio

Academic year: 2021-2022

1. Introduction

To solve physical problems governed by Partial Differential Equations, a wide number of numerical schemes has been developed, such as the Finite Differences Method (FDM), the Finite Elements Method (FEM) and the Finite Volume Method (FVM). The required discretization in time and space leads to the so called Full Order Models (FOMs), which are in general high-dimensional systems. While these methods are able to achieve high accuracy on a vast set of problems, high-dimensionality of the FOMs becomes an issue as soon as the PDEs are parameterized.

To alleviate this, some dimensionality reduction techniques have been traditionally devised which fall under the category of Reduced Order Models (ROMs). The major hypothesis at the base of the ROMs is given by the observation that even the behaviour of a complex system can be often described by a combination of few dominant modes.

The most widespread dimensionality reduction technique is the Proper Orthogonal Decomposition (POD). This method exploits the Singular Value Decomposition (SVD) to extract a set of reduced basis, which represent the dominant dynamics of the underlying problem. The main drawback of SVD-based ROMs lies in the fact that this matrix factorization is essentially a linear technique, therefore the method may suffer especially if the problem at hand is unsteady or advection dominated.

In the present work two new approaches will be investigated with the aim to improve the accuracy provided by the existing linear ROMs. In the first one, the Projection Driven Neural Networks - Autoencoder (PDNNs-Autoencoder), the SVD is replaced by a deep convolutional autoencoder. The second approach, namely the Physics Informed Neural Networks - FOM (PINNs-FOM), is completely different from a conceptual point of view. This method consists of an artificial neural network in which the governing equations of the problem at hand are directly embedded in its training algorithm. Both these methods will be assessed on different test problems and compared against SVD-based ROMs.

2. Reduced Order Models

ROMs strategies are usually implemented in an offline - online paradigm. During the offline stage, a set of high fidelity solutions is collected (snapshots). These are in turn used to extract the reduced basis (RB). During the online stage

instead the reduced coefficients are computed. The solution is then recovered as a linear combination of the RB functions weighted by the reduced coefficients.
The Proper Orthogonal Decomposition (POD) leverages the Singular Value Decomposition (SVD) of a matrix representative of the problem dynamics to retrieve a low rank approximation. Let us consider a matrix $\boldsymbol{A} \in \mathbb{C}^{n \times m}$. This matrix, in the context we are dealing with, consists of a series of high fidelity solutions which are sampled uniformly or randomly over the domain and then appropriately reshaped into column vectors $\boldsymbol{a}_k$.

$$\boldsymbol{A} = \begin{bmatrix} | & | & & | & & | \\ \boldsymbol{a}_1 & \boldsymbol{a}_2 & ... & \boldsymbol{a}_k & ... & \boldsymbol{a}_m \\ | & | & & | & & | \end{bmatrix} \tag{1}$$

In general, $n >> m$, since the mesh used for the computation of the FOM solution is usually fine, and the number of snapshots collected cannot be too high (otherwise the use of a ROM is no more time convenient). The SVD is a unique matrix decomposition which ensures the existence of two unitary matrices $\boldsymbol{U} = \left[\boldsymbol{u}_1|...|\boldsymbol{u}_n\right] \in \mathbb{C}^{n \times n}$ and $\boldsymbol{V} = \left[\boldsymbol{v}_1|...|\boldsymbol{v}_m\right] \in \mathbb{C}^{m \times m}$ with orthonormal columns and a matrix $\boldsymbol{\Sigma} = diag(\sigma_1, ...\sigma_m) \in \mathbb{R}^{n \times m}$ characterized by non-negative real values on the diagonal and zero off it, such that:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*, \tag{2}$$

with * being the Hermitian transpose operator. Since $n > m$, $\boldsymbol{\Sigma}$ has at most m nonzero entries on the diagonal, and thus can be rewritten as:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \hat{\boldsymbol{\Sigma}} \\ \boldsymbol{0} \end{bmatrix}. \tag{3}$$

Therefore, the *economy* SVD can be defined:

$$\boldsymbol{A} = \begin{bmatrix} \hat{U} & \hat{U}^{\perp} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\Sigma}} \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{V}^* = \hat{\boldsymbol{U}}\hat{\boldsymbol{\Sigma}}\boldsymbol{V}^*. \tag{4}$$

The Eckart - Young theorem [1] guarantees that the low rank-r approximation given by the SVD is the optimal one in the least squares sense:

$$\arg\min_{\tilde{\boldsymbol{A}}} \left\| \boldsymbol{A} - \tilde{\boldsymbol{A}} \right\|_F = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}}^*, \tag{5}$$

where $\tilde{\boldsymbol{U}}$ and $\tilde{\boldsymbol{V}}$ are respectively the first r leading columns of $\boldsymbol{U}$ and the first r leading columns of $\boldsymbol{V}$, and $\| \cdot \|_F$ denotes the Frobenius norm.

The most relevant drawback of the SVD is that if the data are not aligned, the relevant singular values of the underlying problem increase. This means that the number of columns of $\boldsymbol{U}$ which must be considered to accurately describe the dynamics becomes higher, thus lowering the dimensionality reduction capability of the whole POD procedure.

3. Autoencoders

Artificial Neural Networks (ANNs) represents the most famous and nowadays widespread category of Deep Learning algorithms. Autoencoders are a class of ANNs used in the field of data reconstruction and dimensionality reduction. The network is trained to learn the identity matrix, i.e., to reconstruct the input (after compressing it in a low-dimension space) and it is made up of two portions: the encoder $E(\boldsymbol{x})$ and the decoder $D(\boldsymbol{x})$. More powerful and non-linear representations can be learned by stacking multiple hidden layers. These networks are called deep autoencoders.
The reconstruction loss is given by:

$$\mathcal{L}_{AE}(\boldsymbol{w}) = \frac{1}{N} \sum_{\boldsymbol{x} \in \boldsymbol{S}} \| D(E(\boldsymbol{x})) - \boldsymbol{x}) \|_2^2, \tag{6}$$

where $\boldsymbol{S}$ is the data set containing N generic input arrays $\boldsymbol{x}$.

3.1. PDNNs - Autoencoder

PDNNs consists of a purely non-intrusive data-driven ROM approach. In particular, PDNNs are built and trained in order to obtain a continuous map between samples of the parameter space and the corresponding projection coefficients of the ROM.
The offline stage starts with the collection of the FOM problem snapshots $\boldsymbol{u}_h(t, \boldsymbol{\mu})$ sampled at different values of the parameters $\boldsymbol{\mu}$ and time t. In the present work the Finite Element Method has been implemented to compute the high-fidelity solution. To this aim, the python library FEniCS [3] has been leveraged.
The snapshots go through a preprocessing phase in which they are standardized. In this way the training process gains a speed-up and the accuracy of the reconstructed snapshots increases.
The autoencoder architecture is a mixed convolutional - fully connected one.

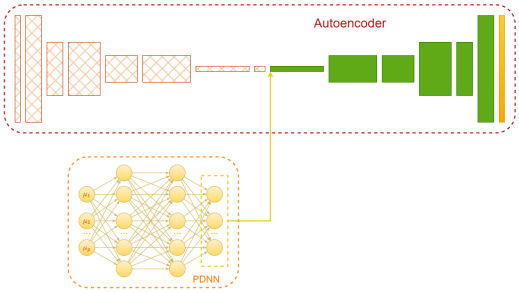


Figure 1: PDNN - Autoencoder online stage. Note that the encoder is not being used in this process.

The loss function to be minimized during the autoencoder training is given by:

$$\mathcal{L}_{AE} = \frac{1}{m} \sum_{\boldsymbol{u}_h(t,\boldsymbol{\mu}) \in \boldsymbol{A}} \| D(E(\boldsymbol{u}_h)) - \boldsymbol{u}_h \|_2^2 , \quad (7)$$

where m corresponds to the amount of available snapshots.
Once the autoencoder is successfully trained, the encoder portion is fed with the snapshots corresponding to the sampled time-parameter instances and it outputs their latent representations. In this way it is possible to construct a dataset $\overline{\mathcal{D}}$ of dimension $N_{\overline{D}}$ for the training of the associated PDNN, which consists of the time-parameters instances along with their corresponding latent representation.
This dataset is finally used to train the PDNN in a supervised learning context in order to learn the map $\boldsymbol{g} : \mathbb{R}^{N_p+k} \rightarrow \mathbb{R}^r$ between the time-parameters space and the latent representation one, where again k is 1 if the problem considered is time-dependent, zero otherwise.
The loss to be minimized in order to optimize the PDNN weights is:

$$\mathcal{L}_{PDNN} = \frac{1}{N_{\overline{D}}} \sum_{(t,\boldsymbol{\mu}), E(\cdot) \in \overline{\mathcal{D}}} \| \boldsymbol{g}(\cdot) - E(\boldsymbol{u}_h) \|_2^2 . \quad (8)$$

The online stage (see Figure (1)) is performed by evaluating the PDNN at the desired time-parameter instance and feeding the corresponding latent representation to the decoder which in turns outputs the ROM solution:

$$\hat{\boldsymbol{u}}_h(t,\mu) = D(\boldsymbol{g}(t,\boldsymbol{\mu})). \quad (9)$$

4. Physics Informed Neural Networks

Traditional ANNs exploit the backpropagation algorithm [6] to compute the derivatives of the loss function with respect to the network weights. In the same way, the backpropagation algorithm can be used to compute other derivatives, i.e. the derivative of the output with respect to the network inputs.
Physics Informed Neural Networks (PINNs) are made of a traditional ANN architecture. The input layer is fed with the spatio-temporal coordinates array $\boldsymbol{x}$. In the case of a parameterized PDE, the parameters are treated as coordinates along additional domain dimensions. The output layer delivers a surrogate of the solution $\hat{u}(\boldsymbol{x})$. In addition to a classical ANN, a PINN presents a backpropagation block in which the derivatives of the output are computed with respect to the input layer variables. In such manner it is possible to force the network to comply with the constraints imposed by both the PDE and its initial & boundary conditions.
To measure the error of the network, $\hat{u}(\boldsymbol{x})$ and its derivatives with respect to the inputs are evaluated on a number of points, named collocation points. The values obtained are then substituted inside the PDE and ICs-BCs definitions to retrieve their residuals. The loss function to be minimized is usually constructed as the weighted sum of the L^2-norm of the residuals:

$$\mathcal{L}(\boldsymbol{w}; \mathcal{T}) = \alpha_f \mathcal{L}_f(\boldsymbol{w}; \mathcal{T}_f) + \alpha_b \mathcal{L}_b(\boldsymbol{w}; \mathcal{T}_b), \quad (10)$$

where $\boldsymbol{w}$ are the weights of the network, $\mathcal{T}$ is the set of collocation points, divided into the ones inside the domain to test the PDE residual ($\mathcal{T}_f$) and the ones on the boundaries to test the boundary conditions ($\mathcal{T}_b$). Moreover, $\mathcal{L}_f(\boldsymbol{w}; \mathcal{T}_f)$ is the MSE of the PDE residual, while $\mathcal{L}_b(\boldsymbol{w}; \mathcal{T}_b)$ is the MSE deriving from the ICs-BCs imposition. Finally, α_f and α_b are the respective weighting coefficients, which are generally chosen such that the two terms become of the same order of magnitude.
The minimization of Equation (10) is performed in order to determine the optimal weights $\boldsymbol{w^*}$ of the network. The most widespread optimizers in the PINNs context are Adam [2] and L-BFGS [4]. Usually the training is initially performed via Adam to "warm-up" the weights and

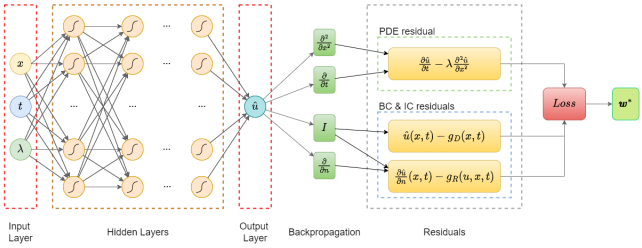


Figure 2: Example of a Physics Informed Neural Network Architecture for solving the 1D diffusion PDE $\frac{\partial u}{\partial t} = \lambda \frac{\partial^2 u}{\partial x^2}$ with mixed Dirichlet - Robin BC.

then continued switching to the second-order optimizer L-BFGS, which in general is more efficient. The complete architecture of a PINN can be visualized in Figure (2).

4.1. PINNs - FOM

Setting up the parameterized problem is straightforward. With respect to a generic non-parameterized case, the only task to be performed is to treat the parameter as an additional input coordinate, just as the spatio-temporal ones. Moreover, since the dimensionality of the problem is increased, it is likely that the network width and depth, the number of residual points to be sampled and the training epochs required will be higher than the non-parameterized counterpart.
Apart from these important considerations, the rest of the algorithm is the same as explained above. The neural network architecture employed for this task can be simply a traditional FFNN with problem-dependent hyperparameters to be tuned.
The high-level python library DeepXDE [5] has been exploited to implement this procedure.

5. Results

The proposed strategies has been tested against two linear SVD-based ROM approaches, namely the PDNNs-SVD and the POD-G-NN, the latter only in the steady cases. To evaluate the performances of the different strategies we rely on the L_2 relative error indicator, defined as:

$$e_{l_2} = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \frac{\|\boldsymbol{u}_h(t_i, \boldsymbol{\mu}_i) - \hat{\boldsymbol{u}}(t_i, \boldsymbol{\mu}_i)\|_2}{\|\boldsymbol{u}_h(t_i, \boldsymbol{\mu}_i)\|_2} \quad (11)$$

over the test sets built for each case. As concern

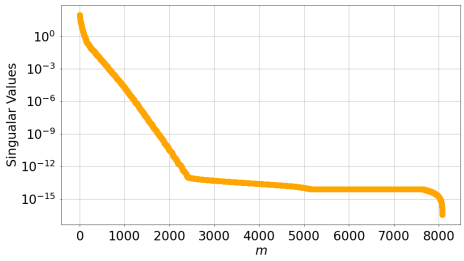


Figure 3: Test 3: Singular Values decay.

the latter, in each case 10 parameter values have been sampled to later build a test set of high-fidelity benchmark solutions ($\boldsymbol{u}_h(\boldsymbol{\mu}_{test})$) used to evaluate and compare the different approaches through Equation (11).
Only the two most remarcable tests, namely Test 3 and Test 4 are here reported.

5.1. Test 3: Pure Advection

Let us consider the parameterized two-dimensional unsteady and linear Pure Advection equation, the circular transport of a Gaussian perturbation with period 1 [s]:

$$\begin{cases} \dfrac{\partial u}{\partial t} + \nabla \cdot (\boldsymbol{b} u) = 0 & \text{in } \Omega \times I, \\ u = u_{ex} & \text{on } \Gamma_D, \\ u|_{t=0} = u_{ex}(x, y, 0) & \text{in } \Omega, \end{cases} \quad (12)$$

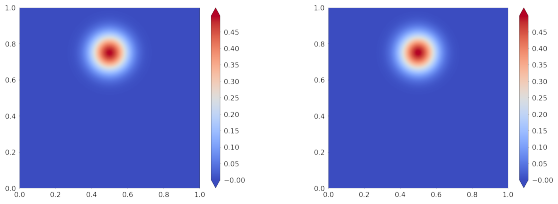
where $\Omega = (0,1)^2$, $I = (0,1)$ and:

$$\begin{aligned} \boldsymbol{b}(x,y) &= [2\pi(y-0.5), 2\pi(0.5-x)]^T, \\ u_{ex}(x,y,t) &= 0.5e^{\left(-\frac{(x-x_c)^2+(y-y_c)^2}{2\sigma^2}\right)}, \\ x_c(x,y,t) &= 0.5 + \frac{\sin(2\pi t)}{4}, \\ y_c(x,y,t) &= 0.5 + \frac{\cos(2\pi t)}{4}. \end{aligned} \quad (13)$$

The single σ^2 parameter space is given by $\boldsymbol{M} = [5e-04, 5e-03]$.
This problem is of particular interest because it shows a strong misalignment in the snapshot matrix $\boldsymbol{A}$, leading to a slow decay of its Singular Values (see Figure (3)).

5.2. PINNs-FOM

A comparison between the high-fidelity FEM solution and the PINNs-FOM one can be appreciated in Figure (4). The parameter value chosen to conduct the comparison is $\sigma^2 = 5e-03$, the



(a) (b)

Figure 4: Test 3: Comparison of the PINNs-FOM solution with the FEM solution. (4a) the FEM solution $\boldsymbol{u}_h$, (4b) the PINNs-FOM solution $\hat{\boldsymbol{u}}$

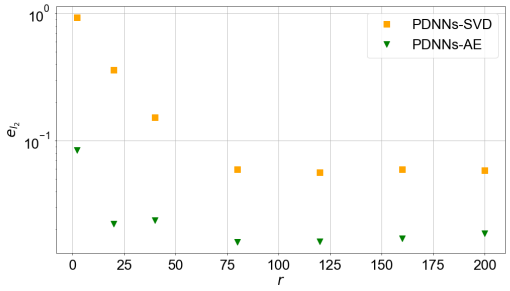


Figure 5: Test 3: L_2 relative error e_{l_2} vs reduced trial manifold dimension r: comparison between the ROM approaches.

one that leads to the largest gaussian, and the time chosen was the final time, $t = 1.0$ [s].
As it can be noticed, the two solutions are qualitatively identical.

5.3. PDNNs-SVD & PDNNs-Autoencoder

Due to data misalignment, problem (12) is a challenging task for linear ROMs. By performing the SVD on the snapshots matrix $\boldsymbol{A}$ it is found that 133 modes are needed in order to capture the 99.99% of the system energy.
The average L_2 relative error between the two different approaches and the FEM solutions, as a function of the dimension r of the corresponding reduced trial manifold is presented in Figure (5). Not surprisingly, Figure (5) shows the superiority of the PDNNs-Autoencoder approach over the PDNNs-SVD one in presence of data misalignment in the snapshot matrix $\boldsymbol{A}$ for the whole range of the reduced trial manifold dimension r considered.

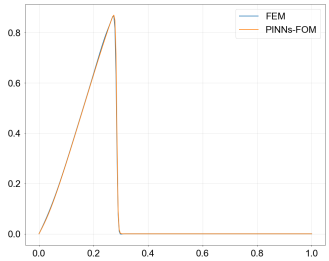


Figure 6: Test 4: Comparison between the PINNs-FOM and the FEM solution for $\mu = 1e-03$ and $t = 0.2$.

6. Test 4: Burgers Equation

Let us consider the parameterized mono-dimensional unsteady and non-linear Burgers equation defined as:

$$\begin{cases} \dfrac{\partial u}{\partial t} + u\dfrac{\partial u}{\partial x} - \mu\dfrac{\partial^2 u}{\partial x^2} = 0 & \text{in } \Omega \times I \\ u = 0 & \text{on } \Gamma_D, \\ u|_{t=0} = u_0 & \text{in } \Omega, \end{cases} \tag{14}$$

where $\Omega = (0, 1)$, $I = (0, 2)$ and:

$$u_0 = \begin{cases} 0.5(1 - cos(8\pi x)), & \text{for } x < 0.25; \\ 0, & \text{otherwise,} \end{cases} \tag{15}$$

The single parameter μ space is given by $\boldsymbol{M} = [1e-03, 1e-01]$.

6.1. PINNs-FOM

A comparison between the high-fidelity FEM solution and the PINNs-FOM one can be appreciated in Figure (6). The parameter value chosen to conduct the comparison is $\mu = 1e-03$, the one that gives the sharpest wave front, and the time chosen was t = 0.2 [s], when the wave peak is still high in magnitude.
As it can be seen in Figure (6), the accuracy of the proposed method is very high. The two solutions overlaps almost perfectly and there's no evidence of unphysical artificial artifacts.

6.2. PDNNs-SVD & PDNNs-Autoencoder

Due to the non-linear time-varying nature of the problem at hand, by performing the SVD on the

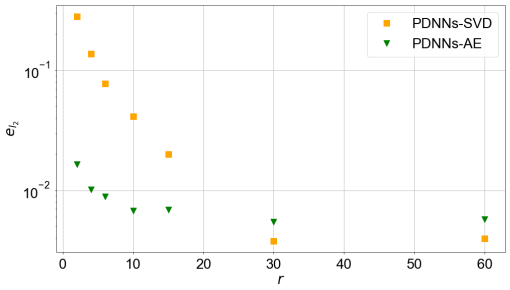


Figure 7: Test 4: L_2 relative error e_{l_2} vs reduced trial manifold dimension r: comparison between the ROM approaches.

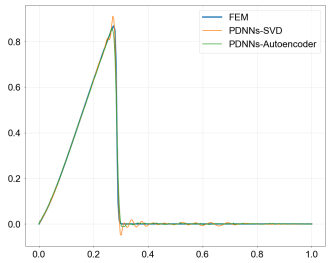


Figure 8: Test 4: Comparison of the ROM approaches with the FEM solution for $\mu = 1e-03$ and $t = 0.2$.

snapshots matrix $\boldsymbol{A}$ it has been found that 27 modes are required to capture the 99.99% of the energy of this simple system.
The average L_2 relative error between the two different approaches and the FEM solutions, as a function of the dimension r of the corresponding reduced trial manifold is presented in Figure (7). The strong dimensionality reduction offered by the PDNNs-Autoencoder approach has been confirmed also in this case: for $r < 10$ the performances are one order of magnitude better than the ones characterizing the PDNNs-SVD approach. As r increases, the increase in accuracy of the autoencoder model rapidly saturates, until the output of the linear ROM becomes the most accurate.
Let us now consider the same time-parameter instance of the PINNs-FOM example, i.e, $\mu = 1e-03$ and $t = 0.2$[s]. The plot of the solutions obtained via the two approaches is compared with the high fidelity one in Figure (8).

6.3. Results summary

PINNs-FOM	PDNNs-Ae	PDNNs-SVD	POD-G-NN
1.32e-03	1.16e-02	8.60e-03	5.55e-03
3.08e-02	1.99e-03	3.03e-05	**2.14e-05**
1.20e-02	1.60e-02	5.62e-02	-
1.57e-02	5.46e-03	**3.79e-03**	-

Table 1: Overall comparison between the investigated methods. The ROMs techniques values has been chosen case by case as the ones which led to achieve the best result.

7. Conclusions

In the present work two new promising strategies to solve parameterized PDEs while allowing at the same time a fast online prediction have been investigated and compared with more traditional SVD-based methods.
Concerning the PINNs-FOM method, the results it achieved are very satisfactory. In fact, it turned out to be the most accurate method on 2 out of 4 tests, outperforming the other strategies, as well as being the easiest to implement. It suffered more on the tests which involved the development of a thin boundary layer (Advection Diffusion, here not reported) and of a moving wave front (6).
An interesting way to improve the method could be to generate and add some high-fidelity snapshots as additional boundary conditions to match. While it is likely that this strategy will lead to better performances, its drawback consists of an additional offline cost given by the need of solving the FOM system via a traditional numerical solver at least for few instances of the parameters.
Regarding instead the PDNNs-Autoencoder approach, it has been proved that its dimensionality reduction capabilities are higher than a traditional SVD-based method, thanks to its nonlinear framework. Moreover, as depicted in Figure(8), in presence of sharp wave fronts, the approximation delivered is much more realistic than the PDNNs-SVD oscillatory one. However, this strategy is not drawbacks-free too. First of all, it must be noticed that the hyperparameter tuning process of the autoencoder architecture and its training procedure is very time consuming. Moreover, in all the test cases it can be

noticed that as the latent space representation dimension increases, the enhancements in the solution accuracy rapidly saturates, as opposed to the SVD-based approaches. We believe that there is still room from improvements in the autoencoder architecture choice.
In conclusion it can be said that this two approaches, despite being totally different from a conceptual point of view, represent both a great alternative to the traditional ROMs. They both have their drawbacks but, as mentioned above, with the mentioned future improvements their potential for possible industrial applications is huge.

References

[1] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[3] Hans Petter Langtangen and Anders Logg. *Solving PDEs in Python*. Springer, 2017.

[4] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[5] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

[6] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.