



OPEN

Generative adversarial reduced order modelling

Dario Coscia, Nicola Demo & Gianluigi Rozza

In this work, we present GAROM, a new approach for reduced order modeling (ROM) based on generative adversarial networks (GANs). GANs attempt to learn to generate data with the same statistics of the underlying distribution of a dataset, using two neural networks, namely discriminator and generator. While widely applied in many areas of deep learning, little research is done on their application for ROM, i.e. approximating a high-fidelity model with a simpler one. In this work, we combine the GAN and ROM framework, introducing a data-driven generative adversarial model able to learn solutions to parametric differential equations. In the presented methodology, the discriminator is modeled as an autoencoder, extracting relevant features of the input, and a conditioning mechanism is applied to the generator and discriminator networks specifying the differential equation parameters. We show how to apply our methodology for inference, provide experimental evidence of the model generalization, and perform a convergence study of the method.

Partial differential equations (PDEs) can model complex physical processes, ranging from fluid dynamics, and aerospace engineering to material science^{1–3}. Nevertheless, only specific types of PDEs can be analytically solved, while most require numerical approximations such as finite difference or finite element methods. Numerical solvers, as the aforementioned ones, have many qualities such as error estimation for the solution, stability for long-time simulations, and generalization across different boundary/ initial conditions or geometries^{4,5}. Nevertheless, these numerical approaches are extremely costly regarding computational resources, requiring significant time to obtain simulation results. This is even worse in parametric contexts, where a new simulation must be performed for each set of new parameters, making unfeasible real-time computations. To overcome this barrier, reduced order models (ROMs) have become an emerging field in computational sciences, fulfilling the demand for efficient computational tools for performing real-time simulations^{6–10}. In particular, data-driven ROMs have grasped increasing attention from the scientific community due to their ability to generalize across multiple system configurations using only a finite set of data. This modeling approach is “agnostic” in the sense that the equations describing the phenomenon are not required, rendering the methodology versatile even when data originate from real-world observations.

ROMs for parametric PDEs are usually composed of two steps: dimensionality reduction, and manifold interpolation. In the dimensionality reduction phase, a low-dimensional space is employed to optimally represent the problem at hand; while the manifold interpolation procedure is used for mapping the parameter space to the mode of variation coefficients space. One of the most used techniques for the ROM's (linear) dimensionality reduction phase is the proper orthogonal decomposition (POD)¹¹, while radial basis functions (RBFs)¹² are usually used as a standard interpolation procedure. Data-driven ROM using linear dimensionality reduction like POD has the major drawback that exhibits limitations for non-linear dynamics¹³.

Deep learning (DL) algorithms have recently emerged as new approaches for performing non-linear ROM using autoencoders^{14–17}. These architectures consist of two neural networks, namely encoder and decoder. The encoder network is used for compressing the data onto a lower-dimensional manifold, while the decoder performs the opposite transformation. The final objective is to minimize the reconstruction error between the encoded-decoded data and the original one, quantified by a specific norm. For interpolating the reduced manifold, the RBF procedure can be used, or neural networks, such as long short-term memory (LSTM) networks for temporal predictions^{18,19}, or feed-forward neural networks in parametric problems²⁰ can be employed.

Regardless of the great advancement achieved in generalizability by neural networks^{21–25}, only discriminative models have been extensively investigated. Discriminative models, such as regression techniques, are machine learning models optimized for learning the conditional probability of obtaining an output given a specific input. In other words, they focus on learning decision boundaries that separate (discriminative part) the data during training. During inference, the new data point is mapped to the respective boundary, depending on a specific distance metric²⁶. Nevertheless, these models lack a semantic understanding of the phenomenology characterizing the data²⁷, and quantifying the uncertainty may be challenging. On the other hand, generative modeling is

Mathematics Area, mathLab, SISSA, via Bonomea 265, I-34136 Trieste, Italy. email: gianluigi.rozza@sissa.it

used to model the dataset probability distribution using a probabilistic approach. Indeed, generative models aim to learn how to generate new data with the same statistics as the underlying dataset distribution, which forces the network to learn specific patterns in the data. Latent variable models²⁸ are an example of generative models that have been successfully applied in many fields, see Refs.^{29–32} as far from an exhaustive list of possible applications.

Recently, latent variable model approaches, specifically variational autoencoder (VAE) models³³, have been applied for reduced order modeling. In particular, in Ref.³⁴ a ROM for fluid flow predictions is presented. The model uses a VAE architecture for dimensionality reduction to produce near-orthogonal representations as in Ref.³⁵, while the interpolation is done using a transformer networks³⁶. VAEs represent a powerful subclass of latent variable models, namely prescribed models. The (parametric) distributions defining the probabilistic model must be chosen upfront in prescribed models. Nevertheless, the quality of the model can be affected if too simplistic distributions are used^{27,37}.

The implicit models — different from the above-mentioned ones — distribution. In particular in GANs, the objective is achieved by implicitly modelling the true unknown probability distribution through a generator network trained by adversarial learning.

The usage of GANs for solving PDEs is an emerging research line, and several works have already addressed the problem. Specifically, in Ref.³⁸ a GAN is used to learn solutions across multiple initial conditions for a specific PDE, by combining the GAN loss with a physics-informed³⁹ one. Other examples of physics-informed GAN are the works in Ref.^{40–42}. The first one⁴⁰, trained by minimizing the classical GAN loss, injects physical information during training by augmenting the generator and discriminator with a physics consistency score, assessing how physical the produced generator field is. The second one⁴¹, on the other hand, uses the GAN architecture to learn the loss for a physics-informed neural network³⁹, i.e. the training is unsupervised. Finally, the third one⁴² uses a variation of the methodology in Ref.⁴⁰ to learn stochastic PDEs. In contrast from the models above where the methodology is limited to learning solutions for a fixed PDE, the works in Refs.^{43,44} introduce a way to condition the generative process for parametric and time-dependent PDEs. A common point of the mentioned methodologies is the fact that vanilla GANs, as introduced in Ref.⁴⁵, are used in training with the addition of physical information. However, in Ref.⁴⁶ the authors show that vanilla GANs, i.e. without any physical information, obtain in general higher errors compared to discriminative models. Since the addition of physical loss to the GAN loss can always be done, we seek in the article to introduce a methodology for solving parametric PDEs that can compete against discriminative models by leveraging only the GAN loss. In addition, differently from Refs.^{43,44} we want the generator to not be deterministic, i.e. we want to introduce stochasticity in the network input in order to perform uncertainty quantification as done in Refs.^{47,48}.

In this work, we present GAROM, a generative adversarial reduce order model. GAROM is a novel reduced order model framework, based on a variation of conditional boundary equilibrium GAN⁴⁹. In particular, GAROM uses adversarial learning to learn the distribution over high-fidelity data, conditioned on domain-specific conditioning. At convergence, the GAROM network is able to generate high-fidelity data given domain-specific conditioning, e.g. PDE parameters, or temporal time steps. We also present a regularized version of the methodology, r-GAROM, providing extra information on the generative process as explained in Sect. [Methods](#). The frameworks are tested in a variety of problems, comparing the prediction ability to state-of-the-art ROM methods, and analyzing the convergence robustness. Empirical results on the network's ability to generalize to unseen data are provided, and different uncertainty quantification strategies based on statistics are presented. Finally, we provide further possible research directions to investigate this novel methodology.

Results

In this section, the performances of GAROM and its improved variant r-GAROM are presented. To estimate the accuracy of the presented methodology, we use the mean l_2 relative error ϵ between the high-fidelity solution $\mathbf{u}(\mathbf{c}) \in \mathbb{R}^{N_u}$, and the generated one $\hat{\mathbf{u}}(\mathbf{c}) \in \mathbb{R}^{N_u}$, for the parameter $\mathbf{c} \in \mathbb{R}^{N_c}$. It is formally defined as:

$$\epsilon = \frac{1}{N} \sum_{i=1}^N \frac{\|\mathbf{u}(\mathbf{c}_i) - \hat{\mathbf{u}}(\mathbf{c}_i)\|_2^2}{\|\mathbf{u}(\mathbf{c}_i)\|_2^2}, \quad (1)$$

with N the number of testing parameters used for the metric evaluation.

All of our experiments are performed on three different test cases, representing benchmark problems in parametric contexts. Aiming at increasing complexity, we take into account a Gaussian signal moving inside a square (algebraic), the Graetz-Poiseuille problem (linear PDE), and the Lid Cavity problem (nonlinear PDE). The datasets for the training are therefore composed of the (parameter-dependent) high-fidelity solutions, which are obtained by solving such problems with the consolidated numerical solver, as exhaustively explained in section "[Datasets description](#)". Moreover, for all the test cases, different latent dimensions for the discriminator network are considered. The other architectural details for the models at hand are fully described in section "[Training setup and model architecture](#)", together with the optimizer settings.

Model inference

In the first experiment, we test the GAROM capability to predict the solution for a new parameter, aka conditioning variable. A visualization of the r-GAROM prediction for a testing parameter, and its l_2 point-wise error is reported in Fig. 1. The spatial distribution of the prediction for all the test cases is almost identical to the original one, as proved also by the low error. The variance reaches low values in all the problems, demonstrating that the model has learned to properly generate the solution field. The spatial distributions of variance and error do not show similarities, but the former identifies the spatial region where the network is more uncertain (we highlight that the plots present the variance for a single test parameter), which can be potentially used to

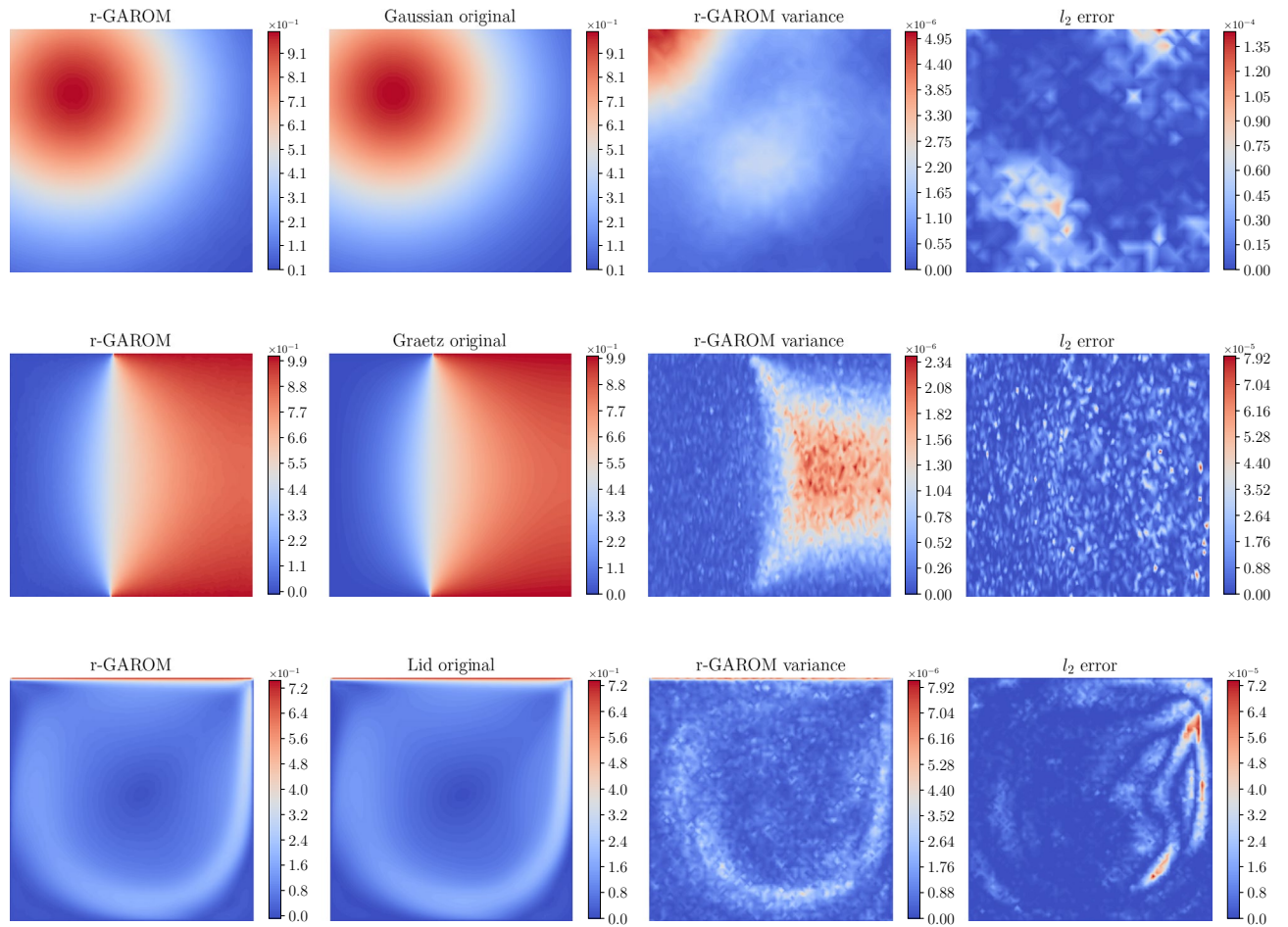


Figure 1. r-GAROM inference results. The images show the generated snapshot representing the magnitude of the unknown field for a testing parameter using a latent dimension of 64, compared to the corresponding high-fidelity solution. *Top:* Gaussian dataset. *Center:* Graetz dataset. *Bottom:* Lid cavity dataset.

adapt the training procedure. A similar analysis like the one in Fig. 1 has been replicated also for the GAROM model. The unregularized model results are barely indistinguishable to the eye with respect to the regularized counterpart, so we have not reported them.

Looking at a more precise measure for the GAROM accuracy, we calculate the mean l_2 relative error over a set of training and testing parameters, comparing it with the error obtained using the ROM baseline models: POD-RBF, POD-NN, AE-RBF, and AE-NN; the generative modelling baselines: cGAN, and r-cGAN; and the Operator Learning baselines: DeepONet, and NOMAD. As we can note from Table 1, for the majority of the experiments POD-RBF is the best across all models on the training dataset (gray rows), but it fails to properly generalize to test data. Indeed, the error computed on the test data exhibits a completely different trend with respect to the one computed over the training snapshots, especially in the Graetz case, showing a poor ability to correctly generalize. On the contrary, ML-based methodologies exhibit comparable train and test errors, due to their ability to learn complex nonlinear relationships by the data. For the test error, which is used to assess the model prediction ability, the r-GAROM model obtains very promising results for almost half of the test cases. Across the different deep learning models, r-GAROM obtains lower error for six out of nine tests, highlighting its encouraging precision even in this preliminary contribution. It is also important to highlight that the (r-)GAROM architecture is fairly simple in all the numerical tests here pursued: both generator and discriminator are composed only of a few dense layers (details in section "Training setup and model architecture"). We suppose the accuracy can be further improved by using more powerful deep learning architectures, as in Refs.^{15,50,51}, at the cost of longer and more expensive training.

Compared to the vanilla (r-)cGAN, the (r-)GAROM methodology obtains lower error in both training and testing, with the difference in accuracy especially evidenced in the most challenging tests, i.e. the Graetz and Lid Cavity. Furthermore, the predictive standard deviation for the (r-)GAROM model is much lower than the (r-)cGAN model, empirically showing that the GAROM model is less uncertain (see *uq* for more details). Finally, the r-GAROM error seems to be almost independent of the latent dimension, since only a small variation of the errors is observed in training and testing for the various dimensions. In such a way, we can empirically prove the better capability to reduce the original dimension of the snapshots, since adding new reduced dimensions does not affect the final accuracy.

Method	Gaussian			Graetz			Lid Cavity		
	4	16	64	16	64	120	16	64	120
GAROM	(1.05 ± 0.17)	(0.77 ± 0.20)	(0.88 ± 0.16)	(0.58 ± 0.16)	(0.55 ± 0.14)	(0.51 ± 0.1)	(10.7 ± 0.22)	(9.50 ± 0.15)	(9.91 ± 0.17)
r-GAROM	(0.80 ± 0.15)	(0.60 ± 0.14)	(0.65 ± 0.14)	(0.32 ± 0.09)	(0.22 ± 0.05)	(0.20 ± 0.04)	(5.02 ± 0.11)	(3.41 ± 0.04)	(3.61 ± 0.06)
cGAN	(7.72 ± 0.30)	(7.72 ± 0.30)	(7.72 ± 0.30)	(21.3 ± 1.10)	(21.3 ± 1.10)	(21.3 ± 1.10)	(68.5 ± 1.55)	(68.5 ± 1.55)	(68.5 ± 1.55)
r-cGAN	(6.04 ± 0.27)	(6.04 ± 0.27)	(6.04 ± 0.27)	(1.65 ± 0.05)	(1.65 ± 0.05)	(1.65 ± 0.05)	(51.6 ± 1.11)	(51.6 ± 1.11)	(51.6 ± 1.11)
POD-RBF	15.4	0.41	0.25	0.48	0.48	0.49	2.98	1.45	1.32
POD-NN	15.4	1.30	1.12	0.41	0.47	0.44	3.02	2.73	3.20
AE-RBF	2.06	0.89	1.00	1.08	1.42	1.60	6.96	3.01	3.90
AE-NN	1.76	1.04	1.12	1.14	1.56	2.71	3.49	3.27	3.83
DeepONet	36.1	36.1	36.1	0.73	0.73	0.65	72.2	72.2	72.2
NOMAD	1.99	1.99	1.99	0.36	0.36	0.36	63.4	63.4	63.4
GAROM	(1.04 ± 0.01)	(0.73 ± 0.01)	(0.86 ± 0.01)	(0.69 ± 0.01)	(0.62 ± 0.01)	(0.57 ± 0.01)	(13.5 ± 0.01)	(11.5 ± 0.01)	(12.6 ± 0.01)
r-GAROM	(0.78 ± 0.01)	(0.54 ± 0.01)	(0.60 ± 0.01)	(0.54 ± 0.009)	(0.19 ± 0.004)	(0.18 ± 0.004)	(6.36 ± 0.007)	(3.75 ± 0.003)	(4.04 ± 0.004)
cGAN	(7.04 ± 0.30)	(7.04 ± 0.30)	(7.04 ± 0.30)	(21.6 ± 1.10)	(21.6 ± 1.10)	(21.6 ± 1.10)	(82.75 ± 1.50)	(82.75 ± 1.50)	(82.75 ± 1.50)
r-cGAN	(5.93 ± 0.27)	(5.93 ± 0.27)	(5.93 ± 0.27)	(1.78 ± 0.05)	(1.78 ± 0.05)	(1.78 ± 0.05)	(64.90 ± 1.11)	(64.90 ± 1.11)	(64.90 ± 1.11)
POD-RBF	15.2	0.23	2.85 · 10⁻⁵	7.73 · 10⁻⁴	1.51 · 10⁻¹¹	2.80 · 10⁻¹²	2.37	0.29	0.04
POD-NN	15.2	0.65	0.59	0.36	0.38	0.40	3.08	2.90	3.26
AE-RBF	1.90	0.70	0.83	1.01	1.41	1.54	7.24	2.73	3.68
AE-NN	1.40	0.82	0.96	1.16	1.67	2.78	3.60	3.34	4.02
DeepONet	35.80	35.80	35.80	0.65	0.65	0.65	76.85	76.85	76.85
NOMAD	1.98	1.98	1.98	0.42	0.42	0.42	78.73	78.73	78.73

Training Testing

Table 1. Accuracy comparison for different test cases, methods, and latent dimensions. The table reports the mean l_2 relative error ϵ (in percent). Gray rows represent the error on training data, while white rows the error on testing data. For GAROM and r-GAROM models, also the predictive standard deviation is reported (in percent). The (r)-cGAN, DeepONet and NOMAD models results do not depend on the hidden dimension, thus we report the same error value. The best models are marked in bold.

Model variance

Figure 1 reports the (r)-GAROM variance obtained by a Monte Carlo approximation with different random inputs for the generator. Contrary to the ROM baseline models, GAROM can provide also an estimate of the model variance, obtained by a probabilistic ensemble (see section "GAROM predictive distribution"). Since the solution is unique by assumption, we expect a zero variance for the generated samples given the conditioning variables. Hence, a high variance in the GAROM estimate gives an uncertain prediction of the model. This is confirmed by observing the variances for different models in Table 1, and the point-wise variance in Fig. 1. Both GAROM and r-GAROM models report a small variance, with training variance always one or two orders of magnitude smaller than the testing one. This indicates that GAROM models are more uncertain in predicting testing data than training ones. This is expected since the GAROM model builds a distribution on the training data. However, it is worth noticing that the l_2 errors between training and testing in Table 1 are almost always identical across different dimensions, showing good generalizability of the model, as also reported in section "Generalization and robustness". It is evident from Table 1 that r-GAROM has a lower variance than GAROM for testing and training. This is mainly due to the regularizer term, which forces the network to converge faster, as explained in the section "Methods".

The variance estimate, although very important for the aforementioned reasons, does not give a quantification of the predictive uncertainty, but only statistical information of the learned distribution. Nevertheless, due to the probabilistic framework adopted, we can obtain bounds in probability for the reconstruction error by using the variance estimate. Indeed, as deeply explained in section "Uncertainty quantification strategies", with the variance information provided by the GAROM framework, it is possible to obtain prediction uncertainty using the Markov inequality, or the confidence interval theory. While extremely important, in this work, we just show how to obtain these naïve bounds. Exploring further to obtain tighter bounds, or employ the uncertainty quantification strategies for noisy data are possible new interesting research topics, which we will explore in the future.

Generalization and robustness

One fundamental aspect, especially in the context of data-driven modelling, is the ability to correctly generalize across new instances of the conditioning domain, which of course were not inside the training dataset. To empirically prove the GAROM ability to generalize over the testing parameters, we define the difference $\delta(\mathbf{c})$ as the mean difference between the truth parametric solution and the corresponding GAROM prediction, such that:

$$\delta(\mathbf{c}) = \frac{1}{N_u} \sum_{i=1}^{N_u} (u_i(\mathbf{c}) - \hat{u}_i(\mathbf{c})), \quad (2)$$

where $\mathbf{u}(\mathbf{c}) = [u_1(\mathbf{c}) \dots u_{N_u}(\mathbf{c})]$ and $\hat{\mathbf{u}}(\mathbf{c}) = [\hat{u}_1(\mathbf{c}) \dots \hat{u}_{N_u}(\mathbf{c})]$. We highlight that we do not consider the absolute value here in order to detect possible systematic over- and under-estimations of the proposed model.

Figure 2 shows the distribution of the δ different for all latent dimensions and for all test cases. We can note that the distribution of the error computed over the training parameters is quite similar to the error distribution for the testing parameters in all the numerical investigations. Such behaviour indicates that both models can correctly generalize outside the training dataset, for the proposed problems. Furthermore, the distributions are

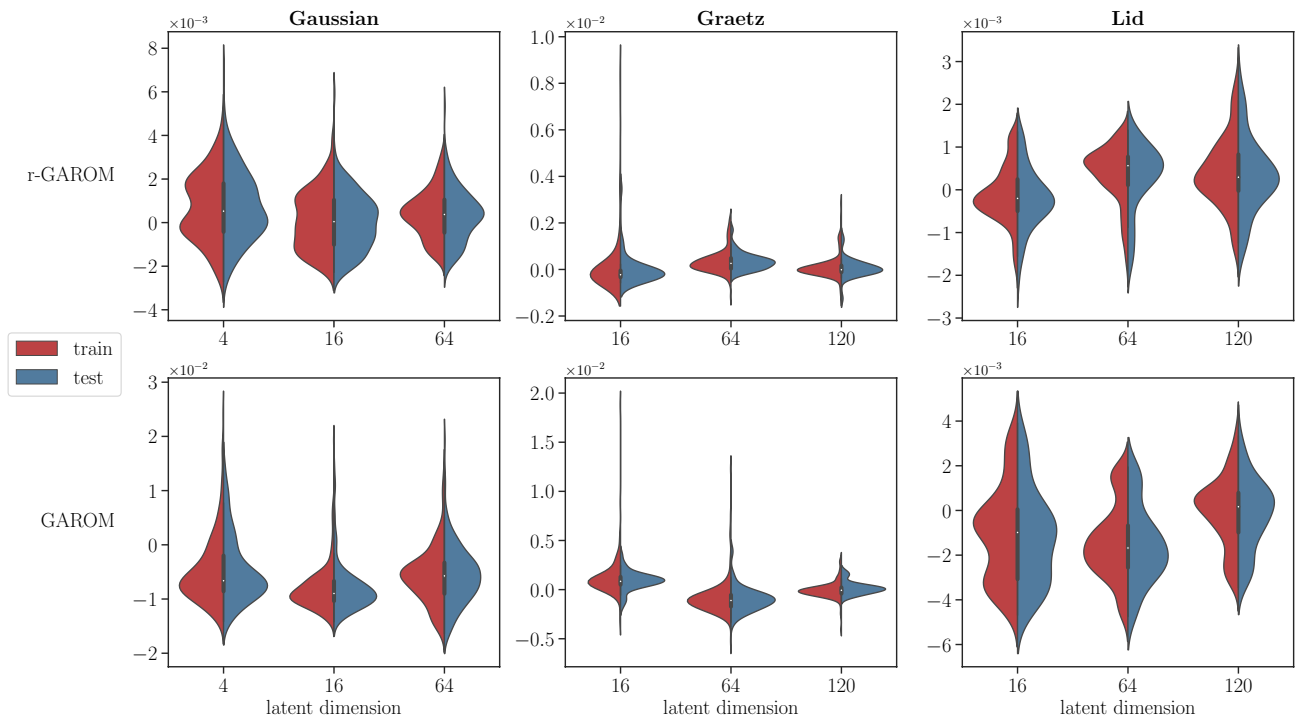


Figure 2. Distribution of the δ difference. The graph depicts, for each latent dimension, train (red) and test (blue) distribution of the δ . *Top:* r-GAROM model. *Bottom:* GAROM model.

centered around zero for both models, indicating a good high-fidelity solution reconstruction, as also shown in the previous Section. Finally, it is worth noticing that r-GAROM distributions are more close to zero than the unregularized counterpart, as a consequence of the regularizer term addition.

The last investigation regards the robustness of the training. It is well established in the deep learning community that generative adversarial networks tend to be unstable during training⁴⁵. Nevertheless, such behaviour is drastically reduced with BEGAN^{49,52}. In order to assess the convergence of the optimization procedure, we perform a statistical analysis studying the error trend during the training, using different random initialization for the weights of the network. Practically, 5 training loops are carried out changing the seed for the random number generator, and using Xavier weight initialization. Figure 3 depicts the mean error and the interval between the minimum and the maximum error across the 5 simulations. The plot indicates a stable convergence of r-GAROM, with sporadic perturbations in the statistical campaign. The only test case⁹ that is showing not a stable trend is the Gaussian one, but only when the latent dimension is 4: we suppose the latent dimension is not sufficiently big to represent the original problem, leading to an aleatory optimization. However, the l_2 relative error for the maximum discrepancy is in the same order of magnitude as the average result, showing that the model has still reached a good performance.

Discussion

In this study, we proposed GAROM, a generative adversarial reduced order modeling approach to tackle parametric PDE problems. The presented methodology is also extended to a regularized version (r-GAROM) in case the solution for the PDE is unique. The methodology is general, and applicable to a variety of ROM problems. The GAROM and r-GAROM methods are tested on three parametric benchmarks for different latent dimensions, showing very promising results in data-driven modeling. Among the deep learning approaches for ROM, r-GAROM outperforms in terms of accuracy in model prediction all of them in seven out of nine tests. Moreover, this approach seems to be more resilient to overfitting. With respect to POD, it emerges in these examples the POD models have a high difference between train and test errors when the POD is not able to efficiently capture the underlying behavior represented by the snapshot — e.g. due to nonlinearity. Differently, GAROM and its variation do not suffer from poor generalization, resulting in a more reliable model, despite the lack of precision in some tests. Surprisingly, it also shows any dependence between the l_2 relative mean error and the latent dimension we use in the discriminator, making it less sensitive to this parameter. It must be repeated that, with the GAROM approach, we are not projecting data on a reduced space, but the inner dimension is only used in the discriminator autoencoder. The adversarial approach combined with the proposed regularisation seems also helpful to stabilize the training in terms of loss trend, making such an approach practically less difficult to tune the hyper-parameters. Finally, the statistical approach allows the computation of the variance learned by the generator, making it possible to quantify the uncertainty of predictions applying different statistical strategies.

We identify multiple exciting research directions for the methodology. Firstly, improving the quality of the neural networks for the generator, discriminator, and conditioning mechanisms, and applying the methodology to more complex problems. As already stated, in the work we have used very simple architectures, which limits

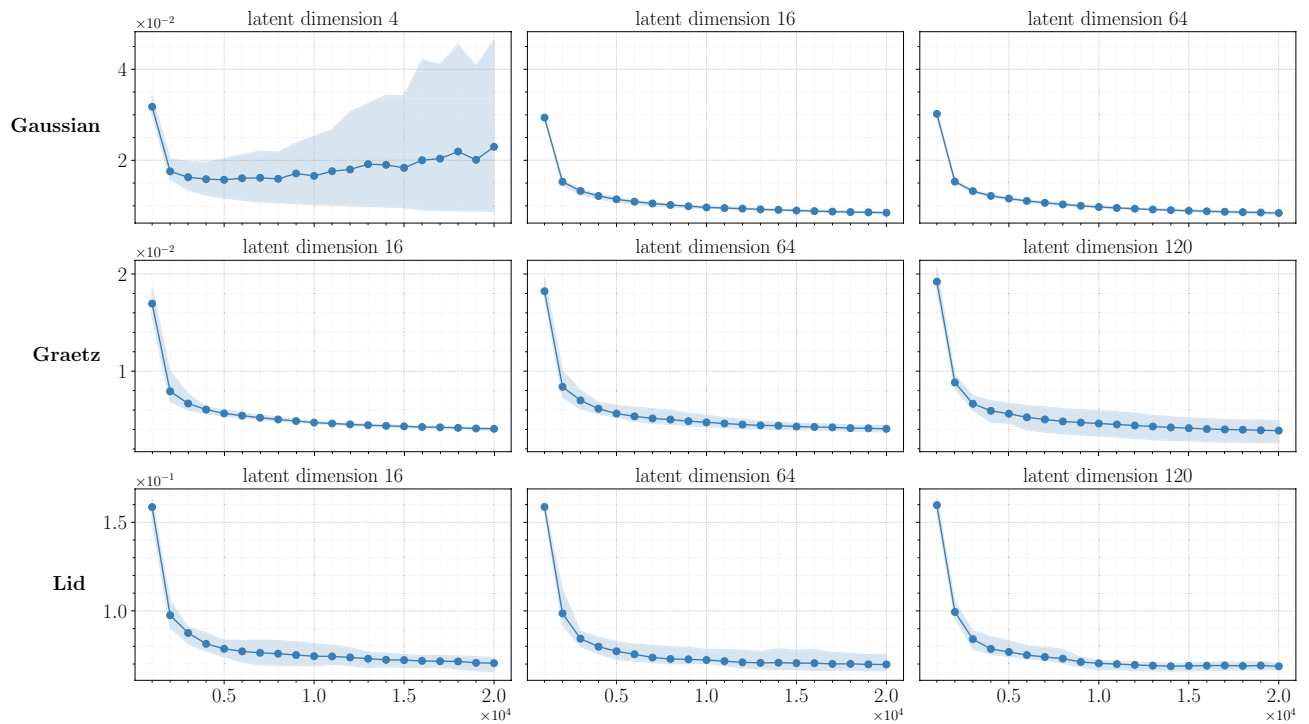


Figure 3. r-GAROM convergence graph in l_2 relative error for multiple training. The solid line indicates the average across all simulations. The shaded area represents the interval obtained by taking the maximum and minimum error across all simulations. The total number of training is 5.

the generation capability. As an example, the framework could be easily extended to time-dependent problems, employing LSTM⁵³ or transformer³⁶ networks for the conditioning mechanism. Alternatively, as also suggested in Ref.⁵², employing different architecture for the discriminator, e.g. U-net⁵⁴, could lead to accuracy improvement. Another possibility is to enhance the conditioning set with more information, e.g. by passing POD modes, or mesh information. Another very interesting extension of this work would be the continuous extension of the entire architecture. With the proposed architecture, both the networks' dimension scales with the number of components in the snapshots vector. To apply such an approach to very fine meshes, the computational requirement to train and infer the model would be huge and sometimes unfeasible. In these cases, a spatial continuous extension can lead to a wider diffusion, especially in more complex problems. Another possibility to mitigate the difficult application to finer meshes (and so snapshots with more degrees of freedom) is the employment of the architecture presented in⁵⁵ where the authors employ continuous convolutional neural networks over unstructured meshes. Finally, extending to a continuous setting might allow the use of our methodology to learn mappings between functional spaces in a discretization invariant manner, as done with Neural Operators^{56,57}.

In summary, the presented new methodology, combined with the different new research directions, paves the way to the application of GAROM in very complex settings, exhibiting great potential for future ROM developments in computational science.

Methods

Conditional boundary equilibrium GAN

Boundary equilibrium generative adversarial network (BEGAN)⁵² is an autoencoder based generative adversarial network. The BEGAN model consists of a generator network $G: \mathbb{R}^{N_z} \rightarrow \mathbb{R}^{N_u}$, generating a domain-specific sample in \mathbb{R}^{N_u} by passing random noise $\mathbf{z} \in \mathbb{R}^{N_z}$ following $p(\mathbf{z})$ distribution; and a discriminator network $D: \mathbb{R}^{N_u} \rightarrow \mathbb{R}^{N_u}$, which encodes and decodes real and generated samples. The core idea of BEGAN is utilizing the autoencoder reconstruction loss distribution derived from the Wasserstein distance⁵⁸ to approximate the data distribution. Specifically, the optimization is done with respect to the Wasserstein distance between the autoencoder reconstruction losses of real and generated samples. In addition, to prevent the imbalance of G and D , an equilibrium term is added in the objective function.

Formally, let the autoencoder reconstruction loss $\mathcal{L}: \mathbb{R}^{N_u} \rightarrow \mathbb{R}^+$ defined as:

$$\mathcal{L}(\mathbf{u}) = |\mathbf{u} - D(\mathbf{u})| \quad \text{where } \mathbf{u} \in \mathbb{R}^{N_u}, \quad (3)$$

with \mathbb{R}^{N_u} the space of real data samples. Thus, the BEGAN objective is:

$$\begin{cases} \mathcal{L}_D = \mathcal{L}(\mathbf{u}) - k_t \mathcal{L}(G(\mathbf{z})) & \text{minimize for } \theta_D \\ \mathcal{L}_G = \mathcal{L}(G(\mathbf{z})) & \text{minimize for } \theta_G \\ k_{t+1} = k_t + \lambda (\gamma \mathcal{L}(\mathbf{u}) - \mathcal{L}(G(\mathbf{z}))) & \text{for each training step } t, \end{cases} \quad (4)$$

with θ_G and θ_D the network parameters for the generator and discriminator, k_t a control term allowing the losses balance at each step t , and λ the learning rate for k_t . Finally, $\gamma \in [0, 1]$ is the diversity ratio defined as the ratio between the generated and real data reconstruction loss expected value:

$$\gamma = \frac{\mathbb{E}[\mathcal{L}(G(\mathbf{z}))]}{\mathbb{E}[\mathcal{L}(\mathbf{u})]}. \quad (5)$$

Therefore, the discriminator has two main goals: encoding and decoding real data, and discriminating real from generated data. The γ ratio is used to balance the two objectives. In fact, for lower values of γ , the discriminator focuses more heavily on autoencoding real data, leading to smaller data diversity. BEGAN has been proved to be effective for generating high resolution images, but it lacks to specifically condition the network.

Conditional BEGAN (cBEGAN)⁴⁹ has been introduced to overcome this issue. The cBEGAN objective is very similar to the one presented in Eq. (4), but the generator and discriminator are conditioned on conditioning variables $\mathbf{c} \in \mathbb{R}^{N_c}$. Hence, the cBEGAN objective becomes:

$$\begin{cases} \mathcal{L}_D = \mathcal{L}(\mathbf{u} | \mathbf{c}) - k_t \mathcal{L}(G(\mathbf{z} | \mathbf{c})) & \text{minimize for } \theta_D, \\ \mathcal{L}_G = \mathcal{L}(G(\mathbf{z} | \mathbf{c})) & \text{minimize for } \theta_G, \\ k_{t+1} = k_t + \lambda (\gamma \mathcal{L}(\mathbf{u} | \mathbf{c}) - \mathcal{L}(G(\mathbf{z} | \mathbf{c}))) & \text{for each training step } t, \end{cases} \quad (6)$$

with the autoencoder reconstruction loss:

$$\mathcal{L}(\mathbf{u} | \mathbf{c}) = \|\mathbf{u} - D(\mathbf{u} | \mathbf{c})\|. \quad (7)$$

In practice, the generator is conditioned by concatenating the random noise vector \mathbf{z} with the conditioning variable \mathbf{c} . The concatenation of the two represents the input for G . On the other hand, the discriminator is conditioned by concatenating the encoder output with the conditioning variable, before passing the concatenation to the decoder.

GAROM implementation details

The final objective of GAROM is to obtain a unique neural network which can perform ROM, while maintaining high reconstruction accuracy and generalization. We follow a *data-driven* approach for ROM, where a sample of high fidelity results from a domain-specific simulation is collected and used for training. Let $\mathbf{u} \in \mathbb{R}^{N_u}$ indicating the output of the real simulation, and $\mathbf{c} \in \mathbb{R}^{N_c}$ the variable affecting the simulation, e.g. parameters, time steps or boundary conditions. We name the variables \mathbf{c} as the conditioning variables, represented in this work by the free simulation parameters, as explained in section "Datasets description".

GAROM framework is based on an implicit generative modeling approach, specifically using cBEGAN. Indeed, cBEGAN allows the discriminator to learn a latent space for the real data manifold by autoencoding, forcing the generator to learn fundamental patterns of the data. Furthermore, having defined the discriminator as an autoencoder allows to impose constraints on the latent space, e.g. orthogonality, or to change the architecture to make it more robust, e.g. utilizing denoising⁵⁹ or contractive⁶⁰ regularization. These adjustments would not be possible with a vanilla GAN⁴⁵, where the discriminator outputs a real number in $[0, 1]$ only.

Formally, GAROM aims to approximate the density of the high fidelity solution data given the conditioning variables. In the present work the uniqueness of the high fidelity solutions is assumed, thus for a given parameter \mathbf{c} a unique solution \mathbf{u} is obtained. Following the cBEGAN framework, the GAROM objective is defined as:

$$\begin{cases} \mathcal{L}_D = \mathcal{L}(\mathbf{u} | \mathbf{c}) - k_t \mathcal{L}(G(\mathbf{z} | \mathbf{c})) & \text{minimize for } \theta_D \\ \mathcal{L}_G = \mathcal{L}(G(\mathbf{z} | \mathbf{c})) + \eta \|\mathbf{u} - G(\mathbf{z} | \mathbf{c})\| & \text{minimize for } \theta_G \\ k_{t+1} = k_t + \lambda (\gamma \mathcal{L}(\mathbf{u} | \mathbf{c}) - \mathcal{L}(G(\mathbf{z} | \mathbf{c}))) & \text{for each training step } t, \end{cases} \quad (8)$$

where $\eta \in \{0, 1\}$ ensure the absence or presence of the regularization parameter. Indeed, when $\eta = 0$ the GAROM objective is the same as the cBEGAN objective, which forces the generator to learn a distribution without any information of the uniqueness of the solution. On the contrary, setting $\eta = 1$ forces the generator to learn a unique distribution, which results in a shrinkage of the generator's variance, as well as better accuracy and generalization (see Section "Model variance"). We refer to this model as the regularized generative adversarial reduced order model, r-GAROM. We want to highlight that, on the contrary to many (classical or deep learning based) ROM frameworks, we do not project onto a lower dimensional space and, once the latent representation is learnt, train an interpolator. Instead, no interpolation is done since the conditioning is passed directly to the generator and discriminator. Hence, we perform only one neural network training with our methodology.

The (r-)GAROM architecture is depicted schematically in Fig. 4. Notice that the conditioning variables (the PDE parameters) are not concatenated directly, as done in cBEGAN, but passed to a domain-specific decoder for the generator f_τ , and discriminator g_ψ . This is done in order to pass more information to the generator similarly to⁶¹, and to avoid a high mismatch in the dimension of \mathbf{z} and \mathbf{c} . Finally, it is worth noticing that in general different conditioning variables can be applied, e.g. parameters, POD modes or coordinates information. As a consequence, a specific network is needed to encode the information into one high-dimensional vector. In the present work, only parametric problems are considered, and we leave for future works the investigation of highly effective conditioning mechanisms.

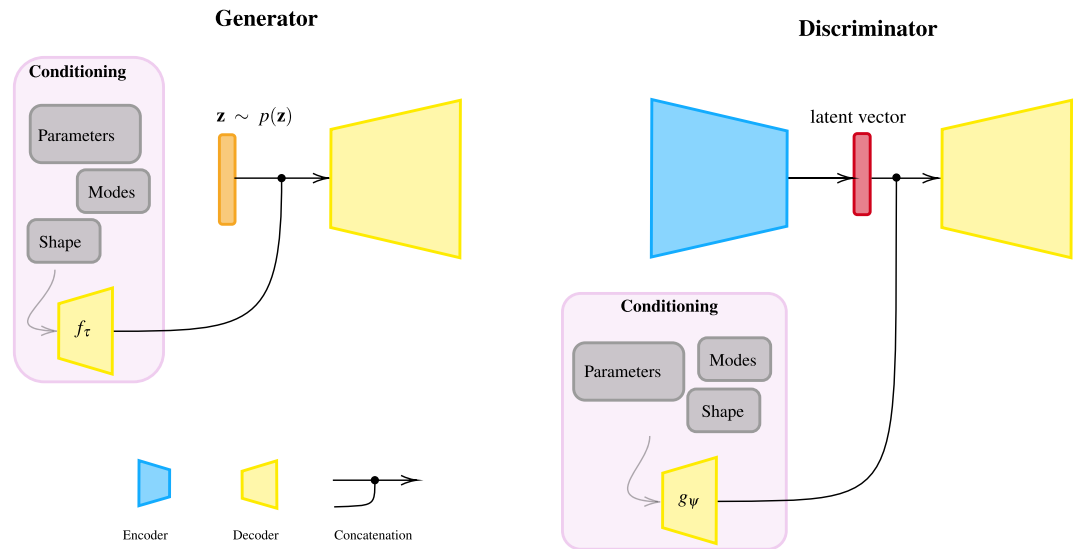


Figure 4. A schematic representation for GAROM generator and discriminator. The *Generator* input is the concatenation of random noise \mathbf{z} , and the conditioning representation $f_\tau(\mathbf{c})$. The *Discriminator* encodes the input obtaining a latent vector, which is concatenated with the conditioning representation $g_\psi(\mathbf{c})$ before it is passed to the decoder.

GAROM predictive distribution

Once the (r-)GAROM model is trained, a probabilistic ensemble for the output variable $\mathbf{u}(\mathbf{c})$ can be constructed leveraging $G(\mathbf{z} | \mathbf{c})$. Following a similar approach to⁶², we can predict the mean prediction $\hat{\mathbf{u}}$ and its variance $\hat{\sigma}^2$, for a new conditioning variable \mathbf{c}^* , by Monte Carlo sampling:

$$\hat{\mathbf{u}} = \mathbb{E}_{p_G}[\mathbf{u} | \mathbf{z}, \mathbf{c}^*] \approx \frac{1}{K} \sum_{k=1}^K G(\mathbf{z}_k | \mathbf{c}^*),$$

$$\hat{\sigma}^2 = \text{Var}_{p_G}[\mathbf{u} | \mathbf{z}, \mathbf{c}^*] \approx \frac{1}{K-1} \sum_{k=1}^K [\hat{\mathbf{u}} - G(\mathbf{z}_k | \mathbf{c}^*)]^2.$$
(9)

In particular, p_G is the implicit distribution learned by the generator via adversarial optimization, K is the number of Monte Carlo samples used for approximating the distribution moments, and \mathbf{z}_k are the samples from the latent distribution $p(\mathbf{z})$.

Uncertainty quantification strategies

Quantifying the uncertainty of a prediction is a common problem for data-driven reduced order models⁶³. Exploiting the predictive distribution of (r-)GAROM one can obtain moment estimates by Monte-Carlo sampling, as discussed in section "GAROM predictive distribution". These estimates can be used during inference to compute bounds in probability of the prediction error, i.e. the difference between real and average (r-)GAROM solution. As an example, employing the Markov inequality we can find the following bound for any threshold $a > 0$:

$$\text{Prob}[(\mathbf{u}_{\text{true}}(\mathbf{c}) - \hat{\mathbf{u}}(\mathbf{c}))^2 \geq a] \leq \frac{1}{a} \left\{ \mathbb{E}_{\text{Prob}} \left[\mathbb{E}_{p_G} [(\mathbf{u}_{\text{true}}(\mathbf{c}) - \mathbf{u}(\mathbf{c}))^2] - \hat{\sigma}^2(\mathbf{c}) \right] \right\},$$
(10)

where Prob is the probability distribution for \mathbf{c} , e.g. random uniform. In practice, the right hand side of Eq. (10) can be estimated, with sufficient statistics, after training by Monte Carlo approximation on the training dataset, resulting in an estimate in probability for the average prediction.

A more tight bound for each parameter \mathbf{c} , can be obtained assuming an approximation of the distribution p_G , and applying the theory of confidence interval. As an example, assuming p_G is well approximated using a normal distribution $\mathcal{N}(\hat{\mathbf{u}}, \hat{\sigma}^2)$, one could estimate the prediction uncertainty in a similar approach to the one adopted by⁶⁴. While the first bound based on Markov inequality is certain, the latter one uses an approximation of the distribution. Nonetheless, as also evidenced in⁶⁴, these kinds of approximations (problem-specific) can obtain very good estimates.

Datasets description

This section is dedicated to introduce the numerical benchmarks applied for testing GAROM and comparing it to the baseline models presented in section "Baseline models". For notation consistency, \mathbf{c} indicates the vector of free parameters while \mathbf{u} the simulation solution.

Parametric Gaussian

The first test case is a simple problem representing a Gaussian function moving in a domain $\Omega = [-1, 1] \times [-1, 1]$. The high fidelity function $u : \Omega \rightarrow \mathbb{R}$ is defined as:

$$u(x, y) = e^{-[(x-c_1)^2+(y-c_2)^2]}, \quad (11)$$

with $\mathbf{c} = [c_1, c_2]$ the centers of the Gaussian. Practically, u is evaluated on 900 points uniformly randomly chosen in Ω for a fixed \mathbf{c} . The evaluation results are flattened (row-major order) in a one-dimensional vector of size 900, namely \mathbf{u} . The dataset is composed by $N = 400$ instances \mathbf{c}_i uniformly randomly chosen in Ω , resulting in N high fidelity solutions \mathbf{u}_i .

Graetz problem

The second test case deals with the Graetz-Poiseuille problem, which models forced heat convection in a channel. The problem is characterized by two parameters: c_1 controlling the length of the domain, and c_2 the Péclet number, which takes into account the heat transfer in the domain. The full domain is $\Omega(c_1) = [0, 1 + c_1] \times [0, 1]$, with $\mathbf{c} = [c_1, c_2] \in [0.1, 10] \times [0.01, 10]$. The high fidelity solution $u : \Omega(c_1) \rightarrow \mathbb{R}$ is obtained by solving:

$$\begin{cases} -\Delta u(x, y; \mathbf{c}) + c_2 y(1-y) \frac{\partial}{\partial x} u(x, y; \mathbf{c}) = 0 & (x, y) \in \overset{\circ}{\Omega}(c_1) \\ u(x = 0, y; \mathbf{c}) = 0 & y \in [0, 1] \\ u(x, y = 0; \mathbf{c}) = 0 & x \in [0, 1] \\ u(x, y = 1; \mathbf{c}) = 0 & x \in [0, 1] \\ u(x, y = 0; \mathbf{c}) = 1 & x \in [1, 1 + c_1] \\ u(x, y = 1; \mathbf{c}) = 1 & x \in [1, 1 + c_1] \\ \partial_{\mathbf{n}} u(x = 1 + c_1, y) = 0 & y \in [0, 1] \end{cases} \quad (12)$$

where Δ indicates the Laplacian operator, and $\partial_{\mathbf{n}}$ the normal derivative. The high-fidelity solutions are numerically computed by finite elements following the work⁶, using a mesh of 5160 points. Specifically, $N = 200$ instances of \mathbf{c}_i are uniformly randomly chosen in $\Omega(c_1)$, resulting in N high fidelity solutions \mathbf{u}_i arranged as a vector (row-major order).

Lid cavity

The last test case is the famous Lid Cavity problem, modeling isothermal, incompressible flow in a two-dimensional square domain. The domain is composed of a top wall moving along the horizontal axis, while the other three walls are stationary. At low Reynolds number (Re) the flow is laminar, but it becomes turbulent when increasing the Re. A complete mathematical formulation of the problem is found in⁶⁵. The full domain is $\Omega = [d, d] \times [d, d]$ with $d = 0.1m$ the domain size. The free parameter c is the magnitude of the velocity of the top wall, measured in ms^{-1} . The Reynolds number is given by:

$$Re = \frac{dc}{\nu}, \quad (13)$$

with the kinematic viscosity $\nu = 10^{-5}m^2s^{-1}$. To compute the high fidelity solution, we simulate for 5s using a time step of 0.0001 keeping only the last time-step simulation. The mesh is composed of hexahedral cells of $70 \times 70 \times 1$ number of cell points in each direction. We use a cell-center finite volume scheme. The dataset is composed by $N = 300$ instances of $c \in [0.01, 1]$ evenly spaced, resulting in N high fidelity solutions arranged as a vector (row-major order).

Baseline models

We compared the performances of the (r-)GAROM approach against some of the more consolidated data-driven frameworks in ROM community⁶³. Such models rely on machine learning and/or linear algebra techniques, resulting on different approaches from the algorithmic perspectives. In such way, we aim for the most possible fair comparison, distinguishing the GAROM advantages and disadvantages with respect to the state-of-the-art in different possible operating contexts.

The baseline comparison models are:

- Proper Orthogonal Decomposition with Radial Basis Function interpolation (POD-RBF)⁶⁶,
- Proper Orthogonal Decomposition with Artificial Neural Network interpolation (POD-NN)²⁰,
- Autoencoders with Radial Basis Function interpolation (AE-RBF)⁵¹,
- Autoencoders with Artificial Neural Network interpolation (AE-NN)⁶⁷⁻⁶⁹.
- Conditional GAN (cGAN)⁷⁰
- Regularized vanilla conditional GAN (r-cGAN)⁴⁶
- Deep Operator Networks (DeepONet)²⁴
- Deep Operator Networks with Non-Linear Manifold Decoder (NOMAD)²⁵

In a few words, POD-RBF, POD-NN, AE-RBF, and AE-NN use proper orthogonal decomposition and an autoencoder for dimensionality reduction, while radial basis function and artificial neural network for approximating the reduced parametric solution manifold. The POD is performed by truncated singular value decomposition with a variable rank (latent dimension in the manuscript), while for radial basis function interpolation, we employed a Gaussian Kernel with length scale equal to 1. Differently, DeepONet and NOMAD are neural operator networks, mapping directly the parameters and field coordinates to the field solution on the specified coordinates. Finally, we also present two vanilla GAN approaches: cGAN and r-cGAN, where at the latter we add an L1 penalty loss to the generator network⁴⁶.

For the neural networks-based approaches we performed hyperparameter optimization to find the architecture with lower training mean square error. The optimization was carried out with RayTune⁷¹. For the specifics on network parameters and training procedures see the Supplementary Information.

Software details

The PyTorch software⁷² is used to implement the GAROM model due to its versatility and its wide used in the deep learning community. The EZyRB software⁷³ and PINA software⁷⁴ are used for performing the comparison between GAROM and state-of-the-art techniques in Section numerics. Openfoam⁷⁵ and FeniCSx⁷⁶ are used for the numerical simulations. An implementation of GAROM is available in PINA⁷⁴, while the datasets used are available in the Smithers package on GitHub at <https://github.com/mathLab/Smithers>.

Training setup and model architecture

An NVIDIA Quadro RTX 4000 GPU was used to train the GAROM models, while the GAROM comparison methods are trained on Intel CPUs. In the Graetz and Lid Cavity experiments, the GAROM and baseline models are all trained on a dataset containing 80% of the total number of high fidelity simulation instances chosen evenly spaced, and tested on the remaining 20%. On the contrary, for the Gaussian test case, 60% of the high fidelity instances chosen evenly spaced are used for training, and the remaining 40% for testing. This choice is due to the higher number of instances with respect to Lid and Graetz test cases.

The GAROM models are all trained for 20000 epochs, with Adam optimizer using a learning rate of 0.001 for both the discriminator and generator. The GAROM training times for the benchmarks are approximate of 45 minutes for the parametric Gaussian, 1 hours and 30 minutes for Graetz, and 2 hours for the Lid Cavity, independently of the discriminator latent dimension. In order to fit all data in GPU we used a batch size of 4 for all tests, where the value was obtained by hyper-parameter optimization using RayTune⁷¹ choosing from {4, 8, 16, 24, 64}. The noise vector \mathbf{z} are uniform random samples in $[-1, 1]$ of dimension 13 for the Gaussian test, 8 for the Graetz test, and 12 for the Lid Cavity test. Also, the noise vector dimension was tuned with RayTune, choosing a random integer in the range [8, 16]. Furthermore, the λ and γ parameters of Eq. (8) are set to 0.001 and 0.3 respectively, as suggested in⁵². The generator, discriminator, generator conditional encoder, and discriminator conditional encoder networks are reported in Supplementary information.

Data availability

Data to reproduce the code is available on GitHub at <https://github.com/mathLab/Smithers>. The GAROM solver is implemented in PINA⁷⁴ on GitHub at <https://github.com/mathLab/PINA>. Supplementary Information.

Received: 12 July 2023; Accepted: 8 February 2024

Published online: 15 February 2024

References

1. Strauss, W. A. *Partial Differential Equations: An Introduction* (Wiley, 2007).
2. Bauer, P., Thorpe, A. & Brunet, G. The quiet revolution of numerical weather prediction. *Nature* **525**, 47–55 (2015).
3. Lomax, H., Pulliam, T. H., Zingg, D. W., Pulliam, T. H. & Zingg, D. W. *Fundamentals of Computational Fluid Dynamics* Vol. 246 (Springer, 2001).
4. Morton, K. W. & Mayers, D. F. *Numerical Solution of Partial Differential Equations: An Introduction* (Cambridge University Press, 2005).
5. Butcher, J. C. *Numerical Methods for Ordinary Differential Equations* (Wiley, 2016).
6. Hesthaven, J. S. *et al. Certified Reduced Basis Methods for Parametrized Partial Differential Equations* Vol. 590 (Springer, 2016).
7. Lassila, T., Manzoni, A., Quarteroni, A. & Rozza, G. Model order reduction in fluid dynamics: challenges and perspectives. In *Reduced Order Methods for modeling and computational reduction*, 235–273 (2014).
8. Lucia, D. J., Beran, P. S. & Silva, W. A. Reduced-order modeling: New approaches for computational physics. *Prog. Aerosp. Sci.* **40**, 51–117 (2004).
9. Kadeethum, T. *et al. Enhancing high-fidelity nonlinear solver with reduced order model. Sci. Rep.* **12**, 20229 (2022).
10. Joshi, A. *et al. Generative models for solving nonlinear partial differential equations. In eprint Proc. of NeurIPS Workshop on ML for Physics* (2019).
11. Berkooz, G., Holmes, P. & Lumley, J. L. The proper orthogonal decomposition in the analysis of turbulent flows. *Annu. Rev. Fluid Mech.* **25**, 539–575 (1993).
12. Lazzaro, D. & Montefusco, L. B. Radial basis functions for the multivariate interpolation of large scattered data sets. *J. Comput. Appl. Math.* **140**, 521–536 (2002).
13. Csala, H., Dawson, S. & Arzani, A. Comparing different nonlinear dimensionality reduction techniques for data-driven unsteady fluid flow modeling. *Phys. Fluids* **34**, 117119 (2022).
14. Kadeethum, T. *et al. Reduced order modeling for flow and transport problems with Barlow Twins self-supervised learning. Sci. Rep.* **12**, 20654 (2022).
15. Maulik, R., Lusch, B. & Balaprakash, P. Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders. *Phys. Fluids* **33**, 037106 (2021).
16. Eivazi, H., Veisi, H., Naderi, M. H. & Esfahanian, V. Deep neural networks for nonlinear model order reduction of unsteady flows. *Phys. Fluids* **32**, 105104 (2020).

17. Demo, N., Tezzele, M. & Rozza, G. A deepnet multi-fidelity approach for residual learning in reduced order modeling. Preprint at <http://arxiv.org/abs/2302.12682> (2023).
18. Mohan, A. T. & Gaitonde, D. V. A deep learning based approach to reduced order modeling for turbulent flow control using LSTM neural networks. Preprint at <http://arxiv.org/abs/2302.12682> (2018).
19. Gonnella, I. C., Hess, M. W., Stabile, G. & Rozza, G. A two stages Deep Learning Architecture for Model Reduction of Parametric Time-Dependent Problems. Preprint at <http://arxiv.org/abs/2301.09926> (2023).
20. Hesthaven, J. S. & Ubbiali, S. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *J. Comput. Phys.* **363**, 55–78 (2018).
21. Romor, F., Stabile, G. & Rozza, G. Non-linear manifold ROM with convolutional autoencoders and reduced over-collocation method. Preprint at <http://arxiv.org/abs/2203.00360> (2022).
22. Fresca, S., Dede, L. & Manzoni, A. A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs. *J. Sci. Comput.* **87**, 1–36 (2021).
23. Gruber, A., Gunzburger, M., Ju, L. & Wang, Z. A comparison of neural network architectures for data-driven reduced-order modeling. *Comput. Methods Appl. Mech. Eng.* **393**, 114764 (2022).
24. Lu, L., Jin, P., Pang, G., Zhang, Z. & Karniadakis, G. E. Learning nonlinear operators via deepnet based on the universal approximation theorem of operators. *Nat. Mach. Intell.* **3**, 218–229 (2021).
25. Seidman, J., Kissas, G., Perdikaris, P. & Pappas, G. J. Nomad: Nonlinear manifold decoders for operator learning. *Adv. Neural Inf. Process. Syst.* **35**, 5601–5613 (2022).
26. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* (MIT press, 2016).
27. Tomczak, J. M. *Deep Generative Modeling* (Springer, 2022).
28. Bishop, C. M. Latent variable models. *Learn. Graph. Models* **371**, 371–403 (1998).
29. Rombach, R., Blattmann, A., Lorenz, D., Esser, P. & Ommer, B. High-resolution image synthesis with latent diffusion models. In eprint Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 10684–10695 (2022).
30. Ilse, M., Tomczak, J. M., Louizos, C. & Welling, M. Diva: Domain invariant variational autoencoders. In *Medical Imaging with Deep Learning* (eds Ilse, M. et al.) 322–348 (PMLR, 2020).
31. Oord, A. v. d. et al. Wavenet: A generative model for raw audio. Preprint at <http://arxiv.org/abs/1609.03499> (2016).
32. Razavi, A., Van den Oord, A. & Vinyals, O. Generating diverse high-fidelity images with VQ-VAE-2. *Adv. Neural Inf. Process. Syst.*, **32** (2019).
33. Kingma, D. P. & Welling, M. Auto-Encoding Variational Bayes. Preprint at <http://arxiv.org/abs/1312.6114> (2013).
34. Solera-Rico, A. et al. β -Variational autoencoders and transformers for reduced-order modelling of fluid flows. Preprint at <http://arxiv.org/abs/2304.03571> (2023).
35. Eivazi, H., Le Clainche, S., Hoyas, S. & Vinuesa, R. Towards extraction of orthogonal and parsimonious non-linear modes from turbulent flows. *Expert Syst. Appl.* **202**, 117038 (2022).
36. Vaswani, A. et al. Attention is all you need. *Adv. Neural Inf. Process. Syst.*, **30** (2017).
37. Rezende, D. J. & Viola, F. Taming vaes. Preprint at <http://arxiv.org/abs/1810.00597> (2018).
38. Malik, S., Anwar, U., Ahmed, A. & Aghasi, A. Learning to solve differential equations across initial conditions. Preprint at <http://arxiv.org/abs/2003.12159> (2020).
39. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707 (2019).
40. Daw, A., Maruf, M. & Karpatne, A. PID-GAN: A GAN Framework based on a Physics-informed Discriminator for Uncertainty Quantification with Physics. In Proc. of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 237–247 (2021).
41. Randle, D., Protopapas, P. & Sondak, D. Unsupervised learning of solutions to differential equations with generative adversarial networks. Preprint at <http://arxiv.org/abs/2007.11133> (2020).
42. Yang, L. et al. Highly-scalable, physics-informed GANs for learning solutions of stochastic PDEs. In 2019 IEEE/ACM Third Workshop on Deep Learning on Supercomputers (DLS), 1–11 (IEEE, 2019).
43. Kadeethum, T. et al. A framework for data-driven solution and parameter estimation of PDEs using conditional generative adversarial networks. *Nat. Comput. Sci.* **1**, 819–829 (2021).
44. Kadeethum, T. et al. Continuous conditional generative adversarial networks for data-driven solutions of poroelasticity with heterogeneous material properties. *Comput. Geosci.* **167**, 105212 (2022).
45. Goodfellow, I. et al. Generative adversarial networks. *Commun. ACM* **63**, 139–144 (2020).
46. Kemna, M., Heinlein, A. & Vuik, C. Reduced order fluid modeling with generative adversarial networks. *PAMM* **23**, e202200241 (2023).
47. Silva, V. L. S., Heaney, C. E. & Pain, C. C. Generative network-based reduced-order model for prediction, data assimilation and uncertainty quantification. In eprint LatinX in AI Workshop at ICML 2023 (Regular Deadline) (2023).
48. He, W. & Jiang, Z. A Survey on Uncertainty Quantification Methods for Deep Neural Networks: An Uncertainty Source Perspective. Preprint at <http://arxiv.org/abs/2302.13425> (2023).
49. Marzouk, A., Barros, P., Eppe, M. & Wernter, S. The Conditional Boundary Equilibrium Generative Adversarial Network and its Application to Facial Attributes. In 2019 International Joint Conference on Neural Networks (IJCNN), 1–7 (IEEE, 2019).
50. Dutta, S., Rivera-Casillas, P., Styles, B. & Farthing, M. W. Reduced order modeling using advection-aware autoencoders. *Math. Comput. Appl.* **27**, 34 (2022).
51. Lee, K. & Carlberg, K. T. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *J. Comput. Phys.* **404**, 108973 (2020).
52. Berthelot, D., Schumm, T. & Metz, L. Began: Boundary equilibrium generative adversarial networks. Preprint at <http://arxiv.org/abs/1703.10717> (2017).
53. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997).
54. Ronneberger, O., Fischer, P. & Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5–9, 2015, Proceedings, Part III 18, 234–241 (Springer, 2015).
55. Coscia, D., Meneghetti, L., Demo, N., Stabile, G. & Rozza, G. A continuous convolutional trainable filter for modelling unstructured data. *Comput. Mech.* **72**, 1–13 (2023).
56. Rahman, M. A., Florez, M. A., Anandkumar, A., Ross, Z. E. & Azzizadenesheli, K. Generative adversarial neural operators. Preprint at <http://arxiv.org/abs/2205.03017> (2022).
57. Seidman, J. H., Kissas, G., Pappas, G. J. & Perdikaris, P. Variational autoencoding neural operators. Preprint at <http://arxiv.org/abs/2302.10351> (2023).
58. Arjovsky, M., Chintala, S. & Bottou, L. Wasserstein generative adversarial networks. In International conference on machine learning, 214–223 (PMLR, 2017).
59. Vincent, P., Larochelle, H., Bengio, Y. & Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In Proc. of the 25th international conference on Machine learning, 1096–1103 (2008).
60. Rifai, S., Vincent, P., Muller, X., Glorot, X. & Bengio, Y. Contractive auto-encoders: Explicit invariance during feature extraction. In Proc. of the 28th International Conference on International Conference on Machine Learning, 833–840 (2011).

61. Reed, S. et al. Generative adversarial text to image synthesis. In International Conference on Machine Learning, 1060–1069 (PMLR, 2016).
62. Yang, Y. & Perdikaris, P. Adversarial uncertainty quantification in physics-informed neural networks. *J. Comput. Phys.* **394**, 136–152 (2019).
63. Rozza, G., Stabile, G. & Ballarin, F. *Advanced Reduced Order Methods and Applications in Computational Fluid Dynamics* (SIAM, 2022).
64. Gundersen, K., Oleynik, A., Blaser, N. & Alendal, G. Semi-conditional variational auto-encoder for flow reconstruction and uncertainty quantification from limited observations. *Phys. Fluids* **33**, 017119 (2021).
65. Schreiber, R. & Keller, H. B. Driven cavity flows by efficient numerical techniques. *J. Comput. Phys.* **49**, 310–333 (1983).
66. Salmoiraghi, F., Scardigli, A., Telib, H. & Rozza, G. Free-form deformation, mesh morphing and reduced-order methods: Enablers for efficient aerodynamic shape optimisation. *Int. J. Comput. Fluid Dyn.* **32**, 233–247 (2018).
67. Ivagnes, A., Demo, N. & Rozza, G. Towards a machine learning pipeline in reduced order modelling for inverse problems: Neural networks for boundary parametrization, dimensionality reduction and solution manifold approximation. *J. Sci. Comput.* <https://doi.org/10.1007/s10915-023-02142-4> (2023).
68. Hasegawa, K., Fukami, K., Murata, T. & Fukagata, K. Machine-learning-based reduced-order modeling for unsteady flows around bluff bodies of various shapes. *Theor. Comput. Fluid Dyn.* **34**, 367–383 (2020).
69. Gonzalez, F. J. & Balajewicz, M. Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems. Preprint at <http://arxiv.org/abs/1808.01346> (2018).
70. Mirza, M. & Osindero, S. Conditional generative adversarial nets. Preprint at <http://arxiv.org/abs/1411.1784> (2014).
71. Liaw, R. et al. Tune: A research platform for distributed model selection and training. Preprint at <http://arxiv.org/abs/1807.05118> (2018).
72. Paszke, A. et al. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.*, **32** (2019).
73. Demo, N., Tezzele, M. & Rozza, G. EZyRB: Easy Reduced Basis method. *J. Open Source Softw.* **3**, 661. <https://doi.org/10.21105/joss.00661> (2018).
74. Coscia, D., Ivagnes, A., Demo, N. & Rozza, G. Physics-informed neural networks for advanced modeling. *J. Open Source Softw.* **8**, 5352. <https://doi.org/10.21105/joss.05352> (2023).
75. OpenFOAM. <https://www.openfoam.com/>. (Accessed 07 September 2022).
76. Logg, A. & Mardal, K.-A. *Automated Solution of Differential Equations by the Finite Element Method* (Springer, 2012).

Acknowledgements

This work is partially supported by European Union Funding for Research and Innovation - Horizon 2020 Program - in the framework of European Research Council Executive Agency: H2020 ERC CoG 2015 AROMA-CFD project 681447 “Advanced Reduced Order Methods with Applications in Computational Fluid Dynamics” P.I. Professor Gianluigi Rozza, by European Union Funding for Research and Innovation — Horizon Europe Program — in the framework of European Research Council Executive Agency: ERC POC 2022 ARGOS project 101069319 “Advanced Reduced order modellinG: Online computational web server for complex parametric Systems” P.I. Professor Gianluigi Rozza, by European High-Performance Computing Joint Undertaking project Eflows4HPC GA N. 955558, by PRIN “Numerical Analysis for Full and Reduced Order Methods for Partial Differential Equations” (NA-FROM-PDEs) project.

Author contributions

D.C., N.D. and G.R. conceived the experiment(s), D.C. and N.D. conducted the experiment(s), and analysed the results. All authors reviewed the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at <https://doi.org/10.1038/s41598-024-54067-z>.

Correspondence and requests for materials should be addressed to G.R.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher’s note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

© The Author(s) 2024