Journal of Sound and Vibration 512 (2021) 116285

Contents lists available at ScienceDirect

Journal of Sound and Vibration

journal homepage: www.elsevier.com/locate/jsvi

Predicting the propagation of acoustic waves using deep convolutional neural networks

Antonio Alguacil [a,b,*], Michaël Bauerheim [b,a], Marc C. Jacob [c], Stéphane Moreau [a]

[a] Université de Sherbrooke, Sherbrooke, Quebéc, J1K2R1, Canada
[b] Institut Supérieur de l'Aéronautique et de l'Espace, ISAE-SUPAERO, 31400, Toulouse, France
[c] Université de Lyon, École Centrale de Lyon, INSA Lyon, Université Claude Bernard Lyon 1, CNRS, LMFA, F-69134, Écully, France

ARTICLE INFO

Keywords:
Aeroacoustic propagation
Data-driven methods
Convolutional neural networks

ABSTRACT

A novel approach for numerically propagating acoustic waves in two-dimensional quiescent media has been developed through a fully convolutional multi-scale neural network following a spatio-temporal auto-regressive strategy. This data-driven method managed to produce accurate results for long simulation times with a database of Lattice Boltzmann temporal simulations of propagating Gaussian Pulses, even in the case of initial conditions unseen during training time, such as the plane wave configuration or the two initial Gaussian pulses of opposed amplitudes. Two different choices of optimization objectives are compared, resulting in an improved prediction accuracy when adding the spatial gradient difference error to the traditional mean squared error loss function. Further accuracy gains are observed when performing an *a posteriori* correction on the neural network prediction based on the conservation of acoustic energy, indicating the benefit of including physical information in data-driven methods. Finally, the method is shown to be capable of relaxing the classical time-step constraint of LBM and to provide parallel properties suitable for accelerating significantly acoustic propagation simulations.

1. Introduction

The prediction of noise generated by aero-acoustic sources has been approached in the last 50 years with a large range of numerical and analytical techniques. Most recent numerical methods encompass direct computational aero-acoustics (CAA) or hybrid CAA coupled with acoustic analogies [1]. Direct CAA solves both source and acoustic propagation, which leads to high accuracy but also to an extreme CPU cost, thus limiting the approach to academic and benchmark cases. [2–4]. Hybrid methods, however, separate the computation of the hydrodynamic fluctuations in the source regions from far field one, where acoustic fluctuations are propagated up to the observer's position. Source regions are either modeled or calculated through high-order CFD whereas far field radiation calculations derive from less expensive methods, such as Lighthill's acoustic analogy and its variants (Ffowcs Williams–Hawkings (FW–H) equation [5]), or from the resolution of the Linearized Euler Equations (LEE). The main difficulty of hybrid methods is the coupling of the source computation with the acoustic propagation computation [6]. Another difficulty arises when acoustic waves must be propagated in complex geometries or mean flows, such as ones encountered in turbomachinery [3,7]. A recent work from Pérez Arroyo et al. [8] showed a first-time Large Eddy Simulation coupled with FW–H and Goldstein [9] analogies for the noise prediction of the full fan stage of the NASA SDT benchmark. One inherent limitation is the use of expensive high-order numerical schemes to ensure low dissipation and dispersion properties. Therefore, these methods remain computationally expensive

* Correspondence to: Département Aérodynamique, Énergétique et Propulsion, ISAE-SUPAER0, BP 54032, F-31055 Toulouse Cedex 4, France.
E-mail addresses: antonio.alguacil-cabrerizo@isae-supaero.fr (A. Alguacil), michael.bauerheim@isae-supaero.fr (M. Bauerheim).

https://doi.org/10.1016/j.jsv.2021.116285
Received 5 January 2021; Received in revised form 5 May 2021; Accepted 10 June 2021
Available online 17 June 2021
0022-460X/© 2021 Elsevier Ltd. All rights reserved.

in particular in aircraft design phases where multiple model evaluations are needed for an optimum to be found. Surrogate models, where the computational cost is significantly smaller than CAA, is of vital importance for the present needs of industry. Therefore, this paper focuses on an alternative to high-order numerical schemes for the propagation of acoustic waves.

Data-driven methods have been increasingly used as surrogate models in fluid mechanics-related problems, as reviewed by Brunton et al. [10]. The objective of such techniques is to produce quick predictions, to control flows, or to optimize designs, without resorting to the full resolution of the fluid mechanics first principles in high-dimensional spaces. They rely on extracting the dominant statistical flow features from a database of representative examples (either simulated or experimental data). Traditionally, data-driven models have relied on some form of dimensionality reduction of high-fidelity data coupled with a time propagator: such reduced-order models (ROM) are based for example on the energy-based proper orthogonal decomposition (POD) [11] or the frequency-based dynamic mode decomposition (DMD) [12]. A general review on such data-driven modal analysis methods can be found in Taira et al. [13]. These techniques encode the statistics of coherent flow structures into modes, reducing the necessary degrees of freedom required to describe the flow. While they have been primarily used for pure flow analysis [14,15], POD and DMD are also employed to build surrogate models of dynamical systems. For instance, Tissot et al. [16] coupled a DMD on experimental data with a time propagator, creating a ROM predicting the dynamics of a cylinder wake. POD can also be combined with time propagators to create ROM predictors [17] or controllers [18]. However, data-dimensionality reductions techniques rely on strong underlying assumptions that can hinder the prediction of dynamics. An example is shown in Balajewicz et al. [17] for turbulence modeling where the POD algorithm must be fine-tuned to correctly account for the smallest turbulent scales, which are naturally not well represented with POD, since the kinetic energy is dominated by large eddies. Thus, a strong domain knowledge is required to achieve this fine tuning. Furthermore, strong assumptions on the linearity of the underlying dynamics (used in POD and DMD) can also be a source of discrepancies. These can be potentially overcome through a change of coordinates, using for instance a Koopman operator [19]. However, finding these coordinates transform is far from trivial as discussed by Brunton et al. [10], which calls for a more systematic method to build ROMs of time-varying linear and non-linear problems. Modern machine learning techniques, such as neural networks (NN), are able to efficiently model the spatio-temporal dynamics such as those found in aeroacoustic problems. In fact, neural networks can leverage their power as universal function approximators [20] to approach any underlying dynamics present in a given dataset, linear or non-linear. They can work either on reduced-state data descriptors (*e.g.* Proper Orthogonal Decomposition – POD – modes) or on full state representations (*e.g.* velocity, vorticity fields) to predict both steady and unsteady phenomena with no *a priori* knowledge on the physics. This flexibility may facilitate the creation of fast and accurate surrogate models.

An early example of modeling partial differential equations by neural networks can be found in Lagaris et al. [21]. However, it is only with the advent of modern-day deep neural networks that such techniques have become readily available. Deep neural networks [22,23] have been successfully used in image recognition tasks since 2012 when for the first time a convolutional neural network won the ILSVRC competition [24], by classifying (*i.e.* assigning labels to images) 1.2 million images of the ImageNet dataset. This achievement has been possible thanks to the combined use of large databases, hardware accelerators (graphical processing units — GPUs) and efficient optimization algorithms. Convolutional neural networks (CNN) are of particular interest as they show an ability to learn spatial statistical correlations from structured data, such as images or CFD-simulated flow fields. They have been applied to a wide variety of aerospace physics-related problems, like surrogate modeling of 2D RANS simulations to predict steady flow fields [25,26]. A non-linear mapping between inputs (boundary conditions) and outputs (pressure and velocity fields) is performed, through the offline supervised training on a database of RANS simulations. CNNs have also been employed for resolving a Poisson equation to enforce a divergence-free velocity in incompressible flow solvers [27,28]. Other examples include shape optimization through an inverse mapping of pressure coefficients to airfoil geometries [29,30]. Fukami et al. [31] applied CNNs to perform super-resolution of under-resolved turbulent fields. Note also that besides CNNs, other types of neural networks exist such as fully-connected networks (FCN) or recurrent neural networks (RNN). Details about the different types of architecture can be found in Goodfellow et al. [32]. Such models have also been used in fluid-mechanics related applications, sometimes in combination with them. For example, a type of RNN called Long Short-Term Memory (LSTM) [33] has been coupled with CNNs to model the complete space–time evolution of flow fields [34,35]. FCNs have also been employed in several contexts, such as improving the performance of RANS models [36], PDE modeling through the use of Physical-Informed Neural Networks (PINNs) [37–39] or for learning efficient data-driven numerical discretizations schemes [40]. These networks rely on non-linear operations to learn correlations from data, in order to predict a new output given an optimization objective. As opposed to traditional approaches where the explicit resolution of equations is needed, neural networks build an implicit understanding of physics from data observation and the choice of an objective functional to minimize (called the loss function). This approach can greatly accelerate calculations, while maintaining acceptable accuracy levels.

In the aeroacoustics field, a recent work by Tenney et al. [41] used several FCN networks to predict the fluctuating pressure signal at some azimuthal and axial positions around a jet flow, using as input data from nearby sensors. Such a type of network was also employed by Sack and Åbom [42] to perform the decomposition of acoustic modes inside ducts, which allows the learned model to account for complex flow effects such as refraction, convection or viscous dissipation. However, to the authors knowledge, the use of neural networks, in particular CNNs, has not been reported for the full-state spatio-temporal propagation of acoustic waves, as confirmed by the recent review of Bianco et al. [43]. Yet, it is well known that CNN has a high ability to capture spatial coherent patterns, which is typically the case of the radiated acoustic pressure fields which exhibit a spatially coherent topology. The spatio-temporal dependence of acoustic propagation remains however a challenge for data-driven methods, due to the data causality of time-series, and the lack of theoretical background on the errors made by CNNs, in particular when applied to a recurrent task as the one proposed in the following. It suggests that a careful attention must be taken on the training strategies to build the CNN

model. Previous works show the usefulness of using Multi-Scale CNNs for seismic wave propagation [44], or coupled LSTM–CNN models for surface waves propagation governed by Saint-Venant equations [45,46]. Results from these works are mostly qualitative and reveal difficulties for predicting accurately wave propagation over long time predictions. With these models, errors tend to accumulate over time up to a point at which the wavefronts completely lose their coherence. A key aspect of this study is thus to evaluate the error propagated over time by the neural network, to establish strategies that limit the accumulation of errors, and finally to propose a benchmark evaluated on a database obtained from a simple acoustic test case.

In the present work, a data-driven approach is proposed for the time-propagation of two-dimensional (2D) acoustic waves by a Multi-Scale CNN, trained on acoustic wavefields generated by a Lattice-Boltzmann Method CFD code. The network is trained to generate the next time-step of the acoustic density field given a number of previous steps. Once trained, the network is employed in an auto-regressive way (using the prediction at a previous time step as a new input for the next prediction) in order to produce the full time-series prediction of propagating waves. Controlling the long-term error propagation of such an iterative method is key to obtain reliable predictions. Physics-informed strategies as used by Raissi et al. [37] tend to improve the control over this error propagation, as the predictions are constrained by the prior physical knowledge of the problem, which is embedded into the loss function or into the network architecture. However, tuning the new terms appearing in the loss functions is a non-trivial task [47] and in this work, an *a posteriori* physics-informed correction is employed. This strategy consists in training the neural network without any prior physical knowledge and adding a physics-informed correction only at testing time, during the auto-regressive phase. Such a correction consists in an energy-preserving correction (EPC) based on the conservation of acoustic energy. This correction is naturally case-dependent and must be adapted to new boundary or initial conditions. However, it may improve the re-usability of the trained neural network, thus alleviating the need to re-train the network on new configurations, which can be computationally costly. The proposed approach is benchmarked on a series of two-dimensional linear acoustic propagation cases, with no mean flow effects, inside a computational domain with reflecting boundary conditions at the four boundary walls. These particular simple linear acoustics applications could be solved using established numerical methods, such as LEE. However, the evaluation of such spatio-temporal data driven approaches for acoustics is not trivial due to the lack of previous results. Therefore, it is necessary to first evaluate the method on simplistic cases such as the "closed box" test, which facilitates the characterization of the neural network temporal behavior. The acoustic waves are trapped for infinite long times inside the computational domain thus allowing the evaluation of the approach for arbitrarily long time-series and for limited spatial resolutions. Once an accurate data-driven methodology for spatio-temporal aeroacoustic predictions is established, the presented framework could be easily extended to more compelling applications, where traditional linearized numerical propagators become more costly or even fails (*e.g.* non-linear propagation for large amplitude signals). In such contexts, the learning power of neural networks and their intrinsic non-linear behavior could provide efficient predictive tools. Typical applications could range from atmospheric propagation or complex acoustic scattering by obstacles. Furthermore, a very similar framework could also be employed for source detection algorithms by simply performing an inverse time mapping.

Therefore, the objectives of the study are the following:

(i) To assess the ability of convolutional neural networks to propagate simple acoustic sources.
(ii) To compare extensively the results with reference cases for propagation (single gaussian pulse, two opposed-amplitude Gaussian pulses, plane wave).
(iii) To study best practices for training accurately the CNN of interest. A particular attention is drawn on for the choice of the optimization criteria.
(iv) To assess these data-driven simulations by a consistent error analysis, in particular to evaluate the benefits of employing an *a posteriori* energy-preserving correction.
(v) To show potential benefits of neural network to accelerate simulations, such as relaxing time-step constraints or proposing high parallelization capabilities.

The paper is divided as follows. In Section 2 Convolutional Neural Networks are presented, along with the training process, the auto-regressive prediction strategies and the *a posteriori* energy-preserving correction formalism. Section 3 describes the Lattice Boltzmann method used for generating the dataset and validates the method in terms of numerical dissipation with respect to analytical test-cases. Section 4 shows results for the three aforementioned test-cases of acoustic propagation in a closed domain without interior obstacles, and evaluates the EPC correction. A discussion about the choice of the Neural Network time-step and the associated computational cost completes the study. Finally, conclusions are drawn in Section 5.

2. Deep convolutional neural network as wave propagator

2.1. Generalities of CNNs

A typical Convolutional Neural Network architecture consists in an input image being convolved successively by multiple filters to output feature maps [48]. Non-Linear activation layers are placed in-between convolutional layers in order to learn a non-linear mapping between input and output data. The most common example of activation layers is the rectified linear unit (ReLU) [49]. CNNs use sliding filters to learn spatially correlated features from data. First layers scan local features of the input image (for velocity, density or pressure fields, small characteristic scales) while deeper layers learn high-level representations of the input images, as convolutions scan a larger area of the image.

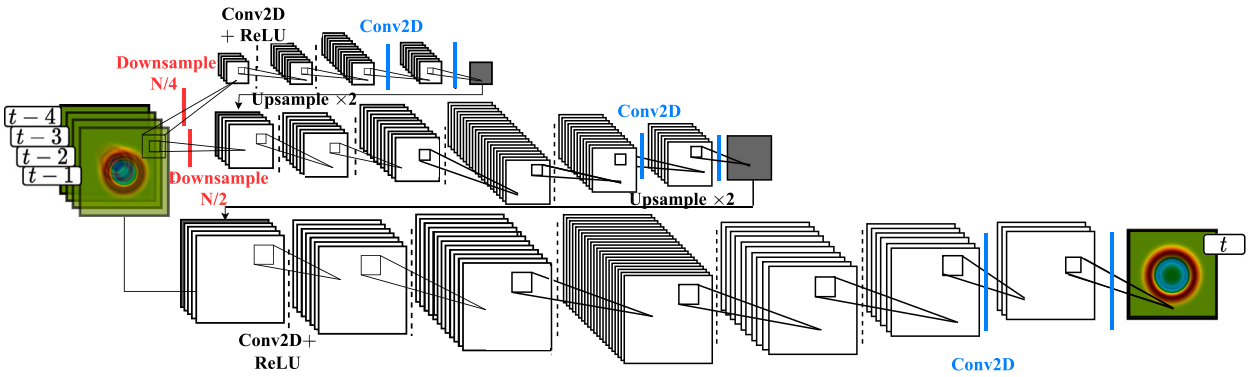

Fig. 1. MultiScale CNN with three convolutional scales at quarter, half and full-resolution of the input field. Input is composed of p consecutive frames $t-1$, $t-2$, …, $t-p$ (figure with $p=4$ as used in this work) and output is a single frame at time t and spatial resolution $N_x \times N_y = 200 \times 200$. 2D-convolution operations are performed between the different feature layers. Activation layers are rectifying linear units (ReLU), represented by a black dashed line. When no activation is used, a blue line is displayed. Depth after each convolution represents the number of filters (*i.e.* feature channels). Appendix A shows the detail for the number of feature channels and convolution operations.

Each of the convolutional filters is parametrized by learnable weights, *i.e.* modified according to an optimization objective during the training phase. These optimization criteria are targeted *via* a *loss function*. Therefore a given architecture will output different results for a same input image if trained on different data, or with a different loss function. The main challenge of data-driven techniques is thus to choose a representative dataset for the given problem, as well as an appropriate network architecture and training loss function.

2.2. Neural network training for predicting discrete time-series

In order to train a neural network for the spatio-temporal propagation of acoustic waves, a supervised strategy on a feed-forward Neural Network is chosen. Such a problem can be formalized by defining a neural network G_θ, parametrized by θ (multiplicative weights and additive biases), over a dataset composed of N pairs of input-target training samples $(X_n, Y_n)_{1 \le n \le N}$. In the particular case of spatio-temporal predictions, the pair (X_n, Y_n) is defined as:

$$\begin{cases} X_n = \{x_n^{t-p}, x_n^{t-p+1}, \dots, x_n^{t-1}\} \\ Y_n = \{x_n^t\} \end{cases} \tag{1}$$

where the x_n^t denotes some time-indexed state. A supervised training of G_θ corresponds to the minimization of a loss function $\mathcal{L}$ calculated as an error between the prediction output $G_\theta(X_n)$ and the target data Y_n.

The minimization problem can be formalized as:

$$\min_\theta \quad \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}\left(G_\theta(X_n), Y_n\right) \tag{2}$$

A mini-batch stochastic gradient descent algorithm is used for minimizing the error, even though there are no guarantees that a global optimum will be found, due to the non-convex nature of the problem. As this technique relies on calculating the gradient of the error with respect to the different layer weights, an efficient backpropagation algorithm [50] is used, where the analytical gradients of every layer output with respect to their weights are calculated at the forward pass (going from input to output). These gradients are then used in a backward pass for updating all weights of the network, so that the cost function is minimized. Such operations are performed automatically by the Pytorch open-source library [51], implementing automatic differentiation and optimization algorithms. This framework is used in the present study.

2.3. Convolutional multi-scale architecture

G_θ is modeled using a Multi-Scale Convolutional Neural Network approach, as defined by Mathieu et al. [52]. The original Multi-Scale CNN architecture was employed to predict the next video frame given 4 concatenated previous frames sampled with equal time-spacing as input. Multiple sequential ConvNets at different spatial resolutions are recursively used to output the prediction.

Reproducing Mathieu's development for completeness, (dropping the n subscript of Eq. (2) for clarity) let X and Y be an input-target tuple from the dataset, with spatial resolution $N_x \times N_y$. The objective is to define a function G_θ such that $G_\theta(X)$ approximates Y, *i.e.* $\hat{Y} = G_\theta(X)$ where the hat notation denotes an approximated solution. Let $s_1, \dots, s_{N_{scales}}$ be the input field sizes for the different network scales, X_k, Y_k the downscaled field of X and Y. For example, if a three-scales MultiScale architecture is chosen then $s_1 = N_x/4 \times N_y/4$, $s_2 = N_x/2 \times N_y/2$ and $s_3 = N_x \times N_y$. u_k is the interpolating function (bi-linear) to go from one resolution to

another. Let G_k be the sequential convolutional network that learns a prediction $\hat{Y}_k$ from X_k and a coarse guess denoted $\hat{Y}_{k-1}$. G_k makes predictions of size s_k and is defined recursively as:

$$\hat{Y}_k = G_k\left(X_k, u_k\left(\hat{Y}_{k-1}\right)\right) \tag{3}$$

The first ConvNet takes the downsampled (quarter scale) input, performs convolutions and applies activation functions while maintaining the same resolution. The output of this first bank is then upsampled to a half-scale and concatenated to the downsampled input field (to the same half-scale), which is then processed by the second ConvNet. Finally, this operation is repeated for the full scale bank and a final output is produced.

Training the MultiScale CNN for the spatio-temporal prediction of acoustic density fields, and following the notation of Section 2.2, the acoustic density ρ' is used as the state variable, $x^t = \rho'(t)$, and $p = 4$, *i.e.* 4 consecutive frames are used as input. Such a procedure has been sketched in Fig. 1. Details about the complete neural network architecture is given in Appendix A.

This type of network has been sparsely used in the deep learning regression literature compared with other architectures such as U-Net [53]. The latter has been for example used by Thuerey et al. [25] for RANS predictions or by Lapeyre et al. [54] for sub-grid scale modeling in turbulent combustion flows. Recent works have also used a Multi-Scale architecture on a range of fluid-mechanics related topics [55–58]. The main advantage of the multi-scale approach is that it separates the problem into simpler tasks: coarser resolution scales tend to filter high wave numbers but are able to learn long-range space dependence whereas full-scale convolutions are able to focus on high wavenumber information. Lee and You [58] presented the capabilities of multi-scale CNNs in fluid dynamics predictions. They demonstrated that such a CNN is able to transport and integrate the multi-scale wave number information from the input fields into the output field, through the successive convolutional layers and scales.

2.4. Loss functions

For a set of input and output data, the multi-scale neural network generates predictions that minimize the following loss function:

$$\mathcal{L} = \frac{1}{N_x}\frac{1}{N_y}\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\left\{\lambda_{l2}\mathcal{L}_{l2} + \lambda_{gdl}\mathcal{L}_{gdl}\right\}\left(\hat{Y}_{i,j}, Y_{i,j}\right) \tag{4}$$

where $\hat{Y}_{i,j}$ and $Y_{i,j}$ correspond to the network prediction at the output of the G_3 scale and its associated target field for a cell defined by its indexes (i, j). Both fields have a resolution of $N_x \times N_y$.

$\mathcal{L}_{l2}$ minimizes the L2-norm mean-square error (MSE) of the output with respect to the target fields (here corresponding to density fields). For a cell defined by indexes (i, j):

$$\mathcal{L}_{l2}\left(\hat{Y}_{i,j}, Y_{i,j}\right) = \left[\hat{Y}_{i,j} - Y_{i,j}\right]^2 \tag{5}$$

$\mathcal{L}_{gdl}$ (called Gradient Difference Loss (GDL) and introduced by Mathieu et al. [52]) minimizes the L2-norm mean-square error of both x- and y-components of the spatial gradients of density fields, discretized with forward first-order finite differences. For a cell defined by indexes (i, j):

$$\begin{aligned}\mathcal{L}_{gdl}\left(\hat{Y}_{i,j}, Y_{i,j}\right) = &\left[\left(\hat{Y}_{i+1,j} - \hat{Y}_{i,j}\right) - \left(Y_{i+1,j} - Y_{i,j}\right)\right]^2 + \\ &\left[\left(\hat{Y}_{i,j+1} - \hat{Y}_{i,j}\right) - \left(Y_{i,j+1} - Y_{i,j}\right)\right]^2\end{aligned} \tag{6}$$

The mean-square error functions remain the classical choice for loss in regression problems. This choice supposes that data is drawn from a Gaussian distribution. Therefore if a multi-modal data distribution is employed, results will tend towards the weighted average between the probability modes. The gradient loss penalizes this feature of the mean-square error loss and forces the output distribution to follow the multi-modal behavior of the target data, and as a result, sharpens the predicted fields.

The choice of weighting parameters λ_{l2} and λ_{gdl} will be carried out in the result section, along with a comparison between a model trained only with the mean-square error loss and one combining both MSE and GDL.

2.5. Auto-regressive prediction strategy

The neural network training is stopped once the loss function converges to a steady value. The network is then employed as an acoustic wave propagator, producing arbitrarily long spatio-temporal series through an auto-regressive strategy. For each new prediction, the input state uses the previous prediction. This procedure is illustrated in algorithm 1, with an example given for the particular case of $p = 4$ inputs. Because of the choice of a multi-step input state, for each new prediction the p input frames are cyclically permuted. The last input frame is sent to the first-to-last index in the input state $\hat{X}^{t-1}$, the first-to-last to the second-to-last, etc. Finally, the first index frame is sent to the last index of the new input state. This operation is denoted by $\sigma_{(p,p-1,\ldots,2,1)}(\hat{X}^{t-1})$. Then the time frame at the last index $\hat{x}^{t-p}$ is replaced by the last prediction $\hat{x}^t$. Such new state $\hat{X}^{t-1}$ is used as the input for the next prediction.

Algorithm 1 Auto-regressive strategy

Train: G_θ
Initialize: $\hat{X}^{-1} = \{x^{-p}, x^{-p+1}, ..., x^{-1}\}$
$= \{x^{-4}, x^{-3}, x^{-2}, x^{-1}\}_{p=4}$
while $t \in [0, t_{max} - 1]$ **do**
  Initial state: $\hat{X}^{t-1} = \{\hat{x}^{t-p}, \hat{x}^{t-p+1}, ..., \hat{x}^{t-1}\}$
  Predict: $\hat{Y}^t = G_\theta(\hat{X}^{t-1}) = \{\hat{x}^t\}$
  Permute: $\sigma_{(p,p-1,...,2,1)}(\hat{X}^{t-1}) = \{\hat{x}^{t-p+1}, \hat{x}^{t-p+2}, ..., \hat{x}^{t-1}, \hat{x}^{t-p}\}$
  $= \{\hat{x}^{t-3}, \hat{x}^{t-2}, \hat{x}^{t-1}, \hat{x}^{t-4}\}_{p=4}$
  Replace: $\hat{x}^{t-p} \leftarrow \hat{x}^t$
  $\hat{X}^{t-1} = \{\hat{x}^{t-p+1}, ..., \hat{x}^{t-1}, \hat{x}^t\}$
  $= \{\hat{x}^{t-3}, \hat{x}^{t-2}, \hat{x}^{t-1}, \hat{x}^t\}_{p=4}$
end while

2.6. Energy preserving correction (EPC)

The strategy presented previously is based on the supervised training of a CNN. The loss function (Eq. (4)) employed for such a process does not make any assumption on the underlying physics present on the data. Recent works by Raissi et al. [37] on Physical-Informed Neural Networks (PINN) show that adding some conservation equation term to the loss function improves the predictive accuracy of the model by enforcing well-known physical constraints during the training process.

In the present work, a similar strategy is employed in order to improve the accuracy of the auto-regressive algorithm. A physics-informed energy preserving correction (EPC) is designed to enforce the conservation of the acoustic energy throughout the neural network recursive prediction strategy. However, the chosen approach differs from Raissi's one, as this correction is not integrated as an additional training loss term. In contrast, the EPC is applied only at test time in order to correct the predictions made by an already trained neural network. Such a choice is made in order to improve the re-usability of the neural network for different problem configurations: the EPC is case-dependent and must be adapted for every type of boundary condition or mean flow features. Thus, we hope to train the neural network only once using standard loss functions (and avoiding training instabilities as shown in Wang et al. [47] when using many loss terms), and adapting the *a posteriori* correction for each particular case. This would yield a smaller computational cost at training time and a greater flexibility and re-usability of neural networks, at the expense of designing *a posteriori* corrections for each new test-case.

For the particular test cases studied in this work, the energy preserving correction is based on the observation that the acoustic propagation takes place in a linear regime. The acoustic energy follows the conservation law in integral form:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_\Omega E dx + \int_{\partial\Omega} \mathbf{I}\cdot\mathbf{n}\mathrm{d}\sigma = -\int_\Omega D\mathrm{d}x \tag{7}$$

where E represents the acoustic energy density and $\mathbf{I}$ the acoustic intensity:

$$\begin{aligned} E &= \frac{p'^2}{2\rho_0 c_0^2} + \frac{\rho_0 v'^2}{2} \\ \mathbf{I} &= p'\mathbf{v}'. \end{aligned} \tag{8}$$

Note that such relationships are only valid for a uniform isentropic field at rest (at Mach $M = 0$). The dissipation D is considered nil and the LBM viscosity is low in order to preserve this hypothesis. For the studied configuration presented in Sections 3 and 4 , the acoustic propagation takes place in a closed domain with reflecting hard walls. This implies that the mean energy flux across the closed domain containing no sources is zero. Since velocity fluctuations are related to pressure fluctuations *via* the Euler equation and the density fluctuations are proportional to pressure fluctuations, the total energy conservation yields the total density conservation.

Assuming a positive uniform drift (hypothesis validated in Section 4.2), let ϵ (≥ 0) be a correction for the density field $\hat{\rho}'(t)$ applied after each autoregressive prediction, such that:

$$\int_\Omega (\hat{\rho}'(t) - \epsilon)^2 \mathrm{d}x = \int_\Omega \rho'(t=0)^2 \mathrm{d}x \tag{9}$$

By symmetry of the problem:

$$\int_\Omega (\hat{\rho}'(t))^2 \mathrm{d}x = \int_\Omega (\rho'(t=0) + \epsilon)^2 \mathrm{d}x \tag{10}$$

Define

$$\begin{aligned} \hat{P} &= \frac{\int_\Omega \hat{\rho}'(t)\mathrm{d}x}{\int_\Omega \mathrm{d}x} \\ P_0 &= \frac{\int_\Omega \rho'(t=0)\mathrm{d}x}{\int_\Omega \mathrm{d}x}. \end{aligned} \tag{11}$$

Combining Eqs. (10) and (9) yields:

$$\int \epsilon^2 dx - 2\int \rho'(t)\epsilon dx = -\int \epsilon^2 dx - 2\int \rho'(t=0)dx \tag{12}$$

Since the drift is supposed uniform, ϵ can move outside of the integrals, leading to the second order equation:

$$\epsilon^2 - \epsilon\hat{P} + \epsilon P_0 = 0 \tag{13}$$

This equation has the null solution $\epsilon = 0$, which could correspond to no correction of the predicted field, and the following non-trivial solution is found for the correction:

$$\epsilon = P_0 - \hat{P} \tag{14}$$

Eq. (14) reveals that the intuitive correction of the drift obtained by removing the mean difference between the current prediction and the initial target field, corresponds actually to the energy conservation in time. Note also that the proposed correction only holds for a particular case of boundary conditions (reflecting wall) and a quiescent mean flow. The change in boundary condition (e.g. non-reflecting boundary conditions) would result in a non-zero acoustic energy flux through the domain boundaries, and thus the proposed EPC would no longer hold.

The EPC correction is integrated in the auto-regressive algorithm as shown in algorithm 2, by simply adding the correcting term ϵ to the predicted acoustic density field at each recursive step.

Algorithm 2 Auto-regressive strategy with energy preserving correction

Train: G_θ
Initialize: $\hat{X}^{-1} = \{x^{-p}, x^{-p+1}, ..., x^{-1}\}$
while $t \in [0, t_{max} - 1]$ **do**
  Initial state: $\hat{X}^{t-1} = \{\hat{x}^{t-p}, \hat{x}^{t-p+1}, ..., \hat{x}^{t-1}\}$
  Predict: $\hat{Y}^t = G_\theta(\hat{X}^{t-1}) = \{\hat{x}^t\}$
  EPC: $\hat{x}^t \leftarrow \hat{x}^t + \epsilon$
  Permute: $\sigma_{(p,p-1,...,2,1)}(\hat{X}^{t-1}) = \{\hat{x}^{t-p+1}, \hat{x}^{t-p+2}, ..., \hat{x}^{t-1}, \hat{x}^{t-p}\}$
  Replace: $\hat{x}^{t-p} \leftarrow \hat{x}^t$
  $\hat{X}^{t-1} = \{\hat{x}^{t-p+1}, ..., \hat{x}^{t-1}, \hat{x}^t\}$
end while

3. Dataset generation

The training dataset used for this study is composed of 2D domains with hard reflecting acoustic walls, no obstacles and Gaussian density pulses as initial conditions. Such a simple academic configuration is preferred to more complex, yet more interesting cases, such as the acoustic propagation in complex mean flows. The main reason for this is the lack of accurate results regarding the capacity of Neural Networks when performing wave propagation. Thus, the approach consists in benchmarking this data-driven method with simple configurations and exposing the potential caveats of the spatio-temporal CNN predictive approach. These problems could then be identified and mitigated, before moving to more complex cases.

The Lattice-Boltzmann Method (LBM) Palabos code [59] is used to generate such data. These methods achieve second-order spatial accuracy, yet resulting in a small numerical dissipation similar to a 6th order optimized Navier–Stokes schemes [60], making LBM highly suitable for aeroacoustic predictions [61]. However classical BGK–LBM collision models can produce high frequency instabilities [62] when viscosity is low. As a consequence, a recursive and regularized BGK model (rrBGK–LBM) [63] is applied to maintain code stability with low numerical dissipation. Details about the Lattice-Boltzmann are provided in appendix A $D2Q9$ lattice discretization is chosen, representing the possible 9 discrete velocities for each two-dimensional (2D) lattice node.

3.1. Numerical setup: Propagation of Gaussian pulses in a closed domain

The first dataset consists of 500 2D-simulations, with hard walls imposed on the four domain boundaries (Fig. 2). Reflecting walls are modeled as classical half-way bounce-back nodes [64]. Density fields are initialized with a random number (varying between one and four) of Gaussian pulses located at random domain locations. The total density of a Gaussian pulse is defined as

$$\rho(x, y, t = 0) = \rho_0 + \varepsilon \exp\left[-\frac{\ln 2}{b^2}\left((x - c_x)^2 + (y - c_y)^2\right)\right] \tag{15}$$

where ε is the pulse amplitude, b is the pulse half-width, c_x and c_y are the x- and y-coordinates of the pulse center. All variables are set in dimensionless units, also called "lattice units" (see Krüger et al. [65]). The reference lattice density is chosen as $\rho_0 = 1$. The computational domain *physical* dimension is chosen as D (see Fig. 3a) , each direction being discretized with $N_{nodes} = 200$ nodes. The parameters ε and b/D are fixed and equal to 0.001 and 0.06 respectively, such that 12 lattice nodes discretize the initial pulse half-width (this choice is justified in Section 3.2). As the background density $\rho_0 = 1$, the fluctuating density ρ' is chosen such that $\rho_0 \gg \rho'$ in order to avoid non-linear effects. Viscosity (in lattice units) is set to $\nu = 0.0$. Zero-viscosity values tend to generate

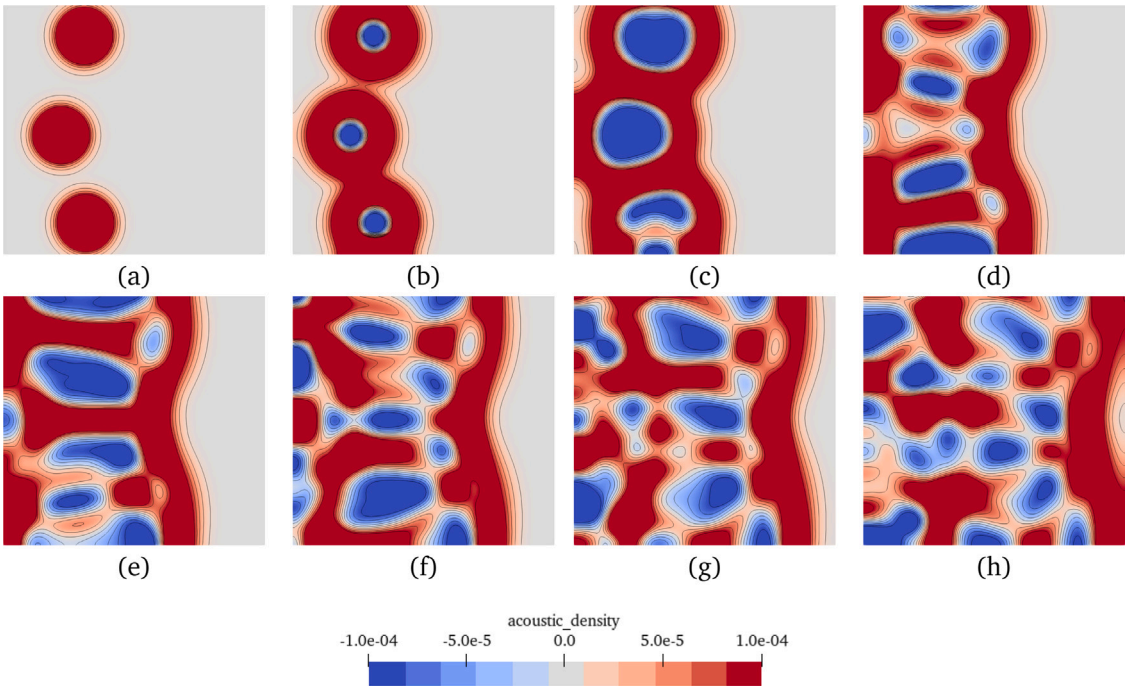

Fig. 2. Example of one simulation from the dataset. Initial conditions are composed of density Gaussian pulses located at random positions. Each initial condition ($it = 0$) is stepped 231 LBM iterations in total. Acoustic density fields are saved every 3 LBM iterations. Shown fields correspond to LBM iteration (a) $0\Delta t$, (b) $30\Delta t$, (c) $60\Delta t$, (d) $90\Delta t$, (e) $120\Delta t$, (f) $150\Delta t$, (g) $180\Delta t$ and (h) $210\Delta t$.

numerical instabilities when employing the traditional BGK collision model. However, the rrGBK model is unconditionally stable at $M = 0$ mean flow conditions [63].

The only relevant time-scale comes here from the acoustic propagation of waves, set to the speed of sound, where the relation between *physical* and *lattice* units is given by

$$(c_s)_{lb} = (c_s)_{phys} \frac{\Delta t}{\Delta x}. \tag{16}$$

Therefore the choice of the LBM time-step is imposed by both the discretization of space and velocity-space, the former imposing Δx and the latter, $(c_s)_{lb}$. Furthermore, the sound speed in the $D2Q9$ lattice $(c_s)_{lb}$ acts as a Courant number in this context, with a fixed value of $1/\sqrt{3} \simeq 0.57$, which is a limitation arising from the explicit time-discretization scheme employed by the LB method. A normalized time may be defined as

$$\tau = N_{iter}(c_s)_{lb}/N_{nodes} \tag{17}$$

where N_{iter} is the number of LB iterations and N_{nodes} is the number of lattice nodes in one direction of the computational domain. This time τ corresponds to the propagation time of an acoustic wave from one boundary to the other. Each training simulation is stopped at $\tau_{\text{training}} = 0.67$ and density fields are recorded at time-steps of $\Delta\tau = 0.0087$ (*i.e.* each 3 LB iterations). The latter time-step is the one used by the CNN.

This particular choice of time step (*i.e.* greater than the one imposed by the LB method) suggests that the CNN could achieve some speed-ups compared with the LBM by performing low-error predictions at larger timesteps. The neural network is in fact not bounded by the explicit time-discretization scheme limit (*e.g.* Courant number), whereas the LB time-step size is directly linked to the spatial discretization. Thus, a NN could theoretically *learn* to overcome this limitations through a training process on data with a lower frequency sampling than the LBM. This paper tries to provide some insights into the physics related to the previous statements. In particular, all the network trainings are performed with an *under-sampling* strategy by setting $\Delta t_{NN} = 3\Delta t_{LBM}$. The objective is to assess whether the CNN is able to accurately predict the acoustic propagation when using such conditions.

In practice, the computed LBM fields are packed into groups of 4 + 1 frames (input+target) for the CNN training. Snapshots from a complete LBM run are shown in Fig. 2.

3.2. Validation of the LBM code

In order to demonstrate the dissipation behavior of the rrBGK–LBM solver, a benchmark simulation is performed by comparing the free-field propagation of a single density gaussian pulse with the analytical solution, given by a zero-order Bessel function

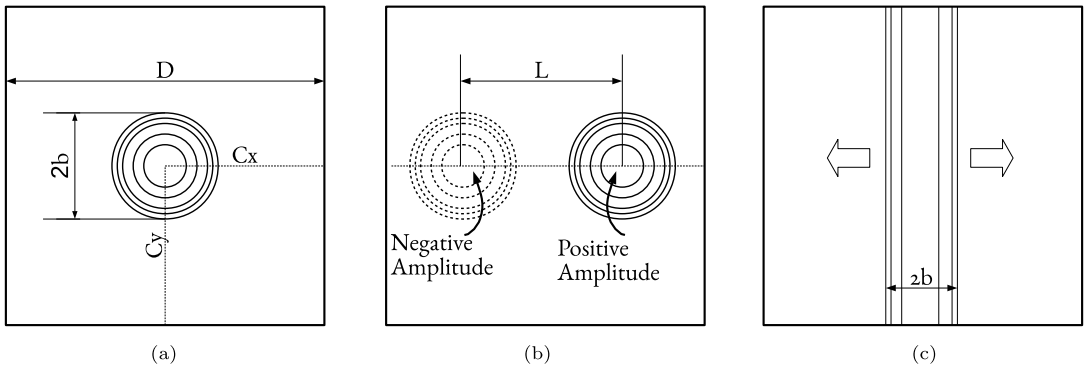

Fig. 3. Schematic of the three test cases. Domain size is $D \times D$. (a) propagation of single Gaussian Pulse of half-width b, (b) Two Pulses with opposed initial amplitudes and (c) Plane wave with Gaussian profile propagating in x-direction.

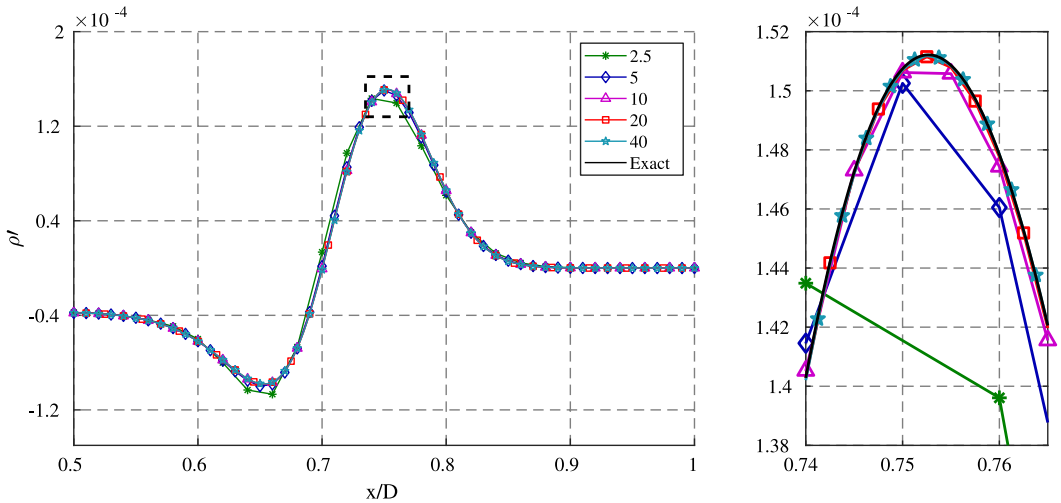

Fig. 4. Propagation of Gaussian pulse with initial parameters $\epsilon = 0.001$ and $b/D = 0.05$. Comparison of the cross profiles of density fluctuations $\rho\prime$ for different grid resolutions of at slice $y/D = 0.5$, $x/D > 0.5$ (axis of symmetry) and dimensionless time $\tau = 0.26$. The legend indicates the number of lattice points per initial Gaussian pulse half-width $b/\Delta x$ and the analytical solution is plotted in black. The figure to the right shows a zoom around the pulse maximum, marked by a dashed rectangle.

J_0 [66]:

$$\rho(x, y, t) = \rho_0 + \frac{\epsilon}{2\alpha} \int_0^\infty \exp(-\xi^2/(4\alpha)) \cos(\xi t) J_0(\xi\eta)\xi d\xi \tag{18}$$

where $\alpha = \log 2/b^2$ and $\eta = \sqrt{x^2 + y^2}$ represents the radial coordinate in absence of mean flow. A schematic of the problem setup is shown in Fig. 3a.

Fig. 4 shows a slice of the density fluctuations in the line $y/D = 0.5$ at time $\tau = 0.26$ before the pulse impinges on any wall. Comparison is made between the analytical solution and various simulations performed with various resolutions $b/\Delta x = 2.5$, 5, 10, 20 and 40. Results show that at least 10 points per half-width are necessary to capture the propagation of the pulse accurately with a small numerical dissipation error. The 12-point-per-half-width criterion, chosen in Section 3.1, is therefore deemed sufficient to capture accurately the acoustic wave propagation.

Furthermore, Fig. 5 shows two types of metrics to evaluate the numerical order of the spatial scheme. First, the L_2-norm of the difference between the analytical and the numerical solutions is shown (Fig. 5a), averaged over the whole slice shown in Fig. 4. The error follows the expected second-order trend. The order of convergence of the spatial scheme is further estimated by a Richardson extrapolation [67], which uses several solutions at different levels of grid refinement. Let p be the order of convergence of the numerical scheme and $f(\Delta x)$ the numerical solution at a grid resolution Δx, then

$$p = \log\left(\frac{f(\Delta x/4) - f(\Delta x/2)}{f(\Delta x/2) - f(\Delta x)}\right) / \log(2). \tag{19}$$

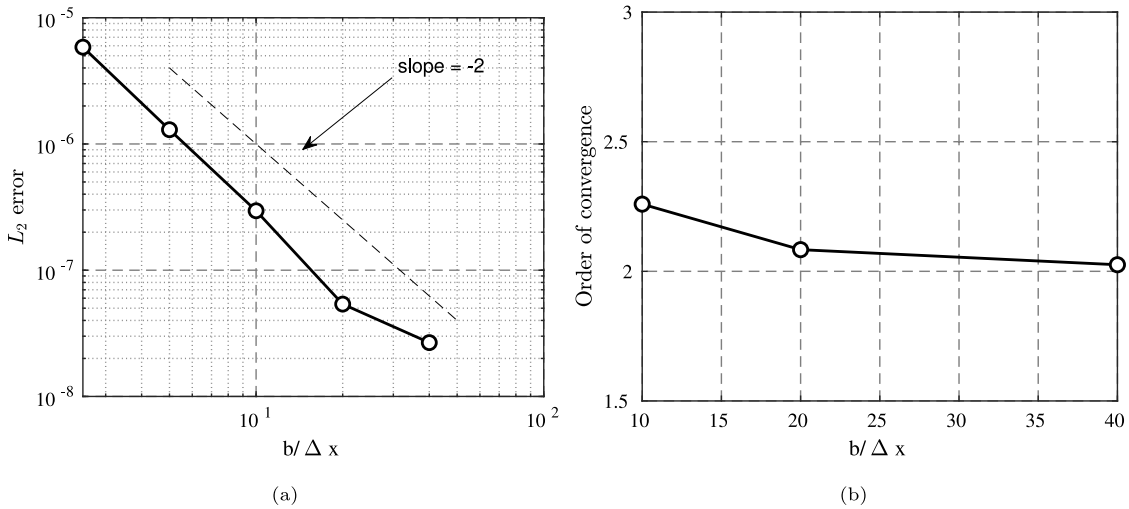

Fig. 5. Estimation of order of convergence of the rrBGK–LBM spatial scheme: (a) evolution of the *rms* error with the lattice resolution compared with the formal spatial scheme order (second order) and (b) order of convergence of the spatial scheme (Richardson extrapolation).

Fig. 5b shows that the order of the numerical method converges to the expected second-order value as the grid is further refined, thus confirming the dissipation properties of the rrBGK–LBM scheme.

3.3. Test cases for the trained neural network

The above validation case is also used for the validation of the Neural Network auto-regressive predictions. Henceforth this single Gaussian pulse case is referred to as "**Test A**". While the training dataset contains very similar initial conditions as this first case, it constitutes a baseline test for the neural network capability. Furthermore, as the training was only performed for one-step-ahead predictions, this test case evaluates the accuracy of the auto-regressive strategy presented in Section 2.5.

In addition to this validation case, two more cases are used for evaluating the network performance:

- **Test B**: Double Gaussian pulse, with opposed initial amplitudes. One pulse is located at $\left(c_x, c_y\right)_1 = (D/2 - L/2, D/2)$ with amplitude $\varepsilon_1 = -\varepsilon$, the other at $\left(c_x, c_y\right)_2 = (D/2 + L/2, D/2)$ with amplitude $\varepsilon_2 = \varepsilon$. A schematic is shown in Fig. 3b. This test aims at verifying the network generalization ability, *i.e.* the capacity to extrapolate a solution when the input falls outside the training space. This test remains nonetheless close to the training data distribution.
- **Test C**: One-dimensional Gaussian plane wave, propagating in the x-direction, as shown in Fig. 3c. This configuration is a challenging generalization case since it is very different from the combination of radially expanding waves addressed by the training. Indeed, Sorteberg et al. [45] showed that "plane wave" cases were difficult to predict when training their network with only cylindrical pulses.

4. Results

This section studies the performance of the Multi-Scale neural network trained with the dataset described in Section 3.1, by comparing the resulting predictions for test cases A and B with the corresponding LBM predictions, taken as "Target" data (as usually defined by the Machine Learning community). For case C, treated in Section 4.4, the target solution employed for comparison is the analytical solution for a 1D propagating Gaussian pulse.

4.1. Training parameters

Two types of training strategies are taken into consideration. Training T_1 is carried out by setting the loss function as a mean-square error such that $\lambda_{l2} = 1$ and $\lambda_{gdl} = 0$. Training T_2 sets $\lambda_{l2} = 0.02$ and $\lambda_{gdl} = 0.98$, combining mean-square error and gradient loss functions. The ratio $\lambda_{l2}/\lambda_{gdl}$ is set to obtain similar values of the two parts of the loss function, so that both mean and spatial gradient errors contribute equally to the parameter update during the optimization of the neural network. Recent works [47] have demonstrated the importance of balancing the contributions from the different loss terms in order to increase the accuracy and stability of the neural network training process.

400 simulations from the dataset are employed for training with 77 density fields per simulation at consecutive timesteps. Training samples are used to calculate losses in order to optimize the network by modifying its weights and biases. The remaining 100 simulations are used for validation purposes: no model optimization is performed with this data. The validation is only used

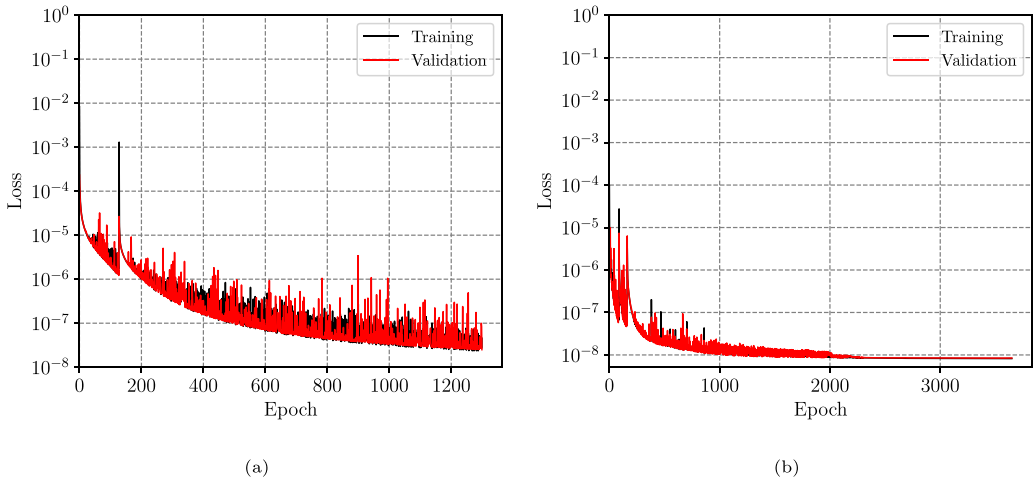

Fig. 6. Evolution of total losses for training samples and validation samples for (a) Training T_1 ($\lambda_{l2} = 1$, $\lambda_{gdl} = 0$) and (b) Training T_2 ($\lambda_{l2} = 0.98$ and $\lambda_{gdl} = 0.02$).

to ensure that a similar accuracy is achieved by the neural network on data unseen during the training phase. During the training, the dataset is processed in batches to produce predictions (forward pass) followed by the update of network weight parameters θ (backward pass to minimize the error). A complete cycle through all dataset samples is named an epoch. This process is repeated until the loss is converged. Training samples are shuffled at each epoch in order to increase data variability. Data points (input and target fields) are also rotated randomly at four right angles in order to perform a second type of data augmentation.

The Adam stochastic optimization algorithm [68] is used, with an initial learning rate set to 1×10^{-4} and a 2% reduction each time the loss value stagnates for more than 10 epochs.

Fig. 6 shows the evolution of both training and validation dataset errors during the optimization run. Network weights are optimized until convergence. For T_2, convergence on both components of the loss is reached. These two trainings provide two optimized neural networks that can be now tested on the three benchmarks proposed in Section 3.3. Although the error is lower for the MSE optimization, the GDL training is expected to be more robust for configurations exhibiting sudden amplitude changes such as wave reflections against hard walls.

For the auto-regressive tests presented in the next sections, the two baseline networks T_1 and T_2 are employed using both algorithm 1 (no EPC applied) and algorithm 2 (EPC applied). For the latter case, results using such a correction are presented as T_1^C and T_2^C.

4.2. Auto-regressive test A: Gaussian pulse

Resulting fields for test case A, corresponding to a centered Gaussian pulse left propagating through the closed domain, are shown in Fig. 7 for both T_1 and T_2 for several dimensionless times $\tau = N_{iter}(c_s)_{lb}/N_{nodes}$. Again, note that a single timestep performed by the neural network corresponds to three LBM iterations $\Delta t_{NN} = 3\Delta t_{LBM}$. Snapshots of the density fluctuations along the line $y/D = 0.5$ are plotted for the target, T_1 and T_2 in Fig. 8 at 18 representative times of the simulation. Note also that the figure scale changes between the different rows. Good agreement is found between the reference field and predictions up to dimensionless times around $\tau \simeq 1.22$ for both networks.

The initial Gaussian pulse spread is well captured (first row in Fig. 8). Both wall-pulse interactions ($\tau \sim 0.5$) and pulse–pulse interactions ($\tau \sim 1$, after wall reflection) seem to be well predicted. Although the network was trained to predict fields one single time-step ahead from the four input data fields, the recursive prediction of outputs does not perturb the predictions significantly until late times ($\tau = 1.22$ corresponding to 141 recursive predictions). Both networks seem capable of predicting both the mean level and the spatial gradients accurately, without introducing significant numerical dissipation or dispersion.

For times greater than $\tau = 1.22$, a uniform drift in density level is observed. This drift is more significant in training T_1 than T_2. This suggests that spatial gradients continue to be accurately predicted and that this drift is mostly homogeneous. As the inputs are the outputs from previous predictions, the neural network can no longer correct itself once the inputs are already shifted. This is a typical error amplification effect encountered with iterative algorithms. Training T_2 suffers also from this mean level drift, even though the amplitude error remains smaller than in T_1. Furthermore, symmetry of the density field is lost from $\tau = 1.22$ onward, as seen in Fig. 7 (third row). This indicates that errors also tend to accumulate in the prediction of spatial gradients, even if this error remains small. The evolution of the root-mean-square (rms) error relative to the maximum spatial values of the density target ρ' and gradient of the density $\nabla\rho'$ is plotted in Fig. 9 and shown in black. These errors are defined as:

$$R(\hat{\rho}') = \sqrt{\frac{1}{N}\sum_N \left(\hat{\rho}' - \rho'\right)^2} \quad \text{and} \quad \tilde{R}(\hat{\rho}') = \frac{R(\hat{\rho}')}{\max(\rho')} \tag{20}$$

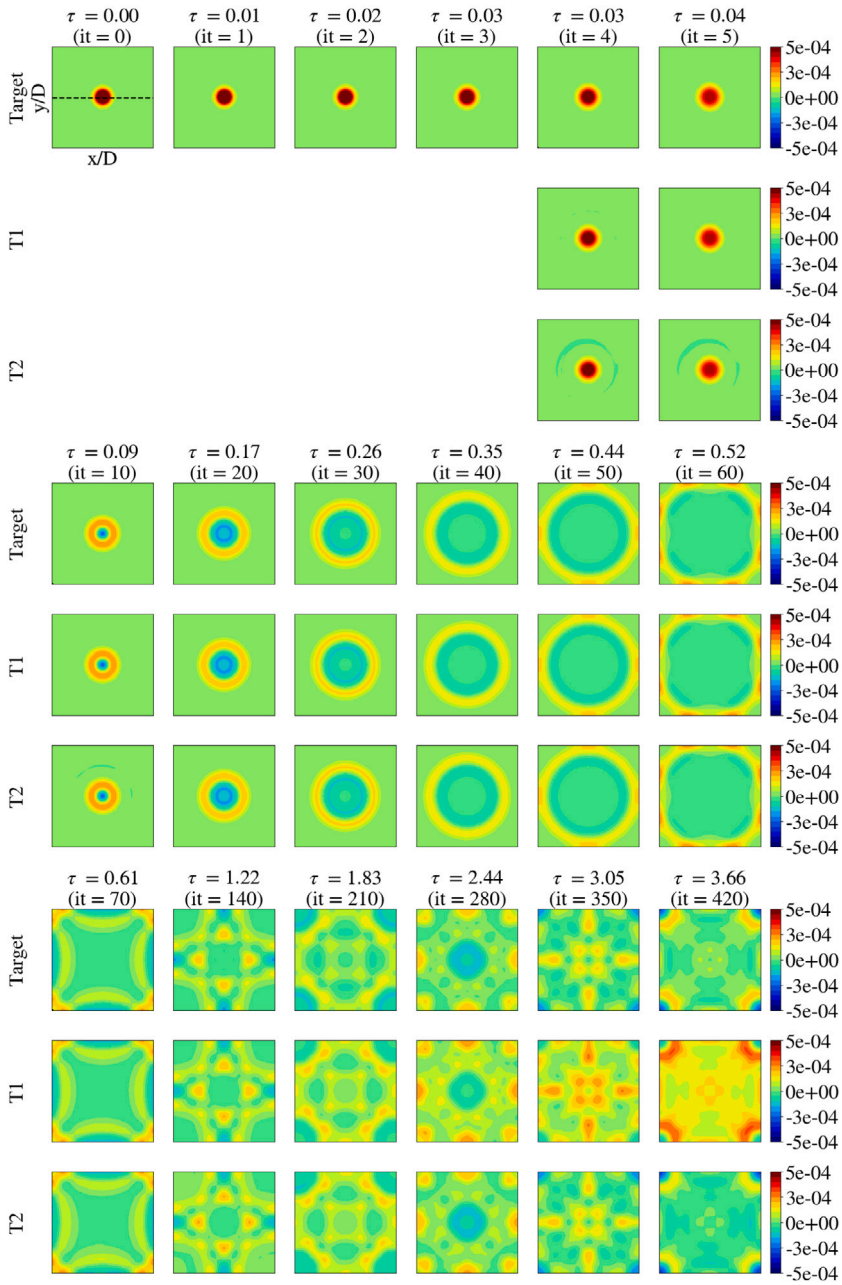

Fig. 7. Results for training T_1 and T_2, compared with LBM target data and test case A (pulse) for dimensionless time $\tau = tc_0/D$.

and:

$$R(\nabla\hat{\rho}') = \sqrt{\frac{1}{N}\sum_{N}\left(\|\nabla\hat{\rho}' - \nabla\rho'\|_2\right)^2} \quad \text{and} \quad \tilde{R}(\nabla\hat{\rho}') = \frac{R(\nabla\hat{\rho}')}{\max(\nabla\rho')} \tag{21}$$

where $\|.\|_2$ represents the L2-norm. Both networks show a similar behavior: the error starts at very low levels ($\sim 10^{-4}$) for mean-square density error, corresponding to the converged loss found during training for the validation samples. Then the error grows abruptly, until reaching 3% of the maximum density value for the MSE. The relative error remains however below 5% for times up to $\tau = 0.5$ for T_1 and $\tau = 1.0$ for T_2.

For longer times ($\tau \geq 0.5$), the relative error in the density fields keeps growing steadily up to values close to 1 in the case of T_1, *i.e.* the relative error is in the order of the signal itself. This behavior comes mostly from the uniform drift, as the errors on field gradients are one order of magnitude lower than errors on the density field.

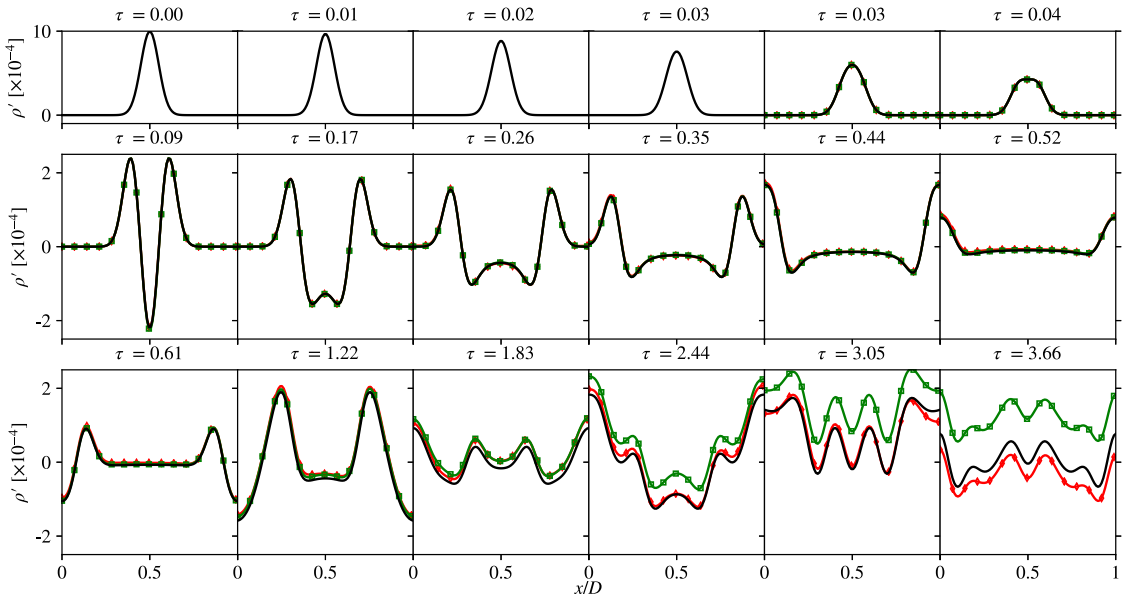

Fig. 8. Slice of density field at $y/D = 0.5$ for Test Case A (pulse). First four images represent input fields, then predictions are performed recursively. (——) Target data (LBM), (—□—) training T_1 (loss in MSE) and (—◇—) training T_2 (loss in MSE+GDL).

This uniform drift fulfills the necessary hypothesis employed in the development of the *a posteriori* energy-preserving correction presented in Section 2.6. As explained in Section 4.1, the EPC is applied to both networks T_1 and T_2 at each recursive prediction, effectively creating two new corrected networks T_1^C (*i.e.* MSE loss coupled with EPC correction) and T_2^C (*i.e.* training with MSE and gradient loss and EPC correction). The error evolution for test A is shown in Fig. 9, in red color. As expected, the evolution of the gradient error is identical with or without correction (Fig. 9b). Such behavior demonstrates that the correction has no effect on the error made by the neural network on spatial gradients, while it improves the error on the mean density level significantly (Fig. 9a). Interestingly, network T_2^C (*i.e.* using both gradient loss and MSE during training) has a greater error than T_1^C between $\tau = 0.2$ and $\tau = 1.2$, even though it remains around the 10% threshold. For longer times, T_1^C and T_2^C errors follow similar trends and levels. This seems to suggest that overall, both networks perform relatively similarly when the *a posteriori* correction is applied, and that accurate solutions are found even for times not seen during the training (only times samples for $\tau \leq 0.67$ were present in training data).

Such a behavior is clearly shown in Fig. 10, where slices of the resulting fields are plotted again at $y/D = 0.5$ for several non-dimensional times. Comparing with the results in Fig. 8, the EPC seems to avoid the long-term energy drift on both trainings T_1^C and T_2^C, obtaining a quasi-perfect fit with the LBM-calculated solution.

It can be concluded that both neural networks are able to predict the propagation of a single pulse recursively, which is a very similar case to the training ones. With EPC, the long-time error is reduced by one order of magnitude. This highlights the high capability of a physics-informed neural network to reproduce physics compared with standard data-driven only methods. Next sections will discuss the ability of the network to predict propagation of acoustic waves with initial conditions that have *not* been sampled during the training process. It will thus be checked if and how well the network is able to effectively learn the underlying Green's function for other types of initial conditions than those of the training.

4.3. Auto-regressive test B: Opposed Gaussian pulses

The second test case consists of two Gaussian pulses, with opposed initial amplitudes, propagating in a closed domain. Trainings T_1^C and T_2^C (EPC neural networks) are evaluated. Fig. 11 displays snapshots of density fields for both networks. Snapshots of the density fluctuations along the line $y/D = 0.5$ are plotted for the target T_1^C and T_2^C in Fig. 12 for the same dimensionless times as in previous figures, and Fig. 13 shows the error evolution. Network T_2^C continues to predict both evolutions accurately for times up to $\tau \sim 1.2$ or even further. It shows a clear advantage over T_1^C as mean square errors are systematically lower for networks using gradient loss during training. In fact, for T_2^C, errors remain below the 10% threshold, whereas for training T_1^C some peaks with a 30% error relative to the rms density value are found before $\tau = 3.0$. As seen in the density fields (Fig. 11), at these times wave signals exhibit complex spatial patterns, with many local extrema over short distances, thus making the gradient prediction harder for a network that has not seen those patterns during training. Both networks continue nonetheless to provide overall accurate gradient predictions, and manage to capture both pulse–pulse and pulse–wall interactions (two front-waves adding or subtracting their amplitudes during a short period of time). In fact, Fig. 13 shows bumps in the gradient error appearing periodically, which correspond to these strong interactions phases. It seems that, although the error tends to grow, the RMSE of gradients tends to

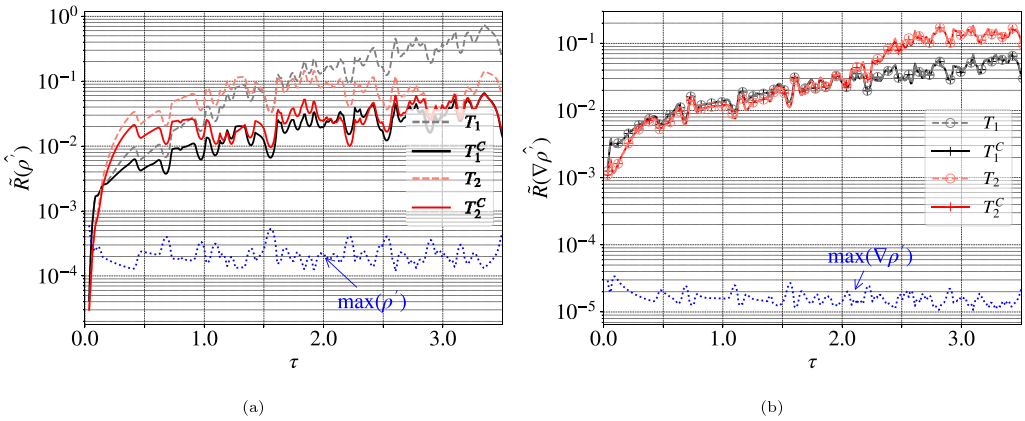

Fig. 9. Comparison of error evolution for trainings T_1, T_2, T_1^C and T_2^C for test Case A (pulse). (a) rms error of density relative to the maximum value of density $\max(\rho')$ at each time-step (in dotted blue) and (b) rms error for the sum of density gradients relative to $\max(\nabla_x\rho', \nabla_y\rho')$ at each time-step (in dotted blue).

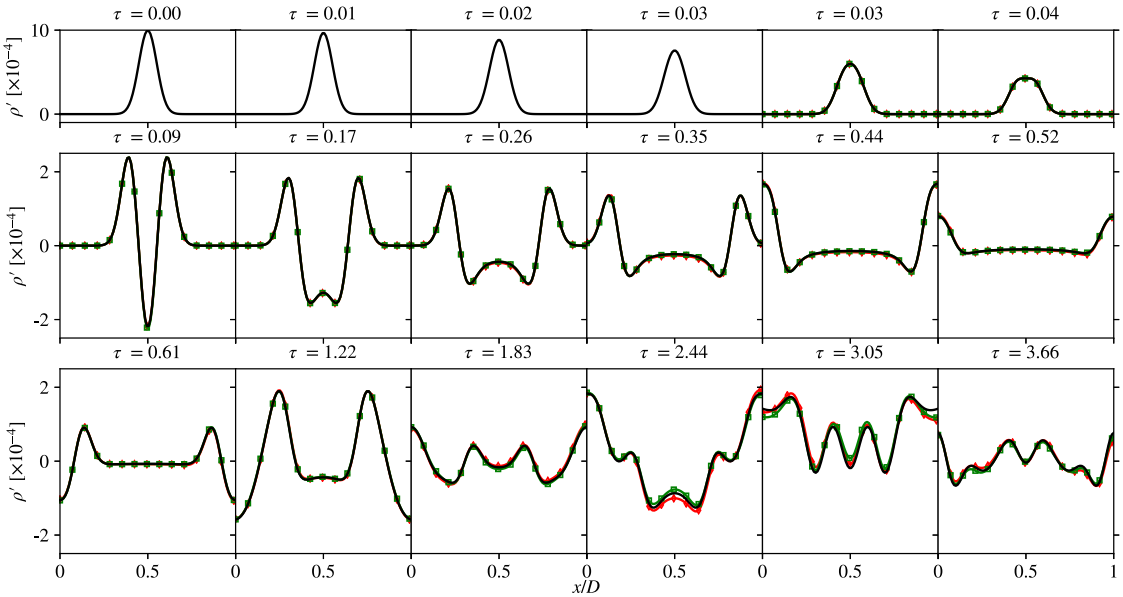

Fig. 10. Slice of density field at $y/D = 0.5$ for Test Case A (Gaussian pulse) using the Energy-Preserving Correction. The four first images represent input fields, then predictions are performed recursively. (——) Target data (LBM), (—□—) training T_1^C (loss in MSE + correction) and (—◇—) training T_2^C (loss in MSE+GDL+correction).

recover after these interactions, and grows again at the next one. This phenomenon might be attributed to the strong unsteadiness appearing during such events, which have not been frequently sampled during training.

4.4. Auto-regressive test C: Plane Gaussian pulse

A third test case is studied with both networks, with and without EPC, where initial conditions correspond to the analytical solution for the one-dimensional propagation of a Gaussian pulse, propagating in both directions along the x-direction. This solution is then extruded into the y-direction, in order to obtain 200×200 grids as for the training data and fed into the neural network. Figs. 14 and 15 show the time evolution of a data slice on the x-axis at $y/D = 0.5$ and the associated error over time respectively. This case differs from the previous ones in that no complex pattern should appear, just two plane Gaussian pulses bouncing on the walls and interacting at the domain center. Since the networks were trained exclusively with cylindrical pulses, the major challenge for the network lies in its ability to understand that the plane-wave pulse must remain straight and coherent. As already mentioned in Section 3.2, this was reported as the critical difficulty of the LSTM–CNN approach proposed by Sorteberg et al. [45] for seismic waves. Good agreement is found for times below $\tau = 0.5$ for all four cases, which correspond to the free-field initial propagation up to the first wall interaction. All wall reflections are clearly marked in the error curves by the sudden variation of the relative error.

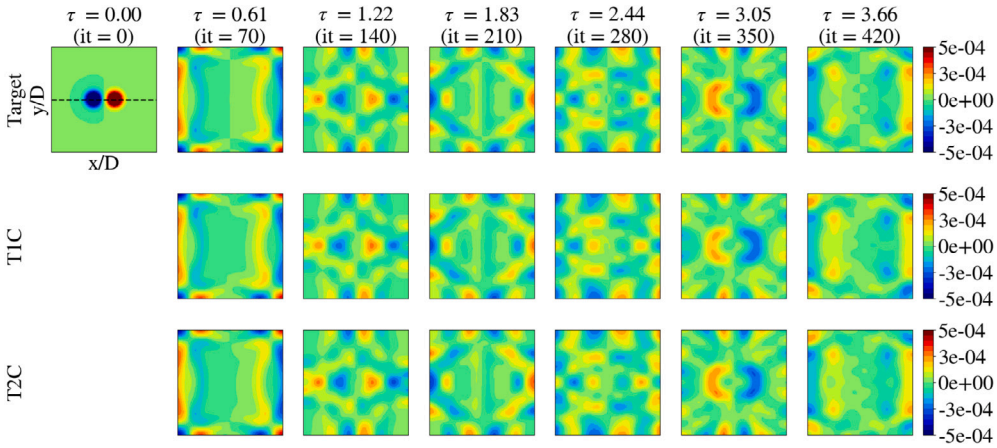

Fig. 11. Results for training T_1^C and T_2^C and test case B (two opposed pulses) for dimensionless time $\tau = tc_0/D$.

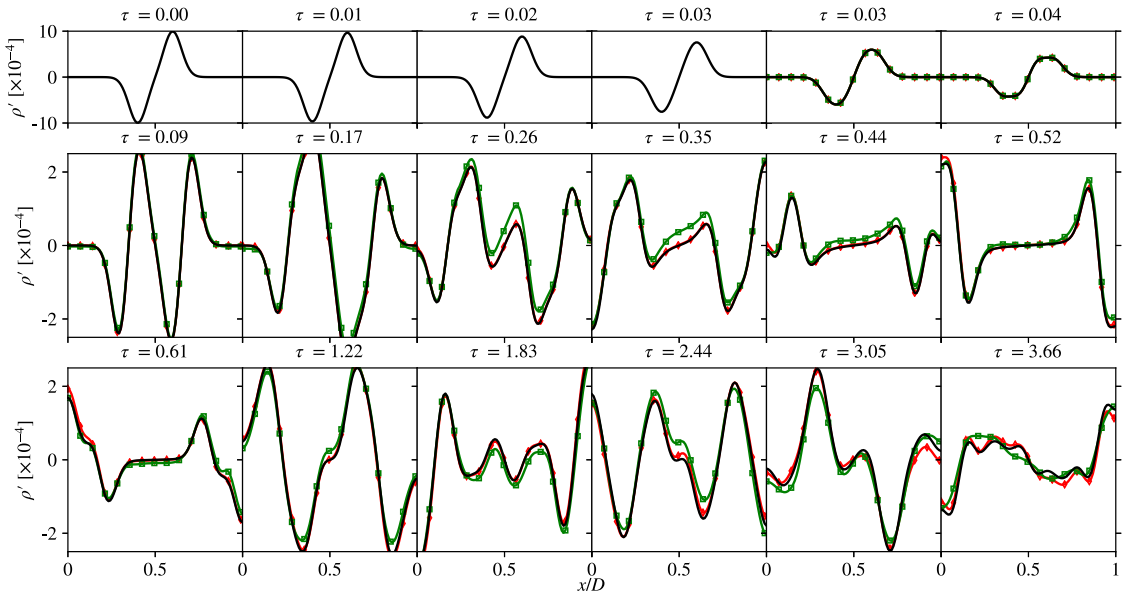

Fig. 12. Slice of density field at $y/D = 0.5$ for Test Case B (two opposed pulses) using the Energy-Preserving Correction. Four first images represent input fields, then predictions are performed recursively. (——) Target data (LBM), (—□—) training T_1^C (loss in MSE + correction) and (—◇—) training T_2^C (loss in MSE+GDL+correction).

The local error diminution is mostly due to a sharp increase of the maximum value related to wall impingement, while the absolute error maintains its value. From $\tau = 0.6$ onwards, both T_1 and T_2 perform worse that their EPC counterpart (T_1^C and T_2^C). The effect of the EPC on controlling the accuracy of the T_1 prediction is particularly visible, as the EPC allows the network to reduce the error related to the density drift by one order of magnitude. However, the EPC does not seem to completely eliminate such a drift, as T_1^C shows also a beginning of this phenomena, preceded by an increase of the gradient error (visible for example for $\tau = 1.83$). Later times show the propagation of accurate gradient predictions, while it is clear that the network tries to maintain good mean density levels. For trainings T_2, the benefit of using the EPC seems more limited, but still advantageous to the error control related to the density drift.

The Neural Network manages to maintain the straight wavefront, as shown in Fig. 16, although such dynamics were not present in the training set. This demonstrates the CNN ability to successfully extrapolate new data and to capture the underlying physics of wave propagation. It should be however noted that other important parameters were kept constant (pulse spatial resolution, lattice resolution), thus the generalization capability of the network has only been studied in the sense of changing the topology of initial conditions.

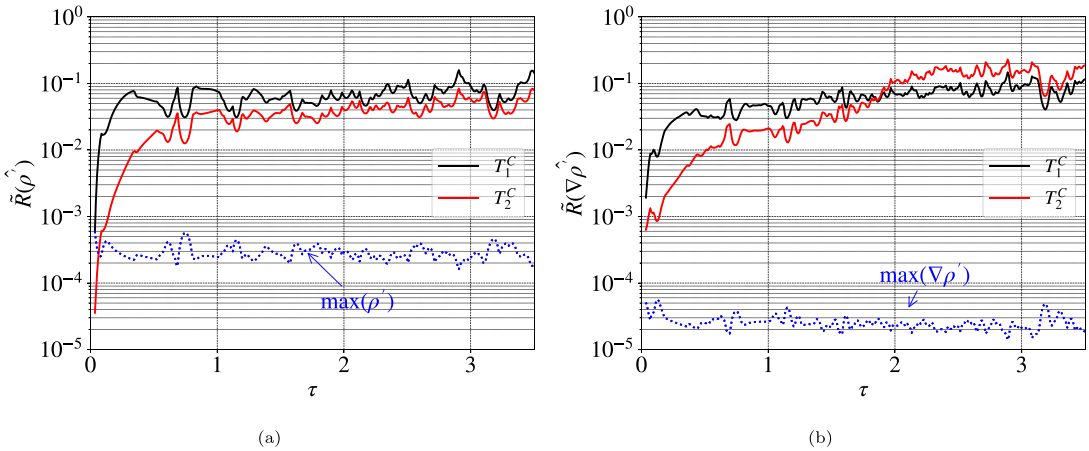

Fig. 13. Comparison of error evolution for trainings T_1^C and T_2^C for Test case B (two opposed pulses). (a) rms error of density relative to the maximum value of density $\max(\rho')$ at each time-step (in dotted blue) and (b) rms error for the sum of density gradients relative to $\max(\nabla_x\rho', \nabla_y\rho')$ at each time-step (in dotted blue).

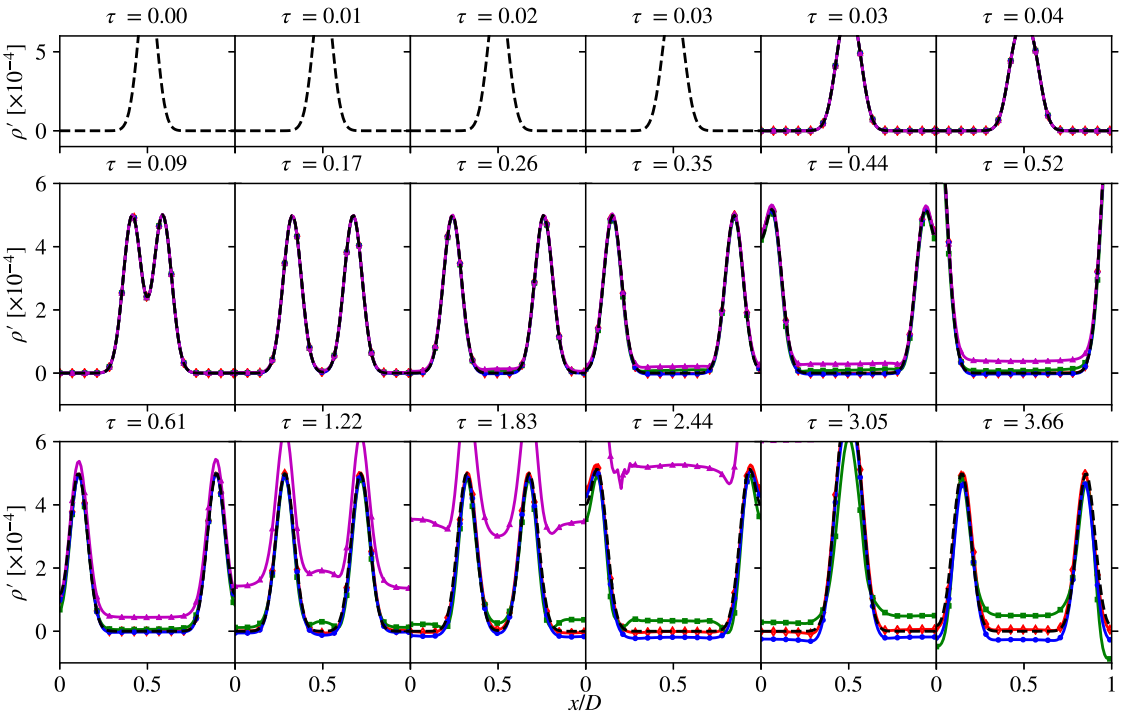

Fig. 14. Slice of density field at $y/D = 0.5$ for test case C (plane Gaussian pulse) using the Energy-Preserving Correction. First four images represent input fields, then predictions are performed recursively. (——) Target data (LBM), (—△—) training T_1 (loss in MSE), (—○—), training T_2 (loss in MSE+GDL), (—□—) training T_1^C (loss in MSE + correction) and (—◇—) training T_2^C (loss in MSE+GDL+correction).

4.5. Parametric study: influence of input time-step

In order to assess the capability of the learned model to perform acoustic predictions at larger time-steps than the LBM reference, a parametric study is performed by training the neural network at $\Delta t_{NN} = 1, 2, 4, 8, 16, 32$ and $64\Delta t_{LBM}$. Each neural network is stepped 500 iterations. Fig. 17 shows the evolution of the relative MSE error with respect to the neural network iterations and the non-dimensional time τ, for the EPC and no-EPC cases. When no EPC is employed, increasing the training time-step of the neural network results in a monotonic increase of the error with respect to the number of performed evaluations. However, because the time-step changes between the different trainings, a fixed time-horizon is reached in fewer iterations for larger values of Δt_{NN}, resulting in an overall reduced error as can be seen in Fig. 17c. This demonstrates that the main source of error in such kind of auto-regressive method is the accumulation of error at each recurrence. This observation is only valid for $\Delta t_{NN} < 8\Delta t_{LBM}$, as

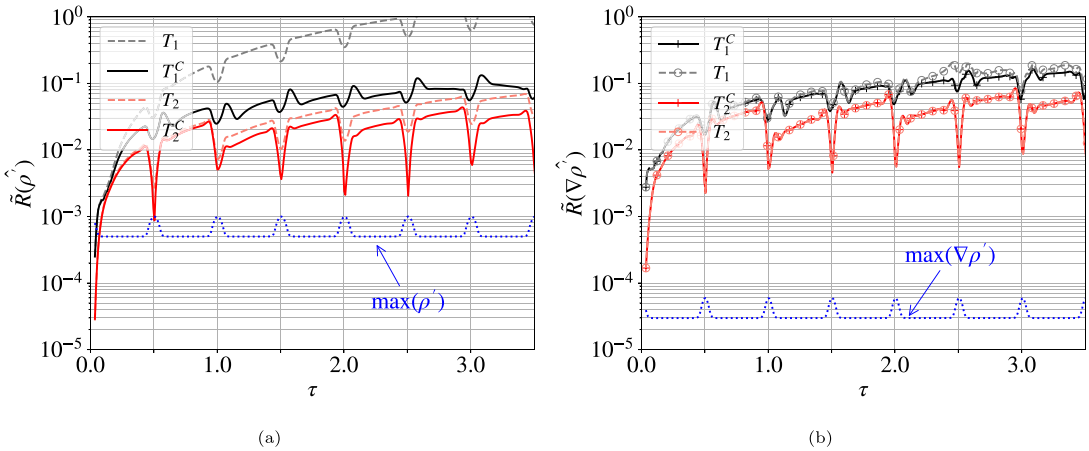

Fig. 15. Comparison of error evolution for trainings T_1, T_2, T_1^C and T_2^C for test Case C (plane wave). (a) rms error of density relative to the maximum value of density $\max(\rho')$ at each time-step (in dotted blue) and (b) rms error for the sum of density gradients relative to $\max(\nabla_x\rho', \nabla_y\rho')$ at each time-step (in dotted blue).

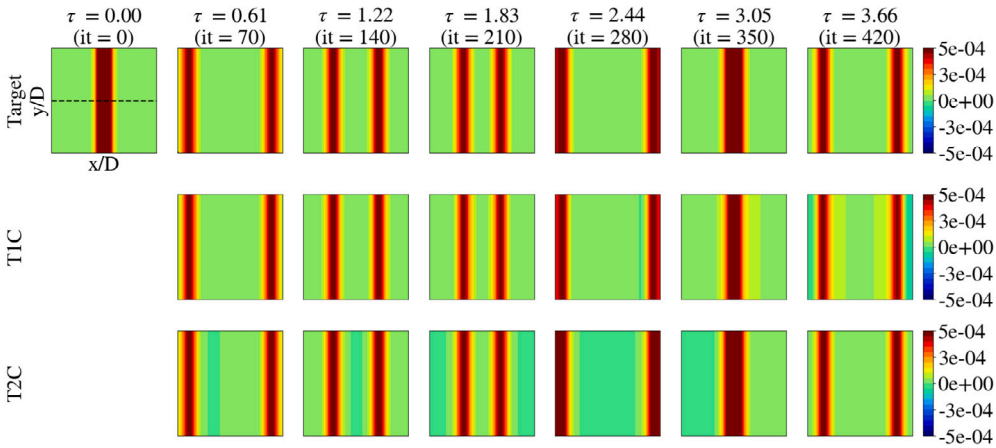

Fig. 16. Results for training T_1^C and T_2^C, test case C (plane pulse) for dimensionless time $\tau = tc_0/D$.

for larger time-steps, the prediction error diverges rapidly. This could be attributed to the increased complexity of the training task when increasing Δt_{NN}: for reduced values, the change between two consecutive density fields is small, and thus the neural network can learn a simple mapping of the time-derivative. However, for increased values of Δt_{NN}, the results suggest that there is a threshold (between $\Delta t_{NN} = 8\Delta t_{LBM}$ and $\Delta t_{NN} = 16\Delta t_{LBM}$) from which the proposed method is incapable of learning such a time-derivative, which is changing rapidly over time. Similar observations have been made by Liu et al. [69] when exploring other types of temporal evolving PDES. Future investigations will continue the research on this topic, to study whether a change in the neural network architecture or the loss function can help to better learn such complex mappings.

The use of the EPC correction (Fig. 17(b) and (d)) significantly limits the error accumulation over time. Surprisingly, the EPC benefits more to the neural networks trained on larger time-steps but does not change the aforementioned threshold for which the network learns the time-mapping. As demonstrated by Liu et al. [69], such results suggest that it is possible to combine several neural networks, each trained on a different time-step, to obtain a temporal multi-scale predictor. This strategy can alleviate even further the accumulation of error over time, as it minimizes the number of iterations required to reach a certain time-horizon. It also allows the predictions to be parallel in time, which could further decrease the computational cost of such surrogates.

4.6. Computational cost

As one of the main objectives for the developing of such surrogates is to accelerate the direct acoustic simulations, the numerical method (LBM) and the data-driven neural network computational costs are compared next. Results are shown in Table 1 for several hardware and choices of neural network hyper-parameters, namely time-step and batch size. The batch size represents the number of simulations which can be fed in parallel to the neural network. Two metrics are employed: the wall-clock time of one single

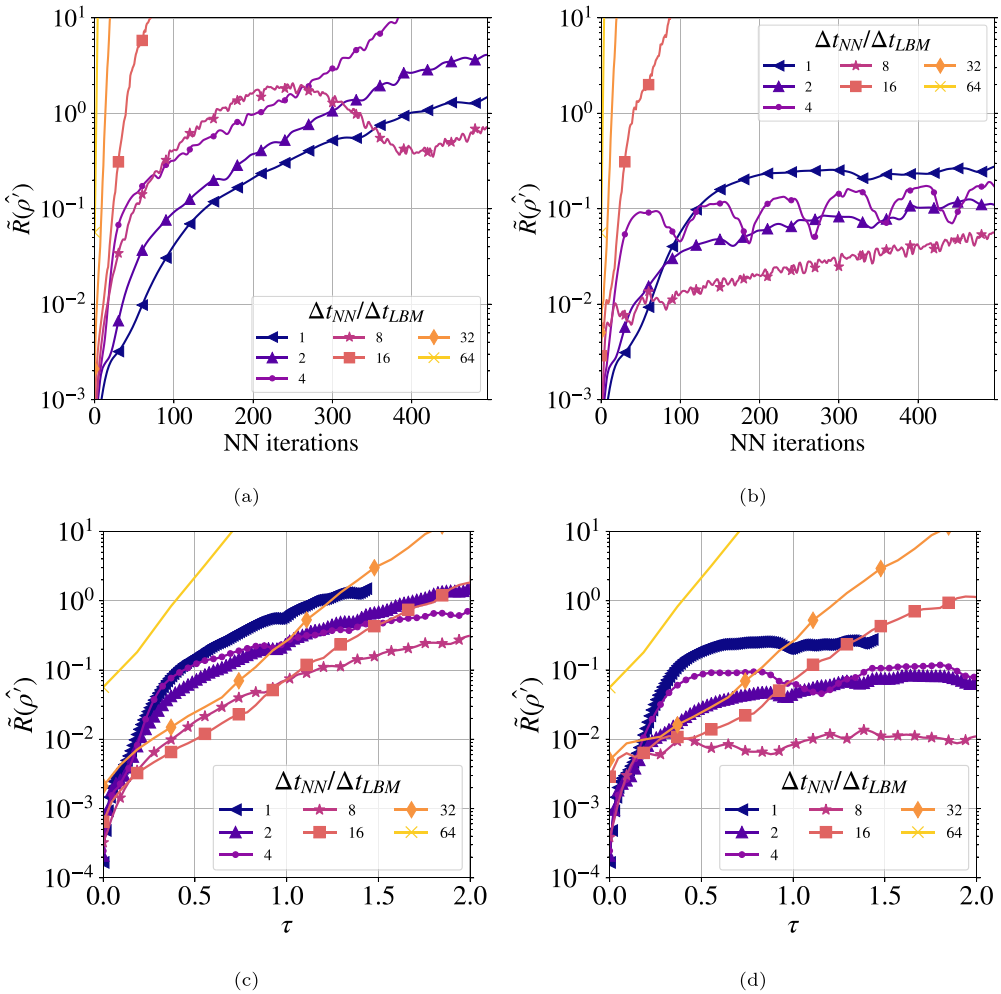

Fig. 17. Evolution of relative error for cases (a and c) T_2 and (b and d) T_2^C with respect to neural network iterations and dimensionless time $\tau = tc_0/D$, for 10 initial random Gaussian pulses, averaged for each neural network iteration, for several training time-steps Δt_{NN}.

model evaluation divided by the batch size (*i.e.* to go from X^{t-1} to X^t) and the wall-clock time necessary to reach a fixed time horizon for one particular initial condition (*i.e.* a fixed τ equal to 1). Notice that for the baseline at $\Delta t_{NN} = \Delta t_{LBM}$ and batch size bsz = 1 the direct LBM simulation outperforms the Neural Network on the wall-time per iteration metric, even on accelerated hardware (GPU). However, the neural network becomes competitive by using two complementary approaches: first, the use of GPUs allows the parallelization of simulations, and up to 256 initial conditions can be fed to the GPU memory, achieving a speed-up of 1.9 times with respect to the LBM baseline. Second, the use of the under-sampling strategy, *i.e.* the predictions at larger time-steps (*e.g.* $\Delta t_{NN} = 8\Delta t_{LBM}$), allows us to relax classical CFD constraints such as the CFL number, resulting in a acceleration of 6.9 times with respect to the reference simulation. This new perspective brought by neural network offers a high potential to accelerate the computations: fewer iterations are needed in order to reach the target time horizon (*e.g.* for $\tau = 1$ shown in Table 1). The combination of both strategies can achieve a speed-up of 15.5 times with respect to the LBM code.

5. Conclusion

A method for predicting the propagation of acoustic waves is presented in this work, based on the implementation of a Multi-Scale convolutional neural network trained on LBM-generated data. Training samples consist of variations around the classical benchmark of the propagation of 1 to 4 2D Gaussian pulses. Two types of training strategies are studied through the variation of loss functions. The neural network is optimized to perform one-step predictions, and then employed in an auto-regressive strategy to produce complete spatio-temporal series of acoustic wave propagation. An *a posteriori* energy-preserving correction (EPC) is proposed to increase the accuracy of the predictions and added to the auto-regressive algorithm. Both networks are then evaluated with initial conditions unseen during the training process, as a way to test the generalization performance. An increased accuracy is shown by the

Table 1
Comparison of the computational cost for the reference LBM code and the neural network, tested on several hardware and with different hyper-parameters. For a batch size bsz > 1, bsz simultaneous predictions are performed.

Method	Timestep $\Delta t/\Delta_{LBM}$	Batch size	Hardware	Wall-time per iteration [s]	Wall-time to τ=1 [s]	Acceleration factor (ref: LBM)
LBM	1	1	Intel Skylake 6126 (CPU)	0.0031	1.076	1.0
Neural network	1	1	Intel Skylake 6126 (CPU)	0.2521	87.825	0.012
Neural network	1	1	Nvidia V100 (GPU)	0.0036	1.249	0.861
Neural network	8	1	Nvidia V100 (GPU)	0.0036	0.156	6.897
Neural network	1	256	Nvidia V100 (GPU)	0.0016	0.555	1.925
Neural network	8	256	Nvidia V100 (GPU)	0.0016	0.069	15.594

network trained with a combination of penalizations on the mean square errors of both the density and its spatial gradients, even for the challenging plane wave case. In all cases, the EPC correction yields a significant accuracy gain, at least one order of magnitude compared to pure neural network predictions. This exemplifies the benefits of physics-informed neural networks compared with pure data-driven methods. Here, the EPC allows such an *a posteriori* correction, that is without having to retrain the neural network, which makes this correction flexible, and can be adapted depending on the case and physics to be solved. Finally, the proposed Multiscale network with EPC reveals its ability to predict an unsteady phenomenon well beyond the duration of the one-step ahead training. Moreover, the implemented neural network is able to predict flow fields at 8 times larger time-steps than the LBM reference. The developed auto-regressive model does not seem to be limited by the classical limitations of standard explicit numerical methods such as the Courant number, at least up to a certain point, opening the path to fast and efficient tools for acoustic propagation. Further investigations should provide more insights into such a phenomenon, in order to formally address the difficulty of neural networks to learn time mappings at very large time-steps. Finally, a similar framework could be employed in more compelling application cases, in particular for the complex acoustic scattering with obstacles or the non-linear wave propagation in combustion or atmospheric flows. The techniques presented in the current work provide some best practices in order to train the neural network on new acoustic databases, for the subsequent temporal propagation of waves.

CRediT authorship contribution statement

Antonio Alguacil: Conceptualization, Methodology, Software, Validation, Investigation, Writing - original draft, Formal Analysis, Visualization, Writing - review & editing. **Michaël Bauerheim:** Conceptualization, Methodology, Resources, Writing - review & editing, Supervision, Funding acquisition. **Marc C. Jacob:** Conceptualization, Methodology, Resources, Writing - review & editing, Supervision, Funding acquisition. **Stéphane Moreau:** Conceptualization, Methodology, Resources, Writing - review & editing, Supervision, Funding acquisition.

Declaration of competing interest

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

Acknowledgments

The author want to thank Wagner Gonçalves Pinto, from Institut Supérieur de l'Aéronautique et de l'Espace (ISAE) for fruitful discussions. Daniel Diaz-Salazar is acknowledged for his help in the validation of the Palabos LBM code. This work was partly supported by the French "Programme d'Investissements d'avenir" ANR-17-EURE-0005 conducted. We also acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC). The simulations were carried out using the CALMIP supercomputing facilities (project P20035) in Toulouse, France and in Compute Canada clusters.

Appendix A. Neural network architecture details

Table A.1 presents the details of the operations employed in the multi-scale neural network architecture. The three scales are denoted G_1 for the quarter resolution, G_2 for the half-resolution and G_3 for the full resolution convolutional neural networks. For G_1 and G_2, an bi-linear down-sampling operation is first performed. Then, for all three scales, a series of parametric convolution operations are performed, using the following notation: $k\#$ denotes the kernel size of the convolution operation (*e.g.* $k3$ represents a 2D kernel of size 3×3). $s\#$ is the stride (*i.e.* the jump used to apply the convolution operator), which is always equal to 1. $p\#$ denotes the amount of border padding used at each convolution operation. Padding is employed to keep the same field resolution before and after the convolution. Here, a replication padding strategy is employed in order to mimic the zero-acoustic density gradient present at hard reflecting walls. aR represents the application of the ReLU non-linear activation function [49] and aL denotes applying an identity activation function (equivalent to no activation function), in order to allow the network to predict both positive and negative values at each scale output. Finally, $\#_{\text{in}} \rightarrow \#_{\text{out}}$ denotes the number of input and output feature fields at each convolution operation, from which the total number of trainable parameters in one layer can be deduced by performing the following operation $\#_{\text{parameters}} = (\#_{\text{in}} \times k\# \times k\# + 1) \times \#_{\text{out}}$.

Table A.1
Summary of operations and number of parameters for Multi-Scale CNN with 3 banks of convolutions.

G_1		G_2		G_3	
Layer	Parameters	Layer	Parameters	Layer	Parameters
Downsample N/4	–	Downsample N/2	–	k5s1p2aR 5 → 32	4032
k3s1p1aR 4 → 32	1184	k5s1p2aR 5 → 32	4032	k3s1p1aR 32 → 64	18,496
k3s1p1aR 32 → 64	18,496	k3s1p1aR 32 → 64	18,496	k3s1p1aR 64 → 128	73,856
k3s1p1aL 64 → 32	18,464	k3s1p1aR 64 → 128	73,856	k3s1p1aR 128 → 64	73,792
k3s1p1aL 32 → 1	289	k3s1p1aR 128 → 64	73,792	k3s1p1aL 64 → 32	18,464
Upsample $N \times 4$	–	k3s1p1aL 64 → 32	18,464	k5s1p2aL 32 → 8	6408
		k3s1p1aL 32 → 1	289	k1s1p0aL 8 → 1	9
		Upsample $N \times 2$	–		

Appendix B. Direct acoustic computation generation: The Lattice-Boltzmann method

The dataset is generated using the multi-physics lattice Boltzmann solver Palabos [59]. The equation that is being solved can be derived from the Boltzmann equation in the discrete velocity space

$$\frac{\partial f_i}{\partial t} + \mathbf{c}_i \cdot \nabla f_i = \Omega_i \tag{B.1}$$

where f_i is the discrete density distribution function, $\mathbf{c}_i$ is the discrete particle velocity in the ith direction and Ω_i is an operator representing the internal collisions between pairs of particles. The solver considers a second-order time–space discretization of Eq. (B.1):

$$f_i(\mathbf{x} + \mathbf{c}\Delta t, t + \Delta t) - f_i(\mathbf{x}, t) = \frac{\Delta t}{2}\left[\Omega_i(\mathbf{x}, t) + \Omega_i(\mathbf{x} + \mathbf{c}\Delta t, t + \Delta t)\right] \tag{B.2}$$

where $\mathbf{x}$ denotes the position, t the time and Δt the time-step.

The simplest form of the collision operator Ω_i corresponds to the BGK model which considers a relaxation of the particle populations f_i towards the local equilibrium f_i^{eq} with a relaxation time τ_f:

$$\Omega_i = \frac{1}{\tau_f}(f_i - f_i^{eq}) \tag{B.3}$$

where f_i^{eq} is the equilibrium distribution (employing Einstein summation convention to the index α):

$$f_i^{eq} = \rho w_i \left(1 + \frac{c_{i\alpha}u_\alpha}{c_s^2} + \frac{(c_{i\alpha}u_\alpha)^2}{2c_s^2} - \frac{u_\alpha u_\alpha}{2c_s^2}\right) \tag{B.4}$$

where c_s is the lattice speed of sound, w_i the weights of the lattice and u_α is the macroscopic velocity in the α direction.

The macroscopic quantities can be recovered through the statistical moments of the distribution functions: $\rho = \sum_i f_i$ and $\rho u_\alpha = \sum_i f_i c_{i\alpha}$. In this work, a more complex recursive and regularized version of the BGK collision model is employed to increase the numerical stability [63].

Appendix C. Influence of the dataset size

Although the cost of generating the dataset of 500 simulations remains small (2D simulations, few time steps and small spatial domain), the extrapolation of such a method to 3D configurations seems unfeasible with such large dataset sizes. The influence of the dataset size is assessed in this section. This can help finding a minimum dataset size for which the accuracy is acceptable, which could then be employed in 3D trainings. For the study, the baseline (denoted as "100%" size) corresponds to 500 simulations with 231 iterations sampled at $\Delta_{NN} = 3\Delta_{LB}$. In order to vary the size of the database, random samples of 4 inputs +1 output are taken in order to increase or reduce the total number of training samples (called data points). Results are shown in Fig. C.1a, depicting the error evolution averaged over 10 initial conditions of Gaussian pulses. The overall trend is that after a certain threshold (30% of the original dataset size), the increase of data points results in a marginal increment of accuracy. This analysis demonstrates that the employed approach can actually learn with fewer data from the one originally used in the baseline. It also suggests that this kind of 2D studies could be performed to get an estimate of an adequate dataset size when training neural networks for 3D predictions (e.g. 150 simulations instead of 500).

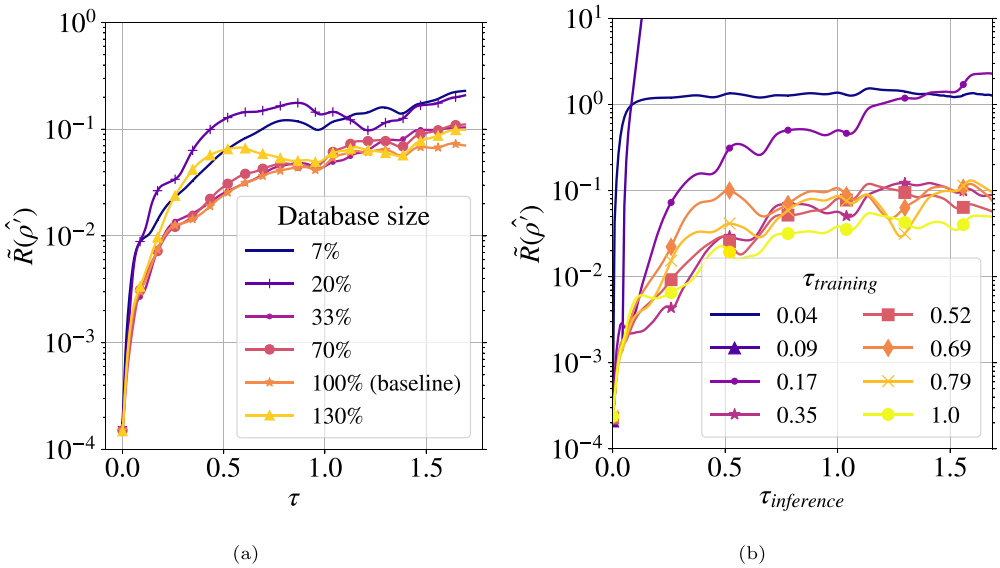

Fig. C.1. Evolution of relative error for case T_2^C (employing the EPC) with respect to dimensionless time $\tau = tc_0/D$, for 10 initial random Gaussian pulses, averaged for each neural network iteration. Study on the influence of (a) the dataset size and (b) the choice of the maximum τ_{training} used during training.

Appendix D. Influence of the dataset time τ

The influence of the time τ_{training} when the training database simulation is stopped is studied here, in order to test whether the neural network overfits the non-dimensional time seen during training. Several datasets have been created, by changing the time horizon of the training simulations. The comparison is performed at iso-dataset size, the total number of data points (groups of 4 inputs and 1 target snapshot) is kept constant at 20% of the original baseline size to speed-up trainings. This implies that for large τ_{training}, fewer simulations with more time-steps are required, while for low τ_{training}, more initial conditions are used, with a low count of iterations per simulation. Even if there can be an implicit bias in the dataset related to the difference in the number of initial conditions, the proposed study is the only way to study the influence of the τ_{training} parameter without accounting for the effect of the dataset size. The time-step of the neural networks remains fixed at $\Delta_{NN} = 3\Delta_{LB}$, and during the inference phase, the surrogate model is stepped up to reach $\tau = 1.6$. Fig. C.1b shows the comparison between the several trainings, for strategy T_2^C. For low non-dimensional times ($\tau_{\text{training}} < 0.17$), the 3 trained neural networks produce inaccurate results. This is explained because of the lack of sufficient examples of wave–wall interactions in the dataset. After a certain threshold ($\tau_{\text{training}} > 0.35$), the neural network accuracy no longer depends on the choice of τ_{training}, as the error follows a very similar trend for all the trained neural networks.

References

[1] S.K. Lele, J.W. Nichols, A second golden age of aeroacoustics?, Phil. Trans. R. Soc. A: Math. Phys. Eng. Sci. 372 (2022) (2014) http://dx.doi.org/10.1098/rsta.2013.0321.

[2] M.C. Jacob, D. Dragna, A. Cahuzac, J. Boudet, P. Blanc-Benon, Toward hybrid CAA with ground effects, Int. J. Aeroacoust. 13 (3–4) (2014) 235–259, http://dx.doi.org/10.1260/1475-472X.13.3-4.235.

[3] N. Schoenwald, L. Panek, C. Richer, F. Thiele, Investigation of sound radiation from a scarfed intake by CAA-fwh simulations using overset grids, in: 13th AIAA/CEAS Aeroacoustics Conference (28th AIAA Aeroacoustics Conference), 2007, p. 3534, http://dx.doi.org/10.2514/6.2007-3524.

[4] M. Sanjose, A. Towne, P. Jaiswal, S. Moreau, S. Lele, A. Mann, Modal analysis of the laminar boundary layer instability and tonal noise of an airfoil at Reynolds number 150,000, Int. J. Aeroacoust. 18 (2–3) (2019) 317–350, http://dx.doi.org/10.1177/1475472X18812798.

[5] J.E. Ffowcs Williams, D.L. Hawkings, Sound generation by turbulence and surfaces in arbitrary motion, Phil. Trans. R. Soc. A: Math. Phys. Eng. Sci. 264 (1151) (1969) 321, 342, http://dx.doi.org/10.1098/rsta.1969.0031.

[6] B.A. Singer, D.P. Lockard, G.M. Lilley, Hybrid acoustic predictions, Comput. Math. Appl. 46 (4) (2003) 647–669, http://dx.doi.org/10.1016/S0898-1221(03)90023-X.

[7] N. Schönwald, Ł. Panek, C. Richter, F. Thiele, Mode propagation in bifurcated bypass ducts: Application oriented simulation approach, in: 16th AIAA/CEAS Aeroacoustics Conference (31st AIAA Aeroacoustics Conference), 2010, http://dx.doi.org/10.2514/6.2010-3895.

[8] C. Pérez Arroyo, T. Leonard, M. Sanjosé, S. Moreau, F. Duchaine, Large eddy simulation of a scale-model turbofan for fan noise source diagnostic, J. Sound Vib. 445 (2019) 64–76, http://dx.doi.org/10.1016/j.jsv.2019.01.005.

[9] M.E. Goldstein, Aeroacoustics, McGraw-Hill International Book Co New York, 1976.

[10] S.L. Brunton, B.R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, Annu. Rev. Fluid Mech. 52 (1) (2020) 477–508, http://dx.doi.org/10.1146/annurev-fluid-010719-060214.

[11] G. Berkooz, P. Holmes, J.L. Lumley, The proper orthogonal decomposition in the analysis of turbulent flows, Annu. Rev. Fluid Mech. 25 (1) (1993) 539–575, http://dx.doi.org/10.1146/annurev.fl.25.010193.002543.

[12] P. Schmid, Dynamic mode decomposition of numerical and experimental data, J. Fluid Mech. 656 (2010) 5–28, http://dx.doi.org/10.1017/s0022112010001217\"\i.
[13] K. Taira, S.L. Brunton, S.T. Dawson, C.W. Rowley, T. Colonius, B.J. McKeon, O.T. Schmidt, S. Gordeyev, V. Theofilis, L.S. Ukeiley, Modal analysis of fluid flows: An overview, AIAA J. 55 (12) (2017) 4013–4041, http://dx.doi.org/10.2514/1.J056060, URL: arXiv:1702.01453, www.aiaa.org/randp.
[14] P. Kholodov, S. Moreau, Identification of noise sources in a realistic turbofan rotor using large eddy simulation, Acoustics 2 (3) (2020) 691–706, http://dx.doi.org/10.3390/acoustics2030037.
[15] Z. Bai, E. Kaiser, J.L. Proctor, J.N. Kutz, S.L. Brunton, Dynamic mode decomposition for compressive system identification, AIAA J. 58 (2) (2020) 561–574, http://dx.doi.org/10.2514/1.J057870.
[16] G. Tissot, L. Cordier, N. Benard, B.R. Noack, Model reduction using dynamic mode decomposition, Compt. Rendus - Mecanique 342 (6–7) (2014) 410–416, http://dx.doi.org/10.1016/j.crme.2013.12.011.
[17] M.J. Balajewicz, E.H. Dowell, B.R. Noack, Low-dimensional modelling of high-Reynolds-number shear flows incorporating constraints from the Navier-Stokes equation, J. Fluid Mech. 729 (2013) 285, http://dx.doi.org/10.1017/jfm.2013.278.
[18] A. Barbagallo, D. Sipp, P.J. Schmid, Closed-loop control of an open cavity flow using reduced-order models, J. Fluid Mech. 641 (2009) 1–50, http://dx.doi.org/10.1017/S0022112009991418.
[19] A.S. Sharma, I. Mezić, B.J. McKeon, Correspondence between koopman mode decomposition, resolvent mode decomposition, and invariant solutions of the Navier-Stokes equations, Phys. Rev. Fluids 1 (3) (2016) 032402, http://dx.doi.org/10.1103/physrevfluids.1.032402, arXiv:1603.08378.
[20] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366, http://dx.doi.org/10.1016/0893-6080(89)90020-8.
[21] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (5) (1998) 987–1000, http://dx.doi.org/10.1109/72.712178, arXiv:9705023.
[22] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural Comput. 1 (4) (1989) 541–551, http://dx.doi.org/10.1162/neco.1989.1.4.541.
[23] Y. LeCun, L. Bottou, Y. Bengio, P. Haaner, Gradient-based learning applied to document recognition, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 86, (11) 1998, pp. 2278–2324, http://dx.doi.org/10.1109/5.726791.
[24] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2017) 84–90, http://dx.doi.org/10.1145/3065386.
[25] N. Thuerey, K. Weißenow, L. Prantl, X. Hu, Deep learning methods for Reynolds-averaged Navier–Stokes simulations of airfoil flows, AIAA J. 58 (1) (2020) 25–36, http://dx.doi.org/10.2514/1.j058291, arXiv:1810.08217.
[26] S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, S. Kaushik, Prediction of aerodynamic flow fields using convolutional neural networks, Comput. Mech. 64 (2) (2019) 525–545, http://dx.doi.org/10.1007/s00466-019-01740-0.
[27] J. Tompson, K. Schlachter, P. Sprechmann, K. Perlin, Accelerating eulerian fluid simulation with convolutional networks, in: Proceedings of the 34th International Conference on Machine Learning (ICML), Sydney, Australia, 2017, pp. 80:3424–3433, URL: arXiv:1607.03597, http://proceedings.mlr.press/v70/tompson17a.html.
[28] C. Yang, X. Yang, X. Xiao, Data-driven projection method in fluid simulation, in: Comput. Animation Virt. Worlds, 27, (3–4) John Wiley and Sons Ltd, 2016, pp. 415–424, http://dx.doi.org/10.1002/cav.1695.
[29] V. Sekar, M. Zhang, C. Shu, B.C. Khoo, Inverse design of airfoil using a deep convolutional neural network, AIAA J. 57 (3) (2018) 993–1003, http://dx.doi.org/10.2514/1.J057894.
[30] P. Baque, E. Remelli, F. Fleuret, P. Fua, Geodesic Convolutional Shape Optimization, in: Proceedings of the 35th International Conference on Machine Learning (ICML), Stockholm, Sweden, 2018, pp. 80:472–481.
[31] K. Fukami, K. Fukagata, K. Taira, Super-resolution reconstruction of turbulent flows with machine learning, J. Fluid Mech. 870 (2019) 106–120, http://dx.doi.org/10.1017/jfm.2019.238, arXiv:1811.11328.
[32] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[33] S. Hochreiter, J.J. Urgen Schmidhuber, Long-short term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[34] S. Wiewel, M. Becher, N. Thuerey, Latent space physics: Towards learning the temporal evolution of fluid flow, Comput. Graph. Forum 38 (2) (2019) 71–82, http://dx.doi.org/10.1111/cgf.13620, arXiv:1802.10123.
[35] S. Wiewel, B. Kim, V.C. Azevedo, B. Solenthaler, N. Thuerey, Latent space subdivision : Stable and controllable time predictions for fluid flow, Comput. Graph. Forum 39 (8) (2020) 15–25.
[36] A.P. Singh, S. Medida, K. Duraisamy, Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils, AIAA J. 55 (7) (2017) 2215–2227, http://dx.doi.org/10.2514/1.J055595, arXiv:1608.03990.
[37] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707, http://dx.doi.org/10.1016/j.jcp.2018.10.045.
[38] M. Raissi, Z. Wang, M.S. Triantafyllou, G.E. Karniadakis, Deep learning of vortex induced vibrations, J. Fluid Mech. 861 (2019) 119–137, URL: arXiv:1808.08952v1.
[39] M. Raissi, Deep hidden physics models: Deep learning of nonlinear partial differential equations, J. Mach. Learn. Res. 19 (2018) 1–24, URL: http://jmlr.org/papers/v19/18-046.html.
[40] Y. Bar-Sinai, S. Hoyer, J. Hickey, M.P. Brenner, Learning data-driven discretizations for partial differential equations, in: Proceedings of the National Academy of Sciences, Vol. 116, (31) 2019, pp. 15344–15349, http://dx.doi.org/10.1073/pnas.1814058116.
[41] A.S. Tenney, M.N. Glauser, C.J. Ruscher, Z.P. Berger, Application of artificial neural networks to stochastic estimation and jet noise modeling, AIAA J. 58 (2) (2020) 647–658, http://dx.doi.org/10.2514/1.J058638.
[42] S. Sack, M. Åbom, Acoustic plane-wave decomposition by means of multilayer perceptron neural networks, J. Sound Vib. 486 (2020) 115518, http://dx.doi.org/10.1016/j.jsv.2020.115518.
[43] M.J. Bianco, P. Gerstoft, J. Traer, E. Ozanich, M.A. Roch, S. Gannot, C.-A. Deledalle, Machine learning in acoustics: Theory and applications, J. Acoust. Soc. Am. 146 (5) (2019) 3590–3628, http://dx.doi.org/10.1121/1.5133944, URL: arXiv:1905.04418.
[44] W. Zhu, Y. Sheng, Y. Sun, Wave-dynamics simulation using deep neural networks, Technical Report, Stanford CS231N reports, 2017, URL: http://cs231n.stanford.edu/reports/2017/pdfs/542.pdf.
[45] W.E. Sorteberg, S. Garasto, A.S. Pouplin, C.D. Cantwell, A.A. Bharath, Approximating the solution to wave propagation using deep neural networks, in: 32nd Conference on Neural Information Processing Systems - NeurIPS, 2018, URL: arXiv:1812.01609.
[46] S. Fotiadis, E. Pignatelli, A.A. Bharath, M. Lino Valencia, C.D. Cantwell, A. Storkey, Comparing recurrent and convolutional neural networks for predicting wave propagation, in: ICLR 2020 Workshop on Deep Learning and Differential Equations, 2020, URL: arXiv:2002.08981v2.
[47] S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient pathologies in physics-informed neural networks, 2020, ArXiv:2001.04536, URL: arXiv:2001.04536v1, https://arxiv.org/abs/2001.04536.
[48] Y. LeCun, K. Kavukcuoglu, C. Farabet, Convolutional networks and applications in vision, in: IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems, 2010, pp. 253–256, http://dx.doi.org/10.1109/ISCAS.2010.5537907.

[49] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Proceedings of the 27th International Conference on Machine Learning (ICML), 2010, pp. 807–814, URL: https://dl.acm.org/doi/10.5555/3104322.3104425.

[50] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533–536, http://dx.doi.org/10.1038/323533a0.

[51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. Devito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An Imperative Style, High-Performance Deep Learning Library, in: Advances in Neural Information Processing Systems, Vol. 32, Vancouver, Canada, 2019, pp. 8024–8035, URL: https://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.

[52] M. Mathieu, C. Couprie, Y. LeCun, Deep multi-scale video prediction beyond mean square error, in: 6th International Conference on Learning Representations (ICLR), Vancouver, Canada, 2016, URL: arXiv:1511.05440.

[53] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, Medical Image Comput. Comput. Assisted Intervent. (MICCAI) 9351 (2015) 234–241, http://dx.doi.org/10.1007/978-3-319-24574-4-28, arXiv:1505.04597v1.

[54] C.J. Lapeyre, A. Misdariis, N. Cazard, D. Veynante, T. Poinsot, Training convolutional neural networks to estimate turbulent sub-grid scale reaction rates, Combust. Flame 203 (2019) 255–264, http://dx.doi.org/10.1016/j.combustflame.2019.02.019, arXiv:1810.03691.

[55] A. Alguacil, M. Bauerheim, M.C. Jacob, S. Moreau, Predicting the Propagation of Acoustic Waves using Deep Convolutional Neural Networks, in: AIAA Aviation Forum, Reston, VA, 2020, pp. 2513, http://dx.doi.org/10.2514/6.2020-2513.

[56] E. Ajuria Illarramendi, A. Alguacil, M. Bauerheim, A. Misdariis, B. Cuenot, E. Benazera, Towards an hybrid computational strategy based on deep learning for incompressible flows, in: AIAA Aviation Forum, American Institute of Aeronautics and Astronautics (AIAA), Reston, Virginia, 2020, p. 3058, http://dx.doi.org/10.2514/6.2020-3058.

[57] S. Lee, D. You, Data-driven prediction of unsteady flow over a circular cylinder using deep learning, J. Fluid Mech. 879 (1) (2019) 217–254, http://dx.doi.org/10.1017/jfm.2019.700, arXiv:1804.06076.

[58] S. Lee, D. You, Mechanisms of a convolutional neural network for learning three-dimensional unsteady wake flow, in: 72nd Annual Meeting of the APS Division of Fluid Dynamics, APS, 2019, p. 123, arXiv:1909.06042v1.

[59] J. Latt, O. Malaspinas, D. Kontaxakis, A. Parmigiani, D. Lagrava, F. Brogi, M.B. Belgacem, Y. Thorimbert, S. Leclaire, S. Li, F. Marson, J. Lemus, C. Kotsalos, R. Conradin, C. Coreixas, R. Petkantchin, F. Raynaud, J. Beny, B. Chopard, Palabos: Parallel lattice Boltzmann solver, Comput. Math. Appl. (2020) http://dx.doi.org/10.1016/j.camwa.2020.03.022.

[60] S. Marié, D. Ricot, P. Sagaut, Comparison between lattice Boltzmann method and Navier-Stokes high order schemes for computational aeroacoustics, J. Comput. Phys. 228 (2008) 1056–1070, http://dx.doi.org/10.1016/j.jcp.2008.10.021.

[61] G.A. Brès, F. Pérot, D. Freed, Properties of the lattice-Boltzmann method for acoustics, in: 15th AIAA/CEAS Aeroacoustics Conference (30th AIAA Aeroacoustics Conference), 2009, p. 3395, http://dx.doi.org/10.2514/6.2009-3395.

[62] F. Brogi, O. Malaspinas, B. Chopard, C. Bonadonna, Hermite regularization of the lattice Boltzmann method for open source computational aeroacoustics, J. Acoust. Soc. Am. 142 (4) (2017) 2332–2345, http://dx.doi.org/10.1121/1.5006900.

[63] O. Malaspinas, Increasing stability and accuracy of the lattice Boltzmann scheme: recursivity and regularization, 2015, ArXiv:1505.06900, URL: arXiv:1505.06900v1, https://arxiv.org/abs/1505.06900.

[64] A.J. Ladd, Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation, J. Fluid Mech. 271 (1994) 285–309, http://dx.doi.org/10.1017/S0022112094001771, arXiv:9306004.

[65] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, E.M. Viggen, The lattice Boltzmann method; principles and practice, Springer International Publishing, 2017, p. 694, http://dx.doi.org/10.1007/978-3-319-44649-3.

[66] C.K. Tam, J.C. Webb, Dispersion-relation-preserving finite difference schemes for computational acoustics, J. Comput. Phys. 107 (2) (1993) 262–281, http://dx.doi.org/10.1006/jcph.1993.1142.

[67] P.J. Boache, Perspective: A method for uniform reporting of grid refinement studies, J. Fluids Eng. Trans. ASME 116 (3) (1994) 405–413, http://dx.doi.org/10.1115/1.2910291.

[68] D.P. Kingma, J.L. Ba, Adam: A method for stochastic optimization, in: 3rd International Conference on Learning Representations - ICLR, 2015, URL arXiv:1412.6980, https://hdl.handle.net/11245/1.505367.

[69] Y. Liu, J. Nathan Kutz, S.L. Brunton, Hierarchical deep learning of multiscale differential equation time-steppers, 2020, ArXiv Preprint arXiv:2008.09768 URL: arXiv:2008.09768v1, https://github.com/luckystarufo/multiscale_HiTS.