



Generative model-based framework for parameter estimation and uncertainty quantification applied to a compartmental model in epidemiology

Vinicius L.S. Silva^{a,*}, Claire E. Heaney^{a,b}, Nenko Nenov^a, Christopher C. Pain^{a,b}

^a Applied Modelling & Computation Group, Imperial College London, UK

^b Centre for AI Physics Modelling, Imperial-X, Imperial College London, UK

ARTICLE INFO

Keywords:

Generative model
Generative adversarial network
Reduced-order model
Data assimilation
Uncertainty Quantification
Compartmental model
Epidemiology

ABSTRACT

We propose a new method in which a generative network (GN) set within a reduced-order model (ROM) framework is used to solve inverse problems for partial differential equations (PDE). The aim is to match available measurements and estimate the corresponding uncertainties associated with the states and parameters of a numerical physical simulation. We train the GN using only unconditional simulations of the discretized PDE model. A GN is used here to exploit the ability of these methods to generate realistic outputs from complex probability distributions, such as the ones that represent the possible states and parameters of a physical problem. We compare the proposed method with the gold standard Markov chain Monte Carlo (MCMC). Additionally, we suggest the use of a novel type of time-stepping regularization to improve the representativeness of the physical solution, and we present a new way of evaluating the GN training, taking advantage of the real/generated sample structure. We apply the proposed approaches to a spatio-temporal compartmental model in epidemiology. The results show that the proposed GN-based ROM can efficiently quantify uncertainty and accurately match the measurements and the gold standard. This is achieved using only a limited number of unconditional simulations from the full-order numerical PDE model (40 simulations). The GN-based ROM operates 60 times faster than the gold standard (MCMC), while producing uncertainties that closely match those generated by the MCMC approach. The proposed method is a general framework for quantifying uncertainties in numerical physical simulations and is not restricted to the specific physics of this application.

1. Introduction

Complex physics and engineering systems are usually described in terms of partial differential equations (PDE) that, for most problems of practical interest, cannot be solved analytically. Then it is necessary to use numerical methods to solve the governing equations. The discretization of these equations is usually performed using finite difference, finite volume, finite element or a combination of these procedures [1,2]. Nonetheless, these methods need a large number of degrees of freedom to solve the PDEs accurately. This fact can generate prohibitively expensive simulations in terms of computational time and memory demand. Furthermore, these models are built on limited information, which makes their predictions uncertain. The actual values of the model states and parameters are not known, and usually measurements are sparse in space and/or time. Therefore, it is necessary to assimilate observed data (calibrate model states and parameters in order to generate results that match the measurements) and formally

propagate the uncertainties through the forward numerical simulator. In this context, reduced-order models (ROM) are computationally appealing and have been attracting significant attention over recent decades [3–6]. The aim of a ROM is to reduce the computational burden of numerical simulations by creating a low-dimensional representation of a high-dimensional model or discretized system [7–9].

In order to quantify the uncertainty in the states and parameters of a numerical PDE simulation, one requires multiple random models that match or are conditional to measurements. After simulating the conditional models, an empirical distribution of the variables of interest can be obtained [10–12]. The validity of the uncertainty quantification depends on the quality of the generated conditional simulations. Nonetheless, it is often difficult and computationally expensive to generate a single conditional model (that honours the measurements), suggesting that the task of quantifying uncertainty must be even more

* Corresponding author.

E-mail address: v.santos-silva19@imperial.ac.uk (V.L.S. Silva).

<https://doi.org/10.1016/j.jocs.2024.102451>

Received 4 October 2023; Received in revised form 23 July 2024; Accepted 24 September 2024

Available online 15 October 2024

1877-7503/© 2024 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

difficult [10,13–15]. It is unfeasible to use methods such as rejection sampling (RS) and Markov chain Monte Carlo (MCMC) to propagate uncertainty through most practical PDE simulations due to their computational cost [10,13,16–18]. Therefore, approximate methods need to be used. Among them, Liu et al. [10] and Emerick and Reynolds [13] showed that the randomized maximum likelihood (RML) [19,20], also called randomize-then-optimize (RTO) [21], performed better than other approximate methods.

In this paper, we propose a new method inspired by the RML (or RTO) in which a generative network (GN) within a ROM framework is used to quantify the uncertainty of a numerical PDE simulation in the presence of measurements. The ROM uses a low-dimensional space for the spatial distribution of the simulation states. Then the GN is used to learn the evolution of the low-dimensional states over time. The GN is trained using only unconditional simulations of the full-order numerical model. After training, the GN-based ROM can be used to predict the evolution of the spatial distribution of the simulation states and observed data can be assimilated. Here, we describe the process required in order to quantify uncertainty, during which no additional simulations of the full-order numerical model are required. We apply these methods to estimate the uncertainty in the states and parameters of a spatio-temporal compartmental model in epidemiology, that was constructed in order to represent the spread of COVID-19 in an idealized town.

In terms of novelty, we are proposing a new method to quantify uncertainty of numerical PDE simulations conditional to measurements. Furthermore, we present a novel type of time-step regularization to improve physical representation, and we propose a new way of evaluating the GN training based on the characteristics of the problem. We list these novelties and contributions of our work as follows:

- We propose a novel approach for quantifying uncertainty in the states and parameters of numerical PDE simulations, taking into account observed data. We compare it with the gold standard MCMC, and show that it can generate predictions and uncertainties that reasonably match the MCMC, but with orders of magnitude less computational time.
- We demonstrate how a new time-stepping regularization can be used to improve the results of already known GN-based ROM approaches, and the new proposed uncertainty quantification method.
- We present a new way of evaluating the GN training, using the fact that for numerical PDE simulations the real/generated samples represent a spatio-temporal map with known structure.
- The proposed method is not limited to the underlying physics of the compartmental model in epidemiology, it is a general framework for quantifying uncertainties of numerical PDE simulations.

The remaining of this paper is structured as follows. In Section 2, we present some related work. Section 3 details the spatio-temporal compartmental model in epidemiology used as a test case. Following that, the proposed framework is described in Section 4 and the results and discussion are presented in Section 5. Finally, concluding remarks are provided in Section 7. The source code, data, hardware configuration, used in this work are available at <https://github.com/viluz/gan>.

2. Related work

Machine learning for solving PDEs

Several works have used machine learning to solve PDEs, such as Lagaris et al. [22], Lopez et al. [23], Guo et al. [24], Raissi et al. [25], Long et al. [26], Sanchez-Gonzalez et al. [27], Pfaff et al. [28], Wang et al. [29], Chen et al. [30], Li et al. [31]. Among them, GNs have been demonstrating promising results. GNs have been used to predict spatio-temporal solutions for super-resolution fluid flow [32], carbon capture [33], incoming waves from Hokkaido tsunami [34], the

spread of COVID-19 [35] and short-timescale weather forecast [36]. All of the previous works tackled the problem of forward simulations in time and/or space. In addition to predicting in time, for many practical applications, the ability to assimilate observed data (solve an inverse problem) and generate the corresponding uncertainties is also highly desirable.

Reduced order modelling

Whilst retaining much of the accuracy of full-order models, ROMs have been used successfully in several fields to reduce the computation time. Non-intrusive ROMs (NIROMs) are particularly flexible, as they avoid the need to make modifications to the original source code [37–39]. To produce a NIROM, first, dimensionality reduction techniques are applied to the solutions from the original full-order model (known as snapshots) in order to find a low-dimensional representation. Proper orthogonal decomposition (POD) [40], autoencoders, or a combination of both [41,42] are commonly used for this purpose. The second stage involves interpolating the reduced variables in order to approximate the evolution of the model. Recently, a variety of machine learning methods have been used for this task, including multi-layer perceptrons [38], cluster analysis [43], Long Short-Term Memory (LSTM) [44, 45], Gaussian Process Regression [46], and generative networks [35, 47–49].

Data assimilation and uncertainty quantification

Data assimilation is an inverse problem with the aim of calibrating uncertain model parameters and states in order to generate results that match observed data within some tolerance. [17,50–54]. The process of assimilating observed data into numerical models has evolved from finding a single “best” solution to finding multiple solutions that take uncertainty into account. Some researchers have used GNs to tackle the problem of generating conditional numerical simulations and their uncertainties [55–58]. However, in these works they still need to simulate the full-order numerical model to predict forward in time. Mücke et al. [59] tackled this problem using a GN inside a MCMC method in order to replace the full-order numerical simulations, although they still need to generate thousands of full-order simulations to train the GN since it generates the solution for all time steps at once. For a review of machine learning methods applied to data assimilation and uncertainty quantification, see Cheng et al. [12].

3. Test case description

The test case used here is the spatio-temporal variation of a virus infection in an idealized town. The extended SEIRS model used in this work [35,49] extends the traditional theory of the dynamics of infectious diseases [60–63] to account for variations not only in time but also in space. The test case and parameter setup presented in the next section are the same as in Silva et al. [49] and Quilodr n-Casas et al. [35].

3.1. Extended SEIRS model and problem set up

The extended SEIRS model used in this work consists of four compartments (Susceptible–Exposed–Infections–Recovered) and two people groups (Home–Mobile). Fig. 1 shows the diagram of how individuals move between compartments and groups. The model starts with some individuals in the infectious compartments (Home-I/Mobile-I). The members of these compartment will spread the pathogen to the susceptible compartments (Home-S/Mobile-S). Upon being infected, the members of the susceptible compartments are moved to the exposed compartments (Home-E/Mobile-E) and remain there until they become infectious. Infectious individuals remain in the infectious compartment

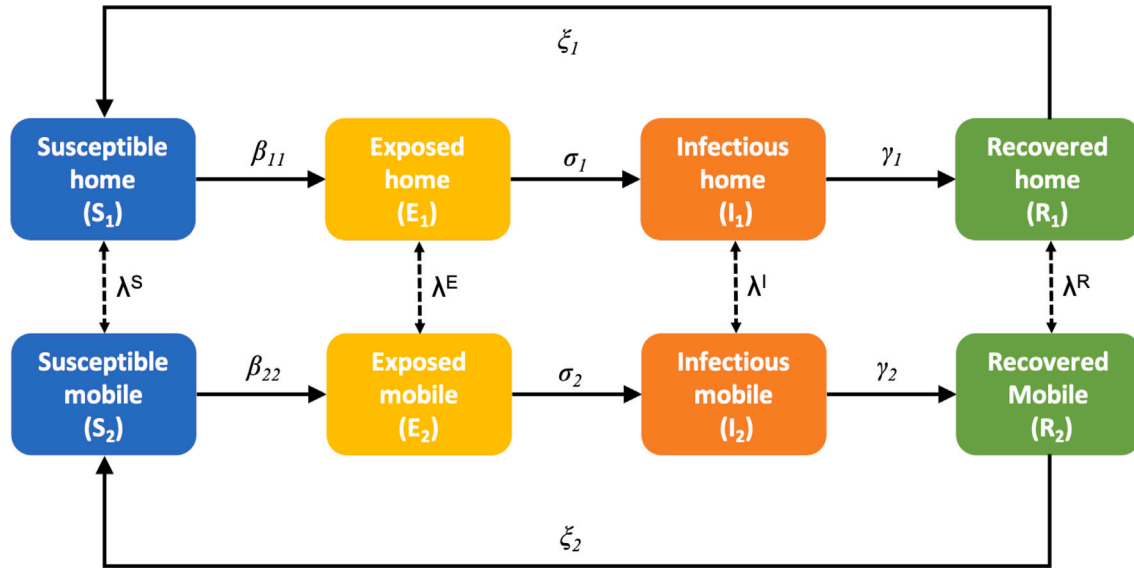


Fig. 1. Diagram of the extended SEIRS model. The diagram shows how people move between groups and compartments at a given point in space (or one cell in the grid). The vital dynamics and the transport via diffusion are not displayed here.

Source: Reprinted from Silva et al. [49].

until they become recovered (Home-R/Mobile-R). Recovered people can also become susceptible again due to the loss of immunity.

Modelling the movement of people is of the utmost importance in the spread of infectious diseases, such as COVID-19. Therefore, the goal of the extended SEIRS model is to reproduce the daily cycle of night and day, in which there is a pressure for mobile people to go to their homes at night, and there will be many people leaving their homes during the day and thus joining the mobile group. To this end, the extended SEIRS model uses a diffusion term (last term on the right of Eq. (1)) and an interaction term (penultimate term on the right of Eq. (1)) to model this process:

$$\frac{\partial S_h}{\partial t} = \eta_h N_h - \frac{S_h \sum_{h'} (\beta_{hh'} I_{h'})}{N_h} + \xi_h R_h - \nu_h^S S_h - \sum_{h'=1}^H \lambda_{hh'}^S S_{h'} + \nabla \cdot (k_h^S \nabla S_h), \quad (1a)$$

$$\frac{\partial E_h}{\partial t} = \frac{S_h \sum_{h'} (\beta_{hh'} I_{h'})}{N_h} - \sigma_h E_h - \nu_h^E E_h - \sum_{h'=1}^H \lambda_{hh'}^E E_{h'} + \nabla \cdot (k_h^E \nabla E_h), \quad (1b)$$

$$\frac{\partial I_h}{\partial t} = \sigma_h E_h - \gamma_h I_h - \nu_h^I I_h - \sum_{h'=1}^H \lambda_{hh'}^I I_{h'} + \nabla \cdot (k_h^I \nabla I_h), \quad (1c)$$

$$\frac{\partial R_h}{\partial t} = \gamma_h I_h - \xi_h R_h - \nu_h^R R_h - \sum_{h'=1}^H \lambda_{hh'}^R R_{h'} + \nabla \cdot (k_h^R \nabla R_h), \quad (1d)$$

where H represents the number of groups. Here, we have two groups of people, hence $H = 2$, one representing people at home $h = 1$, and the second representing people that are mobile $h = 2$ and outside their homes therefore. N_h represents the total number of individuals in each group, $\beta_{hh'}$ is the transmission rate between groups, σ_h is the rate of exposed individuals becoming infectious, γ_h is the recovered rate, and ξ_h is the rate recovered individuals return to the susceptible group due to loss of immunity. The vital dynamics are represented by η_h and ν_h , where η_h is the birth rate and ν_h is the death rate. The diffusion coefficient is represented by k_h and describes the movement of people around the domain. The interaction terms, $\lambda_{hh'}$, control how people move between groups, for example, how people that are in the mobile group move to the home group.

One important factor in dynamics of infectious diseases is the basic reproduction number ($\mathcal{R}_0$), which represents the expected number of new cases caused by a single infectious member in a completely sus-

ceptible population [64,65]. The $\mathcal{R}_0$ controls how rapidly the disease could spread and for each group it is define as

$$\mathcal{R}_{0h} = \frac{\sigma_h}{(\sigma_h + \nu_h)} \frac{\beta_{hh'}}{(\gamma_h + \nu_h)}, \quad (2)$$

where we assume $\beta_{hh'} = 0$ when $h \neq h'$ because people in their homes never directly meet mobile people (who are outside their homes). Here, we have two $\mathcal{R}_0$, one for each group of people. $\mathcal{R}_{01}$ representing the basic reproduction number of people at home, and $\mathcal{R}_{02}$ representing the basic reproduction number of people that are mobile and outside their homes therefore. For this case, we can also calculate an Effective $\mathcal{R}_0$ representing the $\mathcal{R}_0$ seen by the whole population at a specific time. It can be calculated as

$$\text{Effective } \mathcal{R}_0 = \frac{\sum_h S_h \mathcal{R}_{0h}}{\sum_h S_h}, \quad (3)$$

The idealized town and problem setup used in this work are the same as in Silva et al. [49] and Quilodr n-Casas et al. [35]. The simulation comprises 8000 people in an area of 100 km×100 km, discretized on a regular grid. The initial condition is that 0.1% of people have been exposed to the virus and will thus develop an infection. The model parameters are the two $\mathcal{R}_{0h}$ ($h = 1$ stands for home, and $h = 2$ mobile), and the model states are the number of people in each compartment (S, E, I and R) for each people group and for each grid block. Further information about the discretization and solution methods of the full-order numerical simulation can be found in Silva et al. [49] and Quilodr n-Casas et al. [35].

4. Method

We propose a novel framework to quantify uncertainties of numerical PDE simulations. The aim is to condition the numerical simulation to observed data (usually sparse in space and/or time) and estimate the corresponding uncertainties in the model states and parameters. Fig. 2 shows the proposed framework. We start by generating the dataset, running the full-order numerical simulations with different model parameters. The number of runs used to generate the dataset is orders of magnitude smaller than what would be required, if we exclusively rely on the full-order simulations to calculate the uncertainties. After generating the dataset, we apply the dimensionality reduction to the model states at each time step. Having the compressed states and corresponding model parameters, we can train a generative model

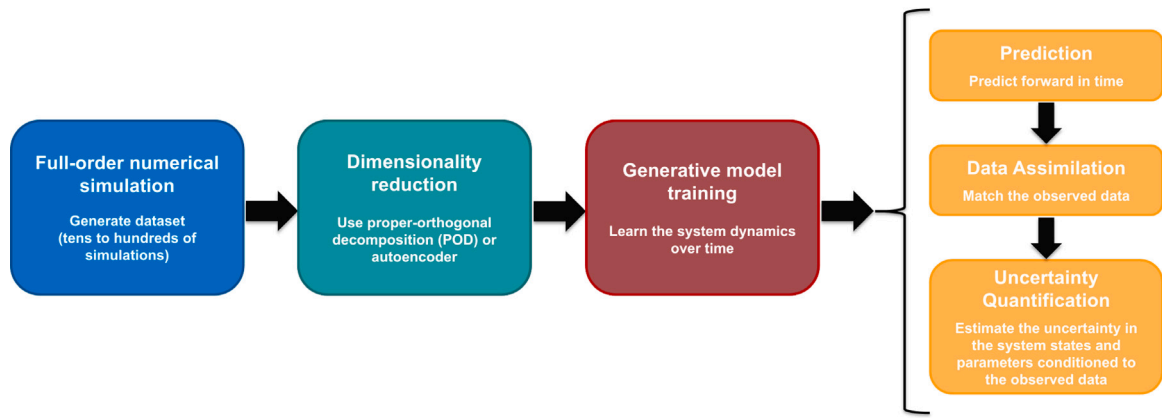


Fig. 2. Proposed framework to quantify uncertainty in the states and parameters of numerical PDE simulations.

to learn the dynamics of the system over time. The goal is for the network to learn how to generate a sequence (in time) of physically plausible compressed states and the corresponding model parameters. Following the training of the generative network, we can use it to predict forward in time [35,49], to find one solution that honours available measurements [49], and/or to find multiple solutions that honour the measurements and also generate the uncertainties in the model states and parameters (as proposed here). In the next sections, we describe each one of this parts.

In this work, we use generative adversarial networks (GANs) as generative models to produce time-sequences of the compressed states and model parameters of a spatio-temporal compartmental model in epidemiology, from a random distribution as input (see Fig. 3). The compression of the model states is performed using POD. After training, the prediction (PredGAN method) is performed in a recurrent way as described in Section 4.3, the data assimilation (DA-PredGAN method) modifies the prediction process to take account of the observed data (see Section 4.4), and the uncertainty quantification (UQ-PredGAN method proposed here) modifies the data assimilation process to work with an ensemble of models instead of a single realization. The proposed uncertainty quantification method is fully described in Section 4.5.

We assume in this work that the goal of the uncertainty quantification is to reduce the epistemic uncertainty in the model states and parameters of a physical system based on the observed data. The methods that inspired this work (RML/RTO) under certain conditions (see Section 4.5) generate the exact posterior; however, for most practical cases these conditions are not met, and an approximate posterior is required. We assume here that the epistemic uncertainty can be represented by approximate posterior distributions (generated by the proposed UQ-predGAN). We also compare these distributions with a method known for generating the correct posteriors (the MCMC).

4.1. Reduced order model and dimensionality reduction

The underlying assumption of a ROM is that the solution of the forward model can be accomplished by using considerably fewer degrees of freedom. One of the most widely used model reduction methods is the POD approach. In this work, we use a NIROM with POD as a compression method. We have also tested autoencoders to compress the data; however, the result using POD was practically the same and it does not need training or hyperparameter optimization.

After the first stage (the compression), the second stage of the NIROM (the evolution in time of the solutions) is accomplished using a particular type of generative network known as a generative adversarial network (GAN), as shown in Figs. 2 and 3. GANs have received much attention, after achieving excellent results for their generation of realistic-looking images [66–68]. Although Long short-term memory

(LSTM) networks are widely recognized as one of the most effective sequential models [69] for time series predictions, Quilodr n-Casas et al. [35] compared the performance of the LSTM and GAN as a NIROM. The GAN was able to learn better the underlying data distribution and reduce the forecast divergence, especially when predicting further in time.

4.2. Generative adversarial network training

Proposed by Goodfellow et al. [70], GANs are unsupervised learning algorithms capable of learning dense representations of the input data and are intended to be used as a generative model. Here, the generator network G directly produces time-sequences of the compressed states and model parameters of a numerical PDE simulation from a random distribution as input (latent vector z).

The discriminator network D attempts to distinguish between samples drawn from the training data and samples drawn from the generator, considered as fake. The output of the discriminator represents the probability that a sample came from the data rather than a “fake” sample from the generator. Fig. 3 show a schematic of the GAN training.

In this work, the GANs are trained using the non-saturating loss [71] for the DCGAN [66], and the Wasserstein loss with gradient penalty for the WGAN-GP [67]. During the training process the latent space z is generated from Gaussian random noise $\mathcal{N}(0, I_L)$, where I_L is an identity matrix of size L . The loss function of the discriminator takes as inputs: a time sequence of compressed states and model parameters from the numerical PDE simulation (“real” sample), and a time sequence of compressed states and model parameters generated by the generator (“fake” sample). The loss function of the generator only takes the “fake” samples as input. Please see Appendix A for more details about the GAN training losses and Appendix B for the GAN architectures. After training, the discriminator can be discarded since only the generator is used during the prediction, data assimilation, and uncertainty quantification processes.

4.3. PredGAN for time series prediction

After training the GAN to produce data (compressed simulation states and parameters) at a sequence of $m + 1$ time steps – given a latent vector z , the output of the generator $G(z)$ will be data at time steps $n - m$ to n no matter which point in time n represents – we can use the generator to make predictions in time in a recurrent way, using a ROM named Predictive GAN (PredGAN) as described in Silva et al. [49] and Quilodr n-Casas et al. [35].

Given known solutions at m consecutive time steps, we can perform an optimization to match the first m time levels in the output of the generator with the known solutions. After convergence, the last time

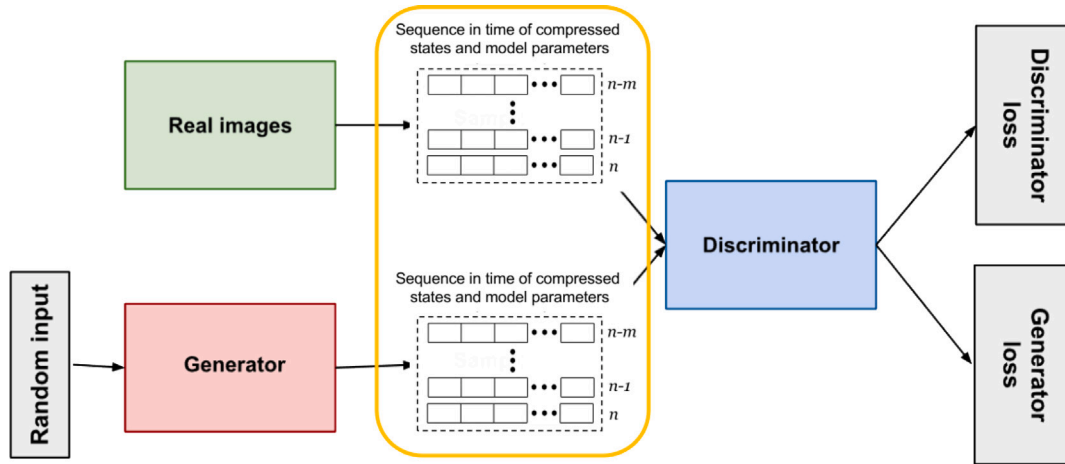


Fig. 3. The generative model (depicted as a GAN here) is trained to generate a sequence in time of compressed states and model parameters (inside the orange box) from the numerical PDE simulation. Modified from https://developers.google.com/machine-learning/gan/gan_structure.

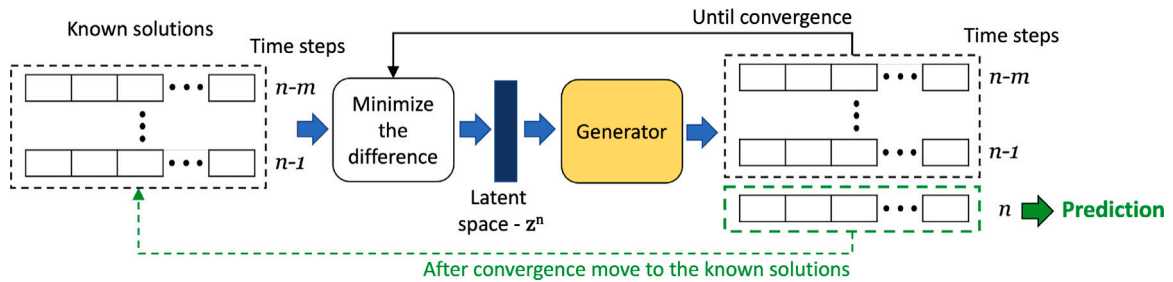


Fig. 4. Overview of the PredGAN process.

step, $m+1$, in the output of the generator is the prediction. We now can use this last time level $m+1$ as a known solution and perform another optimization to predict the time step $m+2$. The process continues until we predict all time steps. Fig. 4 illustrates how the PredGAN works.

In our case, after training, the output of the generator $G(\mathbf{z})$ is made up of $m+1$ consecutive time steps of compressed grid variables α (outputs/states of the numerical PDE simulation), and model parameters μ (inputs of the numerical PDE simulation). The compressed variables are POD coefficients, but could also be latent variables from an autoencoder. For a GAN that has been trained with $m+1$ time levels, $G(\mathbf{z})$ takes the following form

$$G(\mathbf{z}) = \begin{bmatrix} (\alpha^{n-m})^T, (\mu^{n-m})^T \\ \vdots \\ (\alpha^{n-1})^T, (\mu^{n-1})^T \\ (\alpha^n)^T, (\mu^n)^T \end{bmatrix}_{(m+1) \text{ by } (N_{\text{POD}} + N_{\mu})} \quad (4)$$

where the compressed grid variables are defined as $(\alpha^n)^T = [\alpha_1^n, \alpha_2^n, \dots, \alpha_{N_{\text{POD}}}^n]$. N_{POD} is the number of POD coefficients, and α_i^n represents the i th POD coefficient at time level n . The model parameters are represented as $(\mu^n)^T = [\mu_1^n, \mu_2^n, \dots, \mu_{N_{\mu}}^n]$. N_{μ} is the number of parameters, and μ_i^n represents the i th parameter at time level n .

In each iteration of the PredGAN, one new time step is predicted. Assuming the GAN has already been trained, and we have solutions at time levels from $n-m$ to $n-1$ for the POD coefficients, denoted by $\{\tilde{\alpha}^k\}_{k=n-m}^{n-1}$, and also have the model parameters over all time steps $\tilde{\mu}^k$, then to predict the solution at time level n we perform an optimization defined as

$$\mathbf{z}^n = \arg \min_{\mathbf{z}^n} \mathcal{L}_p(G(\mathbf{z}^n)), \quad (5)$$

$$\begin{aligned} \mathcal{L}_p(G(\mathbf{z}^n)) = & \sum_{k=n-m}^{n-1} (\tilde{\alpha}^k - \alpha^k)^T \mathbf{W}_{\alpha} (\tilde{\alpha}^k - \alpha^k) \\ & + \sum_{k=n-m}^{n-1} \zeta_{\mu} (\tilde{\mu}^k - \mu^k)^T \mathbf{W}_{\mu} (\tilde{\mu}^k - \mu^k), \end{aligned} \quad (6)$$

where $\mathbf{W}_{\alpha}$ is a square matrix of size N_{POD} whose diagonal values are equal to the weights that govern the relative importance of the POD coefficients, all other entries being zero. $\mathbf{W}_{\mu}$ is a square matrix of size N_{μ} whose diagonal values are equal to the model parameter weights, and the scalar ζ_{μ} controls how much importance is given to the model parameters compared to the compressed variables. The values for all the weighting terms are the same as in Silva et al. [49]. The tilde ($\tilde{\cdot}$) over the variables represents the known solutions. It is worth noticing that generative models other than GANs would also work with the PredGAN algorithm, since it only needs a generator and a way to optimize its output to match known solutions.

Only the time steps from $n-m$ to $n-1$ are taken into account in the functional (Eq. (6)) which controls the optimization of $\mathbf{z}^n$. After convergence, the newly predicted time level n is added to the known solutions $\tilde{\alpha}^n = \alpha^n$, and the converged latent variables $\mathbf{z}^n$ are used to initialize the latent variables at the next optimization to predict time step $n+1$. The process repeats until all time levels are predicted. The gradient of Eq. (6) is calculated by automatic differentiation [72–74], which means backpropagating the error generated by the loss function in Eq. (6) through the generator.

Fig. 5 shows one example of prediction using the PredGAN. The results represent the evolution, in one cell of the grid (one point in space), of the number of people in each group (home, mobile) and compartment (S, E, I and R) over time. The green circles are the known solutions used to start the prediction process, and the blue and orange lines represent the prediction and the ground truth, respectively. Each cycle in the graphs corresponds to a period of one day, when

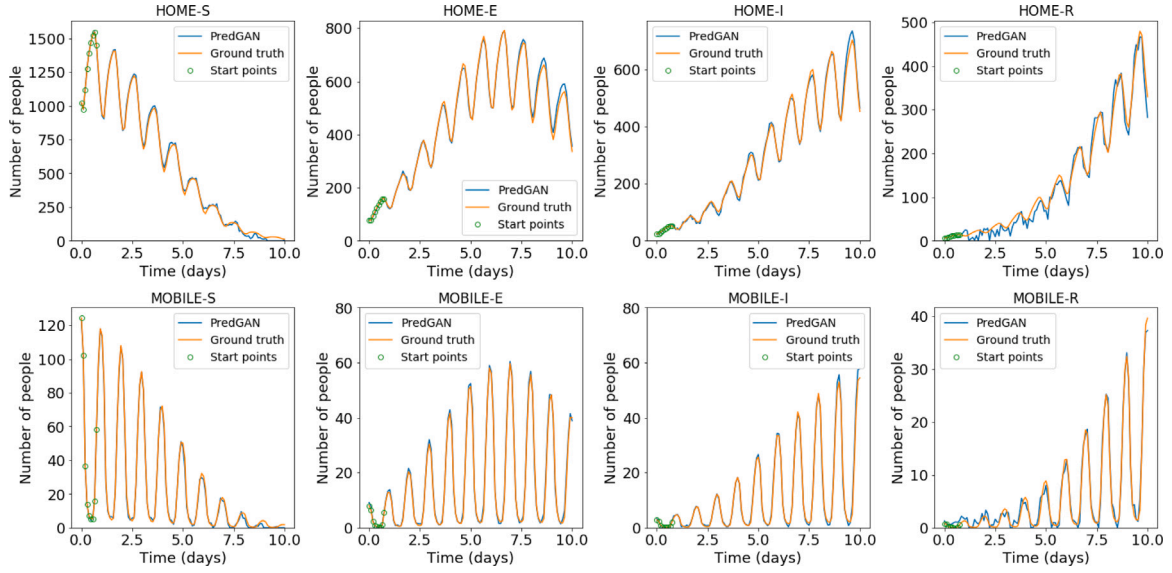


Fig. 5. Prediction using the PredGAN. The green circles are the known solutions used to start the prediction process. The blue and orange lines are the prediction and the ground truth, respectively.

mobile people leave their homes during the day and return at night. After the first nine time iterations the PredGAN does not see any data from the full-order numerical simulation, and relies completely on the predictions from PredGAN. Data from the full-order numerical simulation is only required as a starting point. For other applications of the PredGAN, see Jolaade et al. [47].

4.4. DA-PredGAN for data assimilation

Data assimilation is a type of inverse problem that aims to incorporate observed data into mathematical models. To perform data assimilation using the PredGAN process, we proposed in a previous work the Data Assimilation Predictive GAN (DA-PredGAN) [49] that involves three changes to the PredGAN. (i) One additional term is included in the loss function in Eq. (6) to take account of the data mismatch between the observed data and the generated values. (ii) The aim of the data assimilation is to match the observed data and to determine the model parameters μ^k (inputs of the numerical PDE simulation). Therefore, they are not known a priori, unlike when predicting in time. (iii) The forward marching in time is now replaced by forward and backward marching.

The optimization at each time step n of the forward march is given by

$$\mathbf{z}^n = \arg \min_{\mathbf{z}^n} \mathcal{L}_{da}(G(\mathbf{z}^n)), \quad (7)$$

$$\begin{aligned} \mathcal{L}_{da}(G(\mathbf{z}^n)) = & \sum_k (\tilde{\alpha}^k - \alpha^k)^T \mathbf{W}_\alpha (\tilde{\alpha}^k - \alpha^k) + \sum_k \zeta_\mu (\tilde{\mu}^k - \mu^k)^T \mathbf{W}_\mu (\tilde{\mu}^k - \mu^k) \\ & + \sum_k \zeta_{obs} (\mathbf{d}^k - \mathbf{d}_{obs}^k)^T \mathbf{W}_{obs}^k (\mathbf{d}^k - \mathbf{d}_{obs}^k), \end{aligned} \quad (8)$$

where $k \in \{n-m, n-m+1, \dots, n-1\}$. The observed data at each time step k is stored in the vector $\mathbf{d}_{obs}^k$ of size N_{obs} . $\mathbf{d}^k$ is the generated data calculated based on the output of the generator at time step k . In our case, it represents data at some points in the grid (high dimensional states) and it is calculated through the POD coefficients α^k and stored eigenvectors. $\mathbf{W}_{obs}^k$ is a square matrix of size N_{obs} whose diagonal values are equal to the observed data weights, and the scalar ζ_{obs} direct controls how much importance is given to the data mismatch. The values in the diagonal of $\mathbf{W}_{obs}^k$ are set to zero where we have no observation. After convergence, the newly predicted time level n is added to the known solutions $\tilde{\alpha}^n = \alpha^n$, and different from the

prediction, we also update the model parameters using the newly predicted time step $\tilde{\mu}^n = \mu^n$. After the forward march, the process continues with a backward march. For the latter instead of working forward in time, the process goes backwards in time, from the last time step to the first. The data mismatch functional for the optimization at each iteration n of the backward march is defined as in Eq. (8), but now for the backward march $k \in \{n+m, n+m-1, \dots, n+1\}$.

After performing a forward and backward march using Eqs. (8), the average of the data mismatch (last term on the right of Eqs. (8)) at the end of all iterations n is calculated. If the average mismatch has not converged or the maximum number of iterations is not reached, the process continues with a new forward and backward march. A relaxation factor is also introduced to stabilize the process of marching forward and backward in time as in Silva et al. [49].

Fig. 6 shows an overview of the DA-PredGAN process and Fig. 7 shows two data assimilation results (top and bottom plots). The results show the evolution in time of the simulation states in one cell of the grid (one point in space). We generate observed data from a full-order numerical simulation that was not included in the training set of the GAN. We also added 5% noise to the chosen data. We note that for both cases (top and bottom plots) the observed data is honoured, although their results are slightly different. We only show here the home group in order to save space, the results for the mobile group are similar.

4.5. UQ-PredGAN for uncertainty quantification

The computation of a single model that matches the observed data is usually insufficient to quantify risks and uncertainties. Data assimilation is generally an ill-posed inverse problem [50,75], hence several models can match the observed data, within some tolerance (as in Fig. 7). In order to quantify uncertainty, we propose in this work a ROM named Uncertainty Quantification Predictive GAN (UQ-PredGAN). This method is inspired by the RML (or RTO) as a way of sampling a posterior distribution conditioned to observed data. In the RML/RTO, the numerical simulation is used to predict forward in time, and for each sample, an optimization (data assimilation) is performed to condition the models to the observed data. The challenge is usually to perform the optimization, since the full-order numerical simulation needs to be run several times and usually adjoints are not present. In this work, the proposed method UQ-PredGAN can compute uncertainties relying just on a set of unconditioned numerical simulations. The prediction,

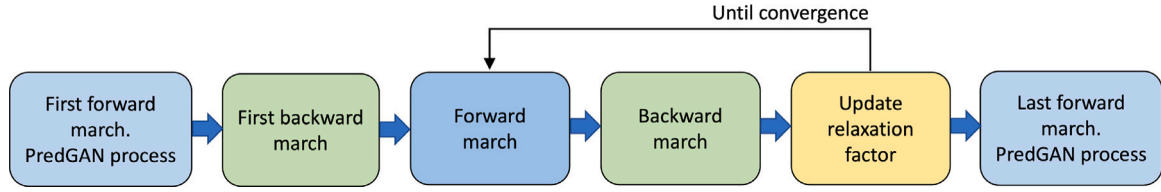


Fig. 6. Overview of the DA-PredGAN. Each march represents going through all time steps. The last forward march is optional (mostly for parameterized problems).

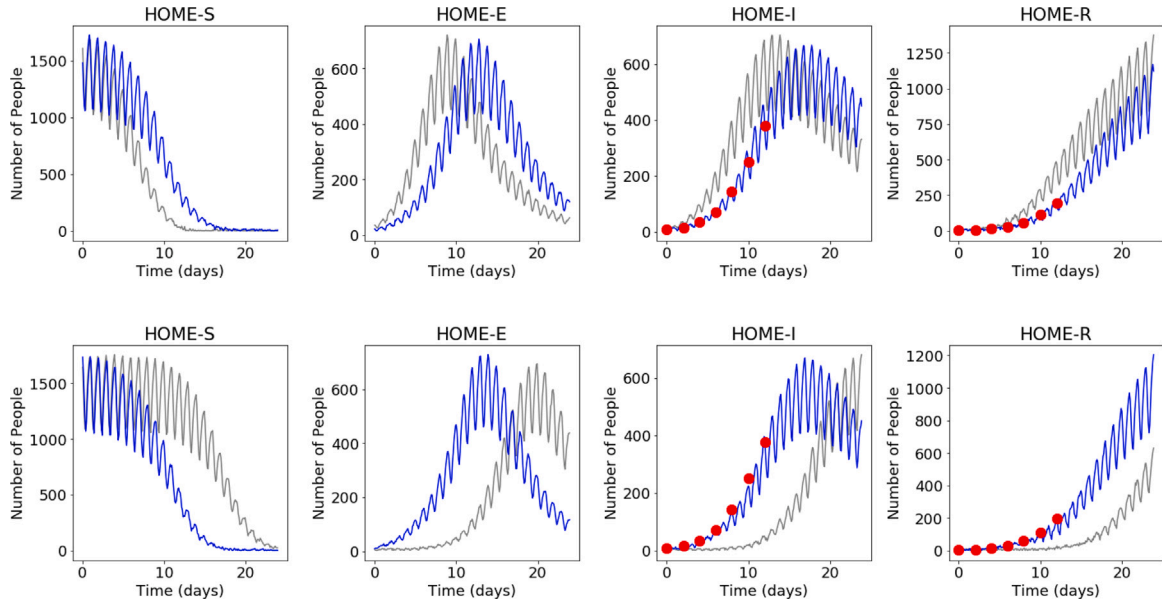


Fig. 7. Two different data assimilations (top and bottom) using the DA-PredGAN. The circles in red are the observed data (the same in both cases), the lines in grey the initial conditions (first forward march), and the blue lines the final results (last forward march).

data assimilation and uncertainty quantification are performed using the inherent adjoint capability present within neural networks, and no additional full-order numerical simulations, other than those used for training the GAN, are required.

The idea is to generate several models that match the observed data and which can be used to quantify the uncertainty in the model states (outputs of the numerical simulation) and model parameters (inputs of the numerical simulation). To this end, we perform several data assimilation using the DA-PredGAN algorithm with the modified optimization process for each forward/backward time step,

$$\mathbf{z}^n = \arg \min_{\mathbf{z}^n} \mathcal{L}_{uq,j}(G(\mathbf{z}^n)), \quad (9)$$

$$\begin{aligned} \mathcal{L}_{uq,j}(G(\mathbf{z}^n)) = & \sum_k \left(\tilde{\alpha}_j^k - \alpha^k \right)^T \mathbf{W}_\alpha \left(\tilde{\alpha}_j^k - \alpha^k \right) \\ & + \sum_k \zeta_\mu \left(\tilde{\mu}_j^k - \mu^k \right)^T \mathbf{W}_\mu \left(\tilde{\mu}_j^k - \mu^k \right) \\ & + \sum_k \zeta_{obs} \left(\mathbf{d}^k - \mathbf{d}_{obs}^k + \epsilon_j^k \right)^T \mathbf{W}_{obs}^k \left(\mathbf{d}^k - \mathbf{d}_{obs}^k + \epsilon_j^k \right), \quad (10) \end{aligned}$$

where for the forward march $k \in \{n-m, n-m+1, \dots, n-1\}$ and for the backward march $k \in \{n+m, n+m-1, \dots, n+1\}$. Considering N_s as the number of data assimilation to be performed, then $j = 1, \dots, N_s$. The value of $N_s = 200$ was chosen based on previous experience using the RML/RTO. The observed data error is represented by the random vector ϵ , and we consider here that all measurement errors are uncorrelated, thus they are sampled from a normal distribution with zero mean and a specified standard deviation. For the test case used here, we set the

standard deviation equal to 5% of the corresponding observed data. For each data assimilation j , we use a different prior $\tilde{\mu}_j^k$ with the corresponding initial condition $\{\tilde{\alpha}_j^k\}_{k=0}^m$, and a different perturbation on the observed data ϵ_j^k .

The UQ-PredGAN is proposed as follows:

1. Sample the model parameters $\tilde{\mu}_j$ from a normal distribution $\mathcal{N}(\tilde{\mu}, \mathbf{C}_\mu)$, where $\mathbf{C}_\mu$ is the covariance matrix of the model parameters, and $\tilde{\mu}$ is the model parameter mean vector.
2. Sample the measurement error ϵ_j from a normal distribution $\mathcal{N}(0, \mathbf{C}_d)$, where $\mathbf{C}_d$ is the covariance matrix of the measurement error.
3. Assimilate data using the DA-PredGAN process, but using Eqs. (9) and (10) for the forward and backward marches.
4. Repeat the process until the final number of data assimilation samples N_s is reached.

After performing N_s steps of the UQ-PredGAN, accept all realizations that obtained an acceptable level of data mismatch. It is worth mentioning that for the RML/RTO, when the case is linear and normal distributions are used to sample the model parameters and measurement error, the RML/RTO samples the corrected posterior distribution [18,20,21]. In this work, we also use normal distributions to sample the model parameters and measurement error; however, the test case is nonlinear, and the weighting terms are seen as tuning parameters (as in Stordal and Nævdal [18]). Thus, the results are an approximate sample of the posterior distribution.

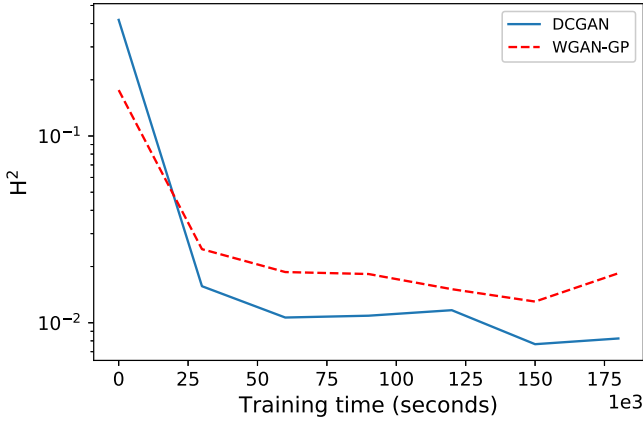


Fig. 8. Comparison of the DCGAN and the WGAN-GP, using the proposed metric (H^2). Smaller values are better.

4.6. Time-stepping regularization

One limitation of the (DA/UQ)-PredGAN is that during the optimization process of each time step, the latent vector $\mathbf{z}$ can end up in a region in the latent space not well populated, which could generate non-physical solutions. This can jeopardize subsequent time step iterations since they use the previous latent values as a initial condition. To solve this problem, we propose here to use a regularization of the latent vector during the time stepping process of the (DA/UQ)-PredGAN. The new time-stepping functionals of the (DA/UQ)-PredGAN are now defined as,

$$\mathcal{L}_p^{\text{reg}}(G(\mathbf{z}^n)) = \mathcal{L}_p(G(\mathbf{z}^n)) + \frac{\lambda_p}{N_z} \|\mathbf{z}^n\|_2^2, \quad (11)$$

$$\mathcal{L}_{da}^{\text{reg}}(G(\mathbf{z}^n)) = \mathcal{L}_{da}(G(\mathbf{z}^n)) + \frac{\lambda_{da}}{N_z} \|\mathbf{z}^n\|_2^2, \quad (12)$$

$$\mathcal{L}_{uq,j}^{\text{reg}}(G(\mathbf{z}^n)) = \mathcal{L}_{uq,j}(G(\mathbf{z}^n)) + \frac{\lambda_{uq}}{N_z} \|\mathbf{z}^n\|_2^2, \quad (13)$$

where N_z is the size of the latent vector, and in this work we use $\lambda_p = \lambda_{da} = \lambda_{uq} = 0.01$. Different values were tested and 0.01 generated the better results. We noticed that by using the regularization, we improve the results of the prediction, data assimilation and uncertainty quantification by generating more plausible physical solutions and then improving the convergence in each time-step iteration of these methods.

4.7. Generative model evaluation

GANs are notoriously difficult to evaluate while training as the loss paths of the generator and discriminator behave in non-predictable ways [76]. However, such evaluation is of particular interest as it helps assessing the machine learning model's convergence state, relative to other models, even before it is used for predictions, data assimilation and uncertainty quantification. In addition, evaluating the generative model performance during training allows us to select the generative model with the best weights from the last learning iterations, and for stopping the training when no improvement is observed.

To evaluate the learning and convergence abilities of generative networks while we train them, we propose an evaluation metric based on the Bhattacharyya coefficient (BC) (or the Hellinger distance (H), since $1 - BC = H^2$) [77]. It can be calculated as

$$H^2(P, Q) = 1 - \sum_{i=1}^k \sqrt{p_i q_i}, \quad H^2 \in [0, 1], \quad (14)$$

where $P = (p_1, \dots, p_k)$ and $Q = (q_1, \dots, q_k)$ are two discrete probability distributions, the first coming from the generated samples and the second from the training dataset. We use the fact that the output of

the generator is a spatio-temporal map (Eq. (4)) that has semantically different axes with one being the time dimension, and the other being the POD coefficients (or autoencoder latent values) and the model parameters. We could expect that if we take the mean and standard deviation from each POD coefficient (one column) from one spatio-temporal map, the distributions of these means and standard deviations over all training spatio-temporal maps will have distinct values representing the characteristics of each POD coefficient. Therefore, we calculate the evaluation metric (H^2) for the mean and standard deviation of each of the POD coefficients, and then we average them to form the final score. A comparison between the distributions of the real and generated data for different GANs can be found in Appendix E.

Other metrics for comparing discrete distributions could also be used Gibbs and Su [78]. We chose the BC coefficient (or H^2) due to its relative simplicity when applied to a large number of samples. This is important as it would be undesirable to “waste” significant amount of time evaluating the model rather than training the model in practice.

5. Results

5.1. Dataset and generative model training

For the training process 40 full-order numerical simulations were performed in order to generate the training dataset. It is important to notice that we chose this value to be much smaller than what would be needed to perform the uncertainty quantification relying only on the full-order numerical simulations. Each simulation consists of two different $\mathcal{R}_{0h}$, one for people at home and another for mobile people. The spatial domain of the numerical simulation is a regular grid of 10×10 (100 cells) (see Silva et al. [49] for more details). Considering that each type of people (people at home and mobile) has the four quantities of the extended SEIRS model (susceptible, exposed, infectious and recovered), there will be eight variables for each cell in the grid, which gives a total number of $100 \times 8 = 800$ variables (model states), per time step. POD is performed to the 800 variables, in order to work with a low dimensional space in the GAN-based ROM. The first 15 POD coefficients and basis functions were chosen because they capture $> 99.99\%$ of the variance held in the time snapshots. Hence the GAN is trained to generate the 15 POD coefficients (α^n) and the two $\mathcal{R}_{0h}$ (μ^n) over a sequence of 10 time steps. We choose this time length because it represents a cycle (one day) in the results.

We choose the size of the latent vector $\mathbf{z}$ to be 100 in order to have enough degrees of freedom to encompass the representation of the compressed physical states and model parameters. The networks receive/generate the 10 time levels as a two-dimensional array (“an image”, Eq. (4)) with 10 rows and 17 columns. Each row represents a time level and each column comprises the 15 POD coefficients and the two $\mathcal{R}_{0h}$. We choose this configuration, instead of a linear representation, to exploit the time dependence captured in the two-dimensional array. We also carried out initial tests using a linear representation of the time level outputs and a multi-layer perceptron as a generator and discriminator. However, it generated worse results than the two-dimensional representation.

We decided to test two well-known GANs in this work, the DCGAN [66] and the WGAN-GP [67]. The generator and discriminator architectures can be found in Appendix B. Fig. 8 shows a comparison of both models over the training time using the proposed evaluation metric (H^2) (Section 4.7). From Fig. 8, we can see that DCGAN outperforms the WGAN-GP. Hence, the former was selected to continue the experiments. Further analysis of the evaluation metric can be found in Appendix E.

5.2. Time-stepping regularization

In this work, we use regularization to improve the prediction, data assimilation and uncertainty quantification. Considering the latter,

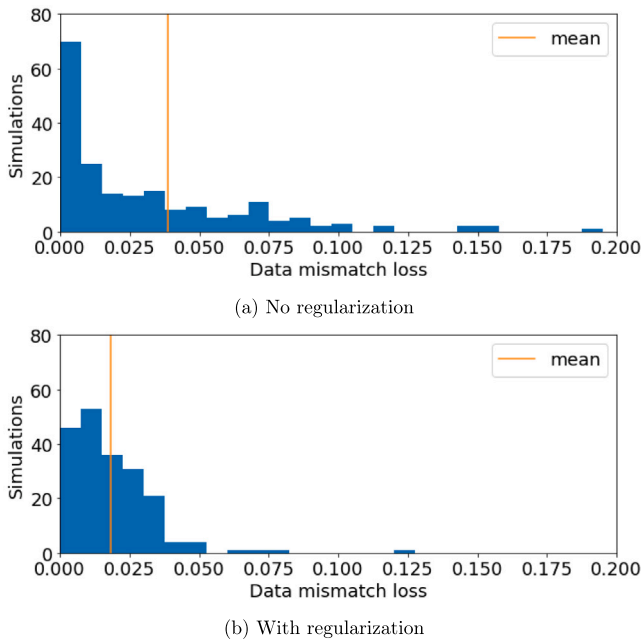


Fig. 9. Histogram of the data mismatch between the observed and simulated data.

after running the UQ-PredGAN for the 200 sampled model parameters $\mathcal{R}_{0h}$ (see Section 5.3), without using regularization, 104 realizations reached an acceptable level of data mismatch, while using regularization, 121 reached an acceptable level. Fig. 9 shows the histogram of the data mismatch between the observed and simulated data (average of the last term on the right of Eq. (8) over all times). We can see that the cases with regularization reached a considerably lower data mismatch. We considered an acceptable level of data mismatch to correspond to a functional value of less than 0.02. For the remainder of this paper regularization of the latent space will be used in all the experiments.

5.3. Parameter estimation and uncertainty quantification

In this section, we apply the proposed framework to solve an inverse problem for the extended SEIRS model. The goal is to quantify the uncertainties in the model states and parameters, considering the availability of observed data. The model represents the spread of COVID-19 in an idealized town (as in Quilodr n-Casas et al. [35] and Silva et al. [49]). We generate the observed data (“measurements”) from a randomly chosen full-order numerical simulation ($\mathcal{R}_{01} = 7.7$, $\mathcal{R}_{02} = 17.4$) that was not included in the training set. Observed data was collected at five points of the domain, the measurements are available every two days, and we measured only infectious and recovered people, as in Silva et al. [49]. The $\mathcal{R}_{0h}$ are not used as observed data. For generating the priors (unconditional simulations) 200 model parameters $\mathcal{R}_{0h}$ were sampled from a normal distribution with a mean of 10 and standard deviation of 4. The mean and standard deviation were chosen based on Kocha czyk et al. [79]. The 200 model parameters and their corresponding initial conditions were used to start the UQ-PredGAN process. For each of the model parameters, one data assimilation was performed as described in Section 4.5. We also use time-stepping regularization to physically constrain the solution (further analysis on the impact of the regularization are presented in Section 5.2). After the data assimilation, 121 realizations were accepted based on their data mismatch error. It is worth noting that for the whole uncertainty quantification process using the UQ-PredGAN, we required only 40 full-order numerical simulations (for training the GAN).

Fig. 10 shows the UQ-PredGAN results for each group and compartment (model states) at one point in space where observed data

was available. The priors (grey lines) are the unconditional simulations used to start each data assimilation, and the posteriors (blue lines) are the conditional simulations, accepted realizations after the data assimilation. The posterior mean (black line) is also shown in the plots. We can see from these figures that the conditional simulations (posteriors) generated by the UQ-PredGAN match the observed data, within some tolerance (we considered a measurement error of 5%), and the uncertainty is propagated through the simulation time. The high frequency oscillation presented in the results corresponds to a daily cycle, when mobile people leave their homes during the day and return to them at night. Comparable results were observed at other points in domain, hence they are not presented here.

In the plots from the first row of Fig. 11, we show the probability density function (PDF) of the $\mathcal{R}_{0h}$ (model parameters) and Effective $\mathcal{R}_0$, for the priors and posteriors. The result shows that the UQ-PredGAN was able to reduce the uncertainty in the model parameters approaching the true values used to generate the observed data. Note that the data assimilation is an inverse and usually ill-posed problem, hence different values of $\mathcal{R}_{0h}$ could match the measurements within some tolerance. We have also run a comparison of the UQ-PredGAN with the gold standard MCMC, using the Metropolis–Hastings algorithm (please, see Appendix F for more details). The results from the MCMC are in the plots from the second row of Fig. 11. It is worth mentioning that to run the MCMC, it took more than a month on a dedicated workstation (more than 100,000 samples and we stopped because of time limitations, see Appendix G for the hardware configuration), while to run the UQ-PredGAN only 12 h (two days if we add the GAN training). We can notice from Fig. 11 that the UQ-PredGAN generates uncertainties that have a reasonable match with the gold standard, but with orders of magnitude less computational time. In practical terms, it is unaffordable to run the MCMC for most numerical PDE simulations.

We can notice that the distributions generated by the UQ-PredGAN have longer “tails” than the ones from the MCMC. We checked the results from the simulations in that portion of the distribution and they reasonably match the observed data. It may indicate that the MCMC has not converged yet, which means that we would need many more simulation samples ($> 100,000$) for the MCMC to generate the correct tails, which makes it impractical. Fig. 11 also shows that the UQ-PredGAN is able to sample the second small mode of the mobile group $\mathcal{R}_{02}$. It is where the true value of the $\mathcal{R}_{02}$ used to generate the observed data occurs. We can also see that for the mobile group, the UQ-PredGAN posterior PDF does not exactly match the MCMC results as well as for the home group. This could be because the number of mobile people is one order of magnitude smaller than the number of people at home, which gives the latter more importance during the data assimilation process, and the relative rate of change of the number of people in the mobile group is much greater than in the home group (Fig. 10), thus small perturbations in the former can cause huge relative deviations.

Fig. 12 shows the prediction of the Effective $\mathcal{R}_0$ at day 16, along with the ground truth, for the UQ-PredGAN and MCMC. Comparable results were observed for other points in time. The last observed data used in this experiment was at day 12. The results demonstrate that the UQ-PredGAN can generate predictions and uncertainties that accurately match the ground truth and have a reasonable match to the gold standard MCMC. Table 1 shows the posterior mean, mode and standard deviation (STD) for different days. The mode is calculated based on the histograms. We can notice that going further in time the posterior uncertainty (the STD) in the Effective $\mathcal{R}_0$ increases, in both methods. This is because the further away the predictions are from the last observed data, the greater the uncertainty will be in those predictions. The difference in the standard deviation of the UQ-PredGAN and the MCMC can be attributed to the impracticability of running the required number of simulations in order for the MCMC to represent accurately the tails of the distributions, as previously mentioned.

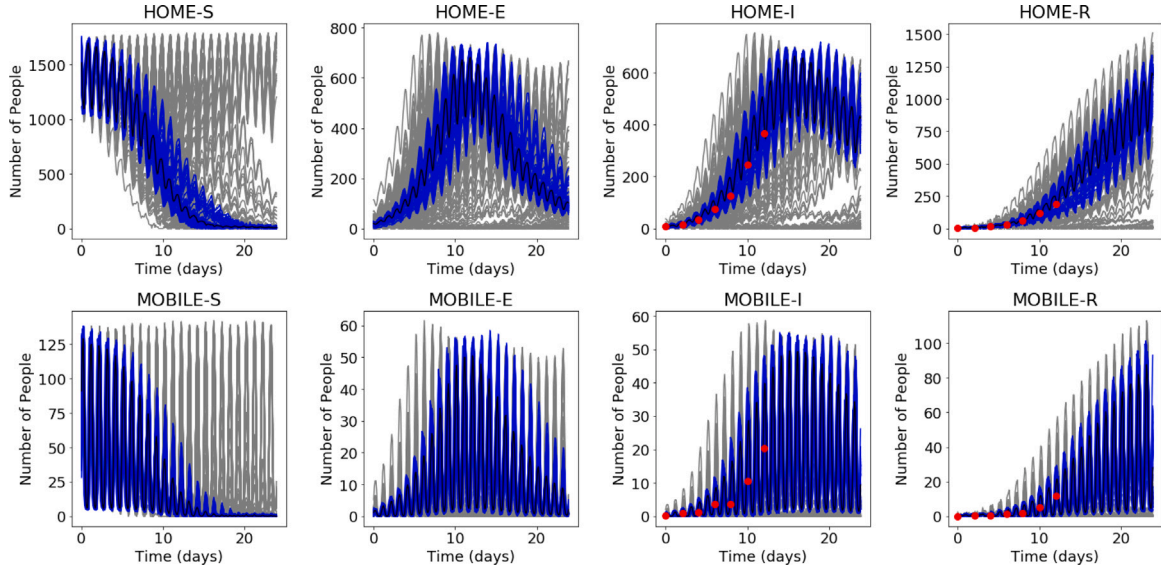


Fig. 10. Results of the UQ-PredGAN for each group and compartment at one point in space. The red dots represent the observed data ("measurements"), the grey lines the unconditional simulations (priors), the blue lines the conditional simulations (posteriors), and the black line the posterior mean.

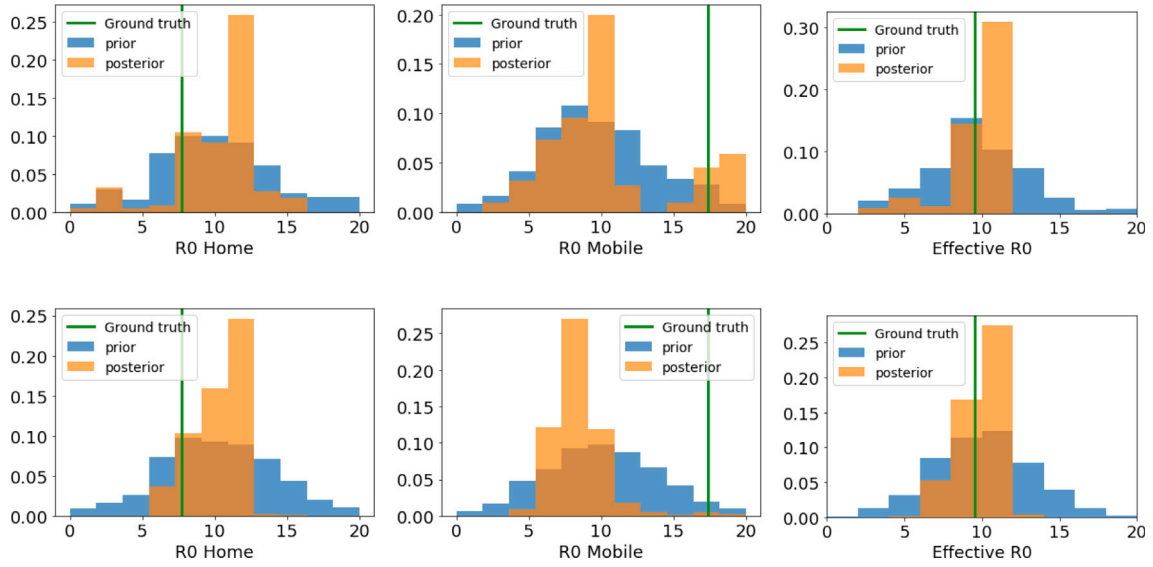


Fig. 11. Probability density functions of the R_{0h} and Effective R_0 at day 12. Comparison of UQ-PredGAN (first row) with the gold standard MCMC (second row). The ground truth represents the values in the full-order numerical simulation used to generate the observed data.

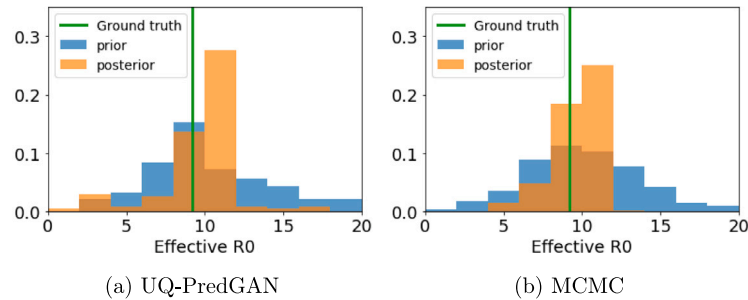


Fig. 12. Predicted probability density functions of the Effective R_0 at day 16 (the last observed data was at day 12). Comparison of UQ-PredGAN with the gold standard MCMC. The ground truth represents the value in the high fidelity numerical simulation used to generate the observed data.

Table 1

Values of the posterior PDF of the Effective R_0 over time. Comparison of UQ-PredGAN with the gold standard MCMC.

Time (days) =>	12	16	22
ground truth	9.52	9.22	9.48
uq-predgan mean	9.86	9.70	9.91
mcmc mean	9.89	9.68	9.92
uq-predgan mode	11.0	11.0	11.0
mcmc mode	11.0	11.0	11.0
uq-predgan std	1.73	2.44	2.70
mcmc std	1.39	1.40	1.51

6. Discussion

Quantifying the uncertainty in the model states and model parameters of numerical PDE simulations, given some observed data (measurements), is a challenge for most practical problems. The computational cost of the numerical simulation itself is usually high. It becomes even higher when data assimilation (model calibration) is considered, even for only one model. In order to quantify uncertainty, an ensemble of model need to be calibrated which makes this process almost unfeasible for some problems when relying only on the numerical solution. Here, we tackle this problem by proposing a ROM based on generative models. The GN-based ROM can be seen as a surrogate of the numerical simulation, that can be calibrated and generate the corresponding uncertainties without any new full-order numerical simulation, apart from those used to train the network (we used 40 simulations in this work).

One question that may arise is how good is the uncertainty generated by the GN-based ROM. To answer this question, we compared our method with the gold standard MCMC. Running the MCMC usually requires hundreds of thousands to millions of simulations, which, even when using a surrogate, can be computationally demanding (in our case a month running in a workstation, see [Appendix G](#)). Representing a speed-up of about 60 $\times$, our GN-based ROM took just 12 h (two days including the network training) and was able to generate uncertainties that reasonably match the gold standard. It is worth mentioning, that most of the methods used in practice to quantify uncertainties of numerical PDE simulations still require a huge computational time, and only generate an approximation of the correct posterior [10,12–14,18,21,55].

In terms of limitation, for the UQ/DA-PredGAN to work well the generative model (here a GAN) needs to represent the physical solution, which means it needs to be well-trained. The training of GANs or other generative models has been a topic of recent interest and advances in the research community [66,67,76,80], what may well help further applications of the proposed methods.

7. Conclusion

We introduce three main novelties in this work: (i) a novel application of generative networks as a reduced-order model (ROM) capable of solving inverse problems in numerical PDE simulations. (ii) The integration of regularization techniques to enhance the plausibility of the physical solutions, thereby improving the results of the DA/UQ-PredGAN methods. (iii) A new method for evaluating the training of generative models by leveraging the known structure of real and generated samples. We applied the proposed framework to a spatio-temporal compartmental model in epidemiology. The results show that the UQ-PredGAN accurately matches the observed data and efficiently quantifies/reduces uncertainty in the model states (groups and compartments) and model parameters (basic reproduction numbers). We compare the UQ-PredGAN with the gold standard MCMC, and show that it can generate predictions and uncertainties that reasonably match the gold standard, but with orders of magnitude less computational

time (speed up of 60 $\times$). The proposed method is not limited to the underlying physics of this application, it is a general framework for quantifying uncertainties of numerical physical simulations.

CRedit authorship contribution statement

Vinicius L.S. Silva: Conceptualization, Data curation, Formal analysis, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Claire E. Heaney:** Conceptualization, Formal analysis, Supervision, Writing – original draft, Writing – review & editing. **Nenka Nenov:** Conceptualization, Methodology, Software. **Christopher C. Pain:** Conceptualization, Formal analysis, Supervision, Writing – original draft, Writing – review & editing.

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.

Acknowledgements

We would like to acknowledge financial support from Petrobras, Brazil for the first author. This work is also supported by the following EPSRC and NERC, UK grants: RELIANT, UK, “Risk Evaluation fAst iNtelligent Tool for COVID19” (EP/V036777/1); CO-TRACE, UK, “COVID-19 Transmission Risk Assessment Case Studies — education Establishments” (EP/W001411/1); INHALE, UK, “Health assessment across biological length scales” (EP/T003189/1); SmartRes, Smart assessment, management and optimisation of urban geothermal resources, UK (NE/X005607/1); MUFFINS, Multiphase Flow-induced Fluid-flexible structure InteractionN in Subsea applications, UK (EP/P033180/1); ECO-AI, UK, “Enabling CO₂ capture and storage using AI” (EP/Y005732/1); and the PREMIERE, UK programme grant (EP/T000414/1).

Appendix A. GAN training losses

In this work, the GAN is trained using the non-saturating loss [71] for the DCGAN [66], and the Wasserstein loss with gradient penalty for the WGAN-GP [67]. The generator network G directly produces time-sequences from a random distribution as input (latent vector $\mathbf{z}$):

$$G : \mathbf{z} \sim \mathcal{N}(0, I_L) \rightarrow \tilde{\mathbf{x}} \in \mathbb{R}^{m+1 \times N} \quad (15)$$

where $\tilde{\mathbf{x}}$ is an array of $m + 1$ time sequences with N dimensions, L is the size of the latent vector, and I_L is an identity matrix of size L . The discriminator network D attempts to distinguish between samples drawn from the training data $\mathbf{x}$ and samples drawn from the generator $\tilde{\mathbf{x}}$, considered as fake. The output of the discriminator $D(\mathbf{x})$ represents the probability that a sample came from the data rather than a “fake” sample from the generator. The output of the generator $G(\mathbf{z})$ is a sample from the distribution learned from the dataset.

Eqs. (16) and (17) show the loss functions of the discriminator and generator for the DCGAN training, respectively:

$$L_D = -\mathbb{E}_{\mathbf{x} \sim p_{data}} [\log(D(\mathbf{x}))] - \mathbb{E}_{\tilde{\mathbf{x}} \sim p_{gen}} [\log(1 - D(\tilde{\mathbf{x}}))] \quad (16)$$

$$L_G = -\mathbb{E}_{\tilde{\mathbf{x}} \sim p_{gen}} [\log(D(\tilde{\mathbf{x}}))] \quad (17)$$

Eqs. (18) and (19) show the loss functions of the discriminator and generator for the WGAN-GP training, respectively:

$$L_D = \mathbb{E}_{\tilde{\mathbf{x}} \sim p_{gen}} [D(\tilde{\mathbf{x}})] - \mathbb{E}_{\mathbf{x} \sim p_{data}} [D(\mathbf{x})] + \lambda \mathbb{E}_{\tilde{\mathbf{x}} \sim p_{gen}} [(\|\nabla_{\tilde{\mathbf{x}}} D(\tilde{\mathbf{x}})\|_2 - 1)^2] \quad (18)$$

$$L_G = -\mathbb{E}_{\tilde{\mathbf{x}} \sim p_{gen}} [D(\tilde{\mathbf{x}})] \quad (19)$$

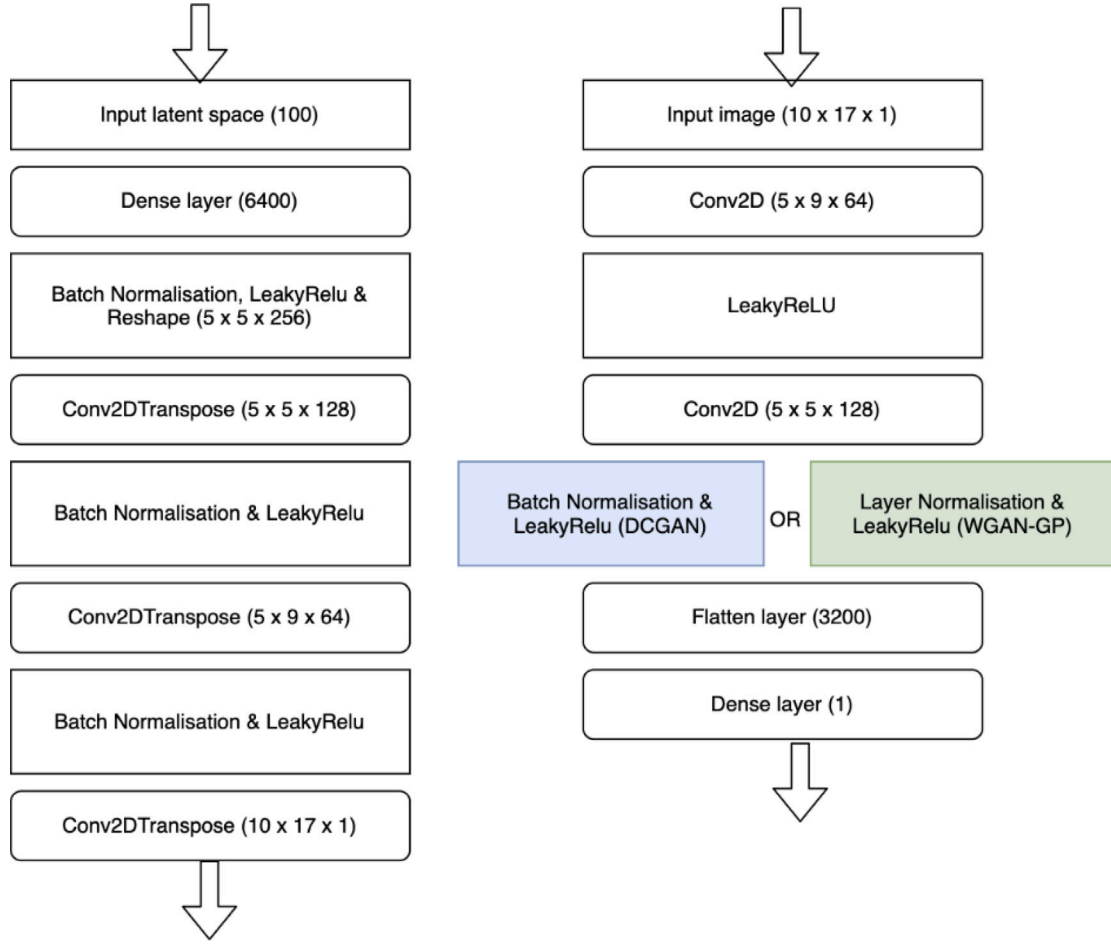


Fig. 13. DCGAN and WGAN-GP architectures.

Appendix B. GAN architectures

The generator and discriminator architectures for the DCGAN and WGAN-GP can be found in Fig. 13.

Appendix C. PredGAN algorithm

Assume we already have solutions for the POD coefficients (compressed versions of numerical PDE solution states), up to a certain time level denoted as m . These solutions can be represented as $\{\tilde{\alpha}^k\}_{k=1}^m$. And consider that the model parameters are known throughout the entire simulation time as $\tilde{\mu}^k$, we can use the following steps to predict upcoming solutions:

1. randomly generated a latent vector $^{(0)}\mathbf{z}^{m+1}$ in order to start the prediction of time level $m+1$. The bracketed superscript to the left of the latent vector represents the count of optimization iterations during a time step prediction;
2. update the time iteration counter to $n = m + 1$;
3. start the optimization iteration counter at $l = 0$;
4. using the GAN's generator, evaluate the current latent vector $^{(l)}\mathbf{z}^n$, yielding

$$G(^{(l)}\mathbf{z}^n) = \begin{bmatrix} (\alpha^{n-m})^T, (\mu^{n-m})^T \\ \vdots \\ (\alpha^{n-1})^T, (\mu^{n-1})^T \\ (\alpha^n)^T, (\mu^n)^T \end{bmatrix}_{(m+1) \text{ by } (N_{\text{POD}} + N_{\mu})} \quad (20)$$

5. calculate the difference between the predicted and the known values using:

$$\begin{aligned} \mathcal{L}_p(G(^{(l)}\mathbf{z}^n)) = & \sum_{k=n-m}^{n-1} (\tilde{\alpha}^k - ^{(l)}\alpha^k)^T \mathbf{W}_{\alpha} (\tilde{\alpha}^k - ^{(l)}\alpha^k) \\ & + \sum_{k=n-m}^{n-1} \zeta_{\mu} (\tilde{\mu}^k - ^{(l)}\mu^k)^T \mathbf{W}_{\mu} (\tilde{\mu}^k - ^{(l)}\mu^k), \end{aligned} \quad (21)$$

where $\mathbf{W}_{\alpha}$ is a square matrix of size N_{POD} whose diagonal values are equal to the weights that govern the relative importance of the POD coefficients. All other entries are zero. The weights could be based on the singular values if a POD method is used for compression, for example. $\mathbf{W}_{\mu}$ is a square matrix of size N_{μ} whose diagonal values are equal to the model parameter weights, and the scalar ζ_{μ} controls how much importance is given to the model parameters compared to the compressed variables. It is worth mentioning that the goal in each time iteration is to predict a new time level n , hence the POD coefficients α^n and model parameters μ^n of this time level are not in the loss function.

6. compute the gradient of the loss $\mathcal{L}_p$ with regard to the latent variables $^{(l)}\mathbf{z}^n$ (by automatic differentiation), and minimize $\mathcal{L}_p$ in the gradient direction leading to an updated set of latent variables $^{(l+1)}\mathbf{z}^n$;
7. increment the optimization iteration counter by one ($l \leftarrow l + 1$);
8. repeat steps 4 to 7 until convergence is reached;
9. save the converged latent variables as $\mathbf{z}^n$ (note, no optimization iteration index) and use it to initialize the latent variables at the next time level, $^{(0)}\mathbf{z}^{n+1} = \mathbf{z}^n$. Add the predicted time step n to the known solutions $\tilde{\alpha}^n = \alpha^n$;
10. update the time iteration counter by one ($n \leftarrow n + 1$);

11. return to step 3 and repeat until the final time level is achieved.

It is important to note that the gradient of Eq. (6) can be computed through automatic differentiation [72–74]. This mean the loss Eq. (6) can be minimized by using back-propagation through the generator, using the same techniques applied during the GAN training.

Appendix D. DA-PredGAN algorithm

We can use the following steps to perform the data assimilation through time:

1. Use the prediction process in Appendix C with Eq. (6) to march forward in time. Start with a initial estimate of the parameters $\{\tilde{\alpha}^k\}_{k=1}^m$ for the first m time steps and $\tilde{\mu}^k$ over all simulation time. This yields an initial approximation for α^n at all n time levels.
2. Time march backwards in time optimizing:

$$\mathbf{z}^n = \arg \min_{\mathbf{z}^n} \mathcal{L}_{da,b}(G(\mathbf{z}^n)), \quad (22)$$

$$\begin{aligned} \mathcal{L}_{da,b}(G(\mathbf{z}^n)) = & \sum_{k=n+1}^{n+m} (\tilde{\alpha}^k - \alpha^k)^T \mathbf{W}_\alpha (\tilde{\alpha}^k - \alpha^k) \\ & + \sum_{k=n+1}^{n+m} \zeta_\mu (\tilde{\mu}^k - \mu^k)^T \mathbf{W}_\mu (\tilde{\mu}^k - \mu^k) \\ & + \sum_{k=n+1}^{n+m} \zeta_{obs} (\mathbf{d}^k - \mathbf{d}_{obs}^k)^T \mathbf{W}_{obs}^k (\mathbf{d}^k - \mathbf{d}_{obs}^k), \end{aligned} \quad (23)$$

beginning from the m final time levels, which are obtained from the previous iteration, we will employ the same prediction process outlined in Appendix C. However, we will modify the functional to Eq. (23) and decrement the time iterations counter ($n \leftarrow n - 1$) instead of increasing it. The subscript b in $\mathcal{L}_{da}$ indicates that this loss function is applied in the backward march. This approach aims to march backwards in time while attempting to match the observations using the observed data mismatch component of the functional (the last term on the right-hand side of Eq. (23)). During the time march update the model parameters using the newly predicted time step $\tilde{\mu}^n = \mu^n$.

3. Time march forward in time optimizing:

$$\mathbf{z}^n = \arg \min_{\mathbf{z}^n} \mathcal{L}_{da,f}(G(\mathbf{z}^n)), \quad (24)$$

$$\begin{aligned} \mathcal{L}_{da,f}(G(\mathbf{z}^n)) = & \sum_{k=n-m}^{n-1} (\tilde{\alpha}^k - \alpha^k)^T \mathbf{W}_\alpha (\tilde{\alpha}^k - \alpha^k) \\ & + \sum_{k=n-m}^{n-1} \zeta_\mu (\tilde{\mu}^k - \mu^k)^T \mathbf{W}_\mu (\tilde{\mu}^k - \mu^k) \\ & + \sum_{k=n-m}^{n-1} \zeta_{obs} (\mathbf{d}^k - \mathbf{d}_{obs}^k)^T \mathbf{W}_{obs}^k (\mathbf{d}^k - \mathbf{d}_{obs}^k), \end{aligned} \quad (25)$$

beginning from the m first time levels, which are obtained from the backward iteration, we will employ the same prediction process outlined in Appendix C. However, we will modify the functional to Eq. (25). The subscript f in $\mathcal{L}_{da}$ indicates that this loss function is applied in the forward march. This approach aims to advance in time while attempting to match the observations using the observed data mismatch component of the functional (the last term on the right-hand side of Eq. (25)). During the time march update the model parameters using the newly predicted time step $\tilde{\mu}^n = \mu^n$.

4. Continue time stepping backwards till the start of time and then forwards to the end of time using Eqs. (23), (25), until the algorithm has converged.
5. (Optional - most used for parametric problems) Upon convergence, execute a final forward march using Eq. (6) and the prediction process in Appendix C. Use the most recently computed parameters $\{\tilde{\alpha}^k\}_{k=1}^m$ and $\tilde{\mu}^k$ for this.

D.1. Relaxation factor and convergence criteria

To ensure stability during the forward and backward marching processes, a relaxation parameter is introduced into the resultant latent vector, $\mathbf{z}^n$, at each time step. After executing a forward and backward time marches according to Eqs. (23) and (25), the latent vector is relaxed using the equation:

$$\mathbf{z}^n = (1 - r^j) \mathbf{z}^{n-1} + r^j \hat{\mathbf{z}}^n \quad (26)$$

where $\hat{\mathbf{z}}^n$ denotes the latent vector obtained at each time iteration by minimizing Eqs. (25) or (23). The variable j signifies an iteration encompassing both a forward and backward march, while r^j denotes the relaxation factor applied during iteration j . It is crucial to note that each r^j is consistently applied across all time iterations n within the forward and backward marches.

The relaxation factor, r^j , is initiated at a value of one for the data assimilation process. Should the average data mismatch at iteration j exceed that of $j - 1$, the value of r^j is halved, $r^j = r^j/2$, and iteration j is repeated. Conversely, if the average data mismatch at iteration j is lower than at $j - 1$, the process progresses to iteration $j + 1$ with an updated factor of $r^{j+1} = 1.5r^j$. It is essential to ensure the relaxation factor does not exceed one; hence if $r^{j+1} > 1$ then $r^{j+1} = 1$. The criterion for convergence is set as $r^j < 0.01$, signalling the termination of the data assimilation process.

D.2. Weighting terms

We can calculate the weighting terms in the loss function of Eqs. (23) and (25) as

$$\zeta_{obs} = \hat{\zeta}_{obs} \left(\frac{\Delta\alpha}{\Delta d} \right)^2 \left(\frac{m \sum_{i=1}^{N_{POD}} (w_\alpha)_{ii}}{\sum_k \sum_{i=1}^{N_c} (w_{obs})_{ii}^k} \right), \quad (27)$$

where the tuning parameter $\hat{\zeta}_{obs}$ is set to 10 in this work. $\Delta\alpha$ and Δd are the ranges of the compressed POD coefficients and the observed data, respectively. $(w_\alpha)_{ii}$ are the terms on the main diagonal of $\mathbf{W}_\alpha$, and $(w_{obs})_{ii}^k$ are the terms on the main diagonal of $\mathbf{W}_{obs}^k$.

$$\zeta_\mu = \hat{\zeta}_\mu \left(\frac{\Delta\alpha}{\Delta\mu} \right)^2 \left(\frac{\sum_{i=1}^{N_{POD}} (w_\mu)_{ii}}{\sum_{i=1}^{N_\mu} (w_\mu)_{ii}} \right), \quad (28)$$

where $\Delta\mu$ denotes the range of the model parameters, $(w_\mu)_{ii}$ are the terms on the main diagonal of $\mathbf{W}_\mu$, and $\hat{\zeta}_\mu$ is a tuning parameter. ζ_μ controls how rapidly the parameters μ_j^k changes during the data assimilation process. We decided to dynamically update ζ_μ , in order to change μ^k more quickly at the start of the data assimilation process and more slowly when the process is near convergence. Therefore, we begin with $\hat{\zeta}_\mu = 10^{-4}$ and increase it by a factor of 1.2 after each forward-backward iteration. For the prediction, we use $\hat{\zeta}_\mu = 10^{-2}$. We use in this work the same values for the weighting terms as in Silva et al. [49].

Appendix E. GAN evaluation

The proposed evaluation metric (Eq. (14)) uses the fact that the output of the generator is a spatio-temporal map (Eq. (4)) that has semantically different axes with one being the time dimension, and the other being the POD coefficients (or autoencoder latent values) and the model parameters. We could expect that if we take the mean

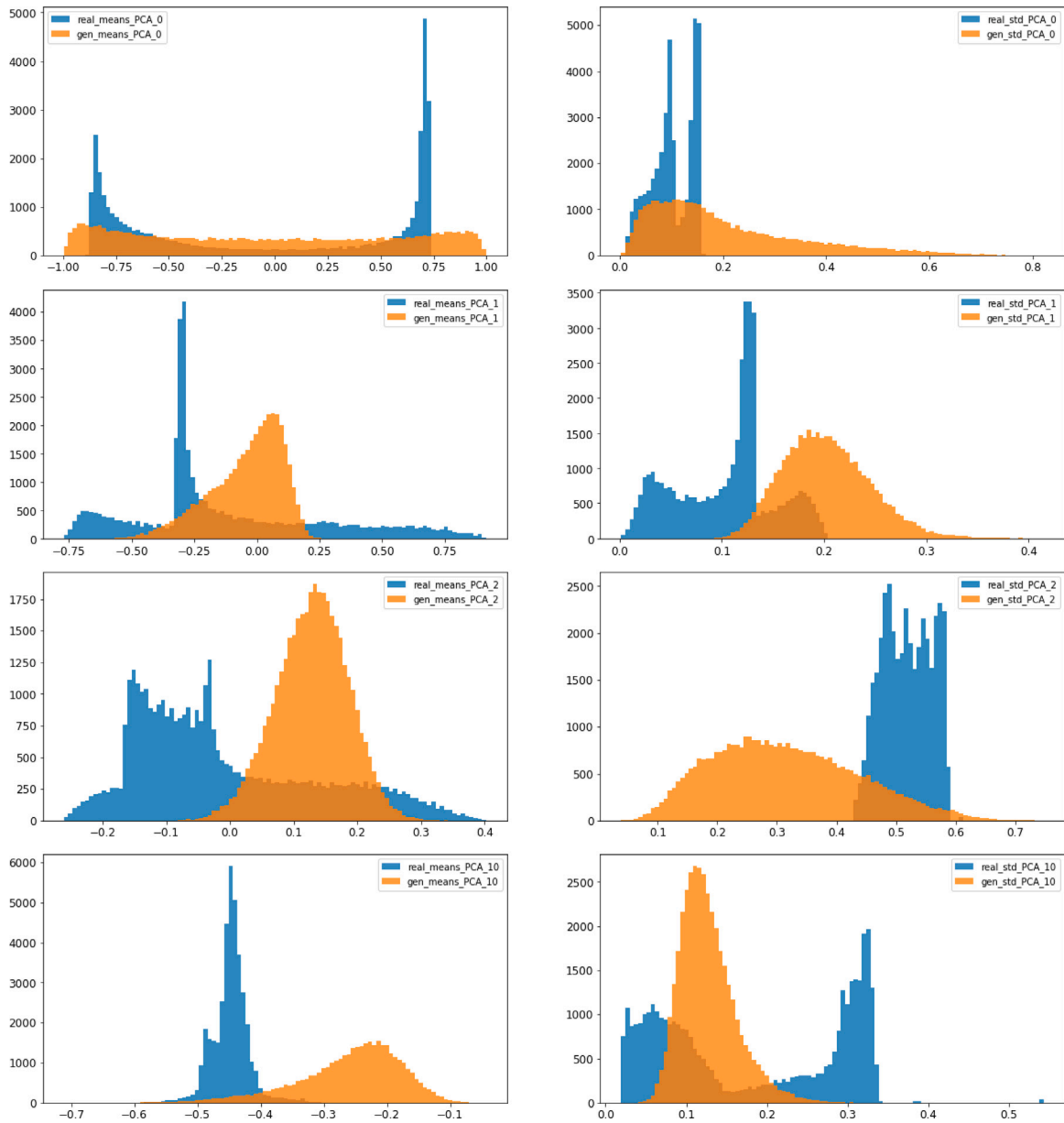


Fig. 14. DCGAN first epoch: POD coefficients mean (first column) and standard deviation (second column). Blue and orange are the distributions from the real and generated data, respectively. Each line represents a different POD coefficient.

and standard deviation from each POD coefficient (one column) from one spatio-temporal map, the distributions of these means and standard deviations over all training spatio-temporal maps will have distinct values representing the characteristics of each POD coefficient. We can see this from Figs. 14, 15, 16 and 17, where we show the distribution of the POD coefficients mean and standard deviation for the real and generated data. We can see that during the training process the generated means and standard deviation distributions and the real means and standard deviations distributions become indistinguishable from each other. Therefore, we calculate the evaluation metric (H^2) for all POD coefficients mean and standard deviation distributions, then we average them to form the final score.

The results from Figs. 14, 15, 16 and 17 correlate with the values of the proposed evaluation metric (H^2) shown in Fig. 8. We can notice that after the first epoch the generated distributions from the WGAN-GP enclose better the real distributions than the generated distribution from the DCGAN. Hence the WGAN-GP starts with a higher evaluation metric (H^2) than the DCGAN, as we can see in Fig. 8. However, the

DCGAN outperforms the WGAN-GP when we look the final results in Fig. 8. We can also see this in Figs. 15 and 17 (the bottom left plot, for example). They show that the result from the WGAN-GP does not match the real distribution as well as the DCGAN.

We chose to generate the distributions of the mean and standard deviation of the POD coefficients since in each sample we expect a strong correlation in time, but not among the coefficients (using POD the directions are perpendicular to each other). Therefore, as each sample represents roughly a day, we are looking the average and the corresponding variation of the POD coefficients on each day.

Appendix F. Markov chain Monte Carlo

Markov chain Monte Carlo (MCMC) is a class of algorithms for sampling a target probability distribution that is known up to a normalizing constant [10,13,17,18,21,59]. The most popular MCMC algorithm is the Metropolis–Hasting algorithm [81,82], which we detail here.

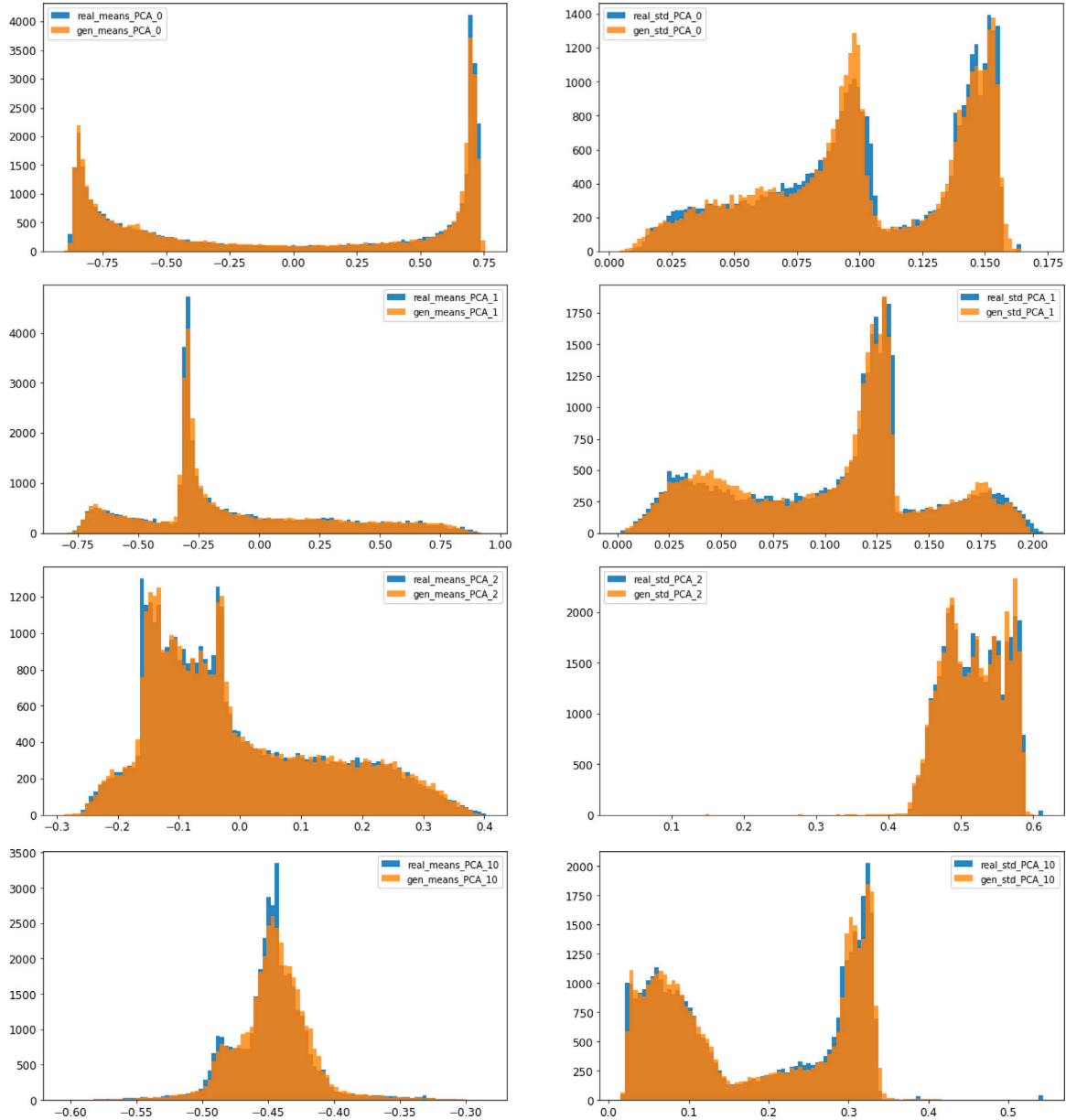


Fig. 15. DCGAN final epoch: POD coefficients mean (first column) and standard deviation (second column). Blue and orange are the distributions from the real and generated data, respectively. Each line represents a different POD coefficient.

Let $\pi(\mathbf{x})$ be the target probability density function we want to sample, and $q(\mathbf{x}^\ell, \hat{\mathbf{x}}^{\ell+1})$ a proposal distribution. The latter represents the probability of proposing a transition from the state $\mathbf{x}^\ell$ in the chain to $\hat{\mathbf{x}}^{\ell+1}$. The new proposed state $\hat{\mathbf{x}}^{\ell+1}$ is sampled from $q(\mathbf{x}^\ell, \hat{\mathbf{x}}^{\ell+1})$.

1. Set $\ell = 0$ and choose the initial state $\mathbf{x}^0$.
2. Propose a new state $\hat{\mathbf{x}}^{\ell+1}$ from $q(\mathbf{x}^\ell, \hat{\mathbf{x}}^{\ell+1})$.
3. Compute the probability of accepting the proposed state

$$P(\mathbf{x}^\ell, \hat{\mathbf{x}}^{\ell+1}) = \min \left\{ 1, \frac{q(\hat{\mathbf{x}}^{\ell+1}, \mathbf{x}^\ell) \pi(\hat{\mathbf{x}}^{\ell+1})}{q(\mathbf{x}^\ell, \hat{\mathbf{x}}^{\ell+1}) \pi(\mathbf{x}^\ell)} \right\}, \quad (29)$$

if we chose

$$q(\hat{\mathbf{x}}^{\ell+1}, \mathbf{x}^\ell) = q(\mathbf{x}^\ell, \hat{\mathbf{x}}^{\ell+1}), \quad (30)$$

then

$$P(\mathbf{x}^\ell, \hat{\mathbf{x}}^{\ell+1}) = \min \left\{ 1, \frac{\pi(\hat{\mathbf{x}}^{\ell+1})}{\pi(\mathbf{x}^\ell)} \right\}. \quad (31)$$

4. Sample y from an uniform distribution $U(0, 1)$.
5. If $y < P(\mathbf{x}^\ell, \hat{\mathbf{x}}^{\ell+1})$, accept the new state $\mathbf{x}^{\ell+1} = \hat{\mathbf{x}}^{\ell+1}$. Otherwise, repeat the current state $\mathbf{x}^{\ell+1} = \mathbf{x}^\ell$.
6. Set $\ell = \ell + 1$ and return to step 2. Stop when the chain has converged or the maximum number of samples has been achieved.

As long as the proposal mechanism is such that it is possible to reach any $\pi(\mathbf{x}) \neq 0$ in a finite number of transitions, the states in the chain will converge to $\pi(\mathbf{x})$.

F.1. Posterior distribution

We are interested in the probability of the model parameters μ given the observed data $\mathbf{d}_{obs}$. Using Bayes Theorem, we can write

$$p(\mu | \mathbf{d}_{obs}) \approx p(\mathbf{d}_{obs} | \mu) p(\mu). \quad (32)$$

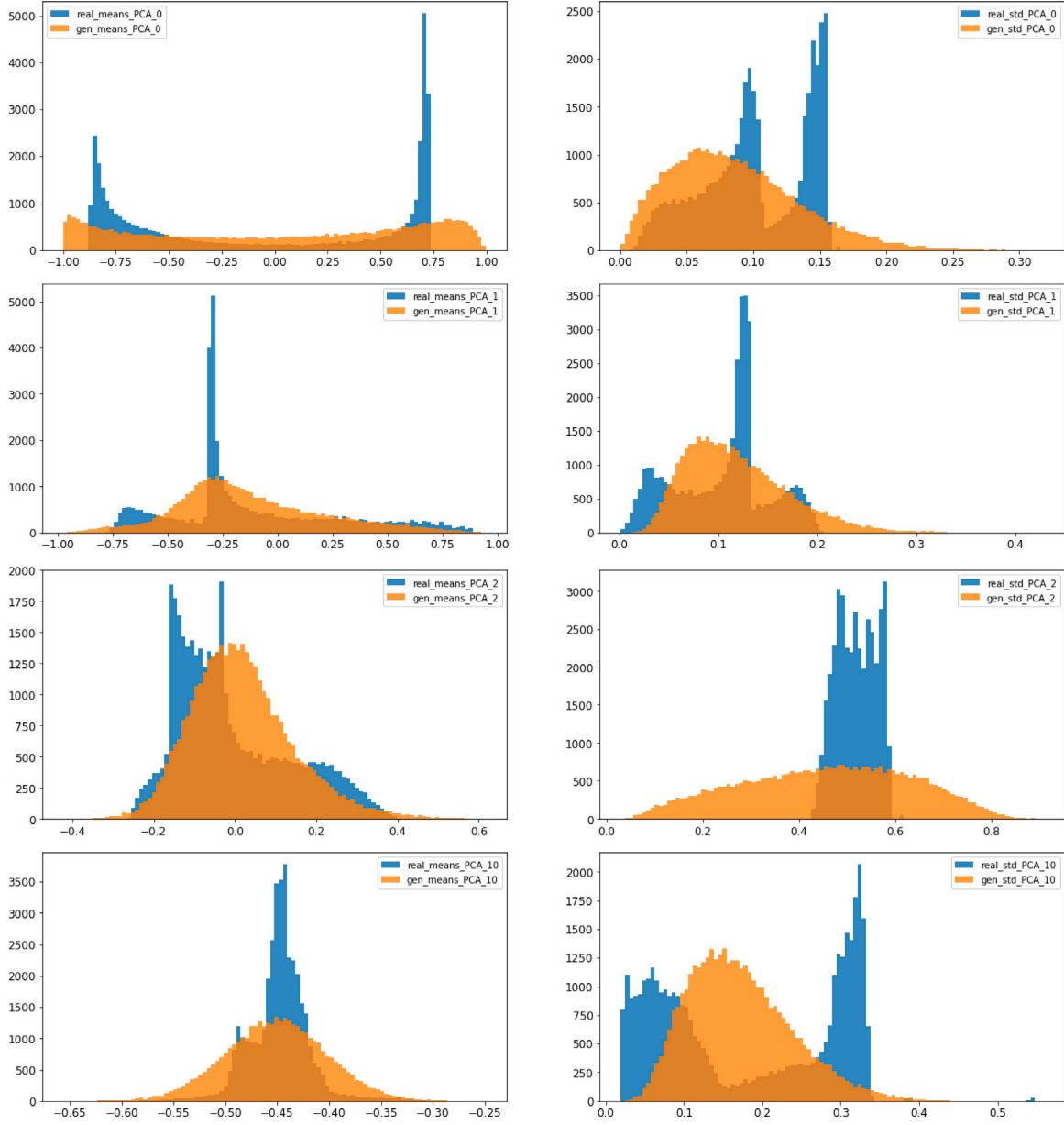


Fig. 16. WGAN-GP first epoch: POD coefficients mean (first column) and standard deviation (second column). Blue and orange are the distributions from the real and generated data, respectively. Each line represents a different POD coefficient.

As for the UQ-PredGAN algorithm (Section 4.5), we approximate the prior $p(\mu)$ and likelihood $p(d_{obs}|\mu)$ as normal distributions. $\mathcal{N}(\bar{\mu}, C_\mu)$ is the prior distribution, where C_μ is the covariance matrix of the model parameters, and $\bar{\mu}$ is the model parameter mean vector. For the observed data, the measurement error ($\epsilon = d_{obs} - d$) is sampled from $\mathcal{N}(0, C_d)$, where C_d is the covariance matrix of the measurement error. Then we can write the posterior distribution as

$$\pi(\mu) = p(\mu|d_{obs}) = a \exp(-O(\mu)), \quad (33)$$

where a is a normalizing constant and $O(\mu)$ is defined by

$$O(\mu) = (\mu - \bar{\mu})^T C_\mu^{-1} (\mu - \bar{\mu}) + (d_{obs} - d)^T C_d^{-1} (d_{obs} - d). \quad (34)$$

We consider μ , d and d_{obs} along all time steps of the forward simulation. Having the posterior probability defined by Eqs. (33) and (34), we apply the Metropolis–Hasting algorithm described in the previous section to generate the samples.

Appendix G. Hardware configuration

A Linux (Ubuntu 18.04.6 LTS) workstation was used to train the machine learning models and run all the numerical simulations. Because we had only one GPU available, all the training and runs of the methods presented in this work were performed using the 64 CPU cores. During the training and inference of the machine learning models and during the data assimilation process for the UQ-PredGAN, we use all the cores available on the CPU (when supported by the TensorFlow

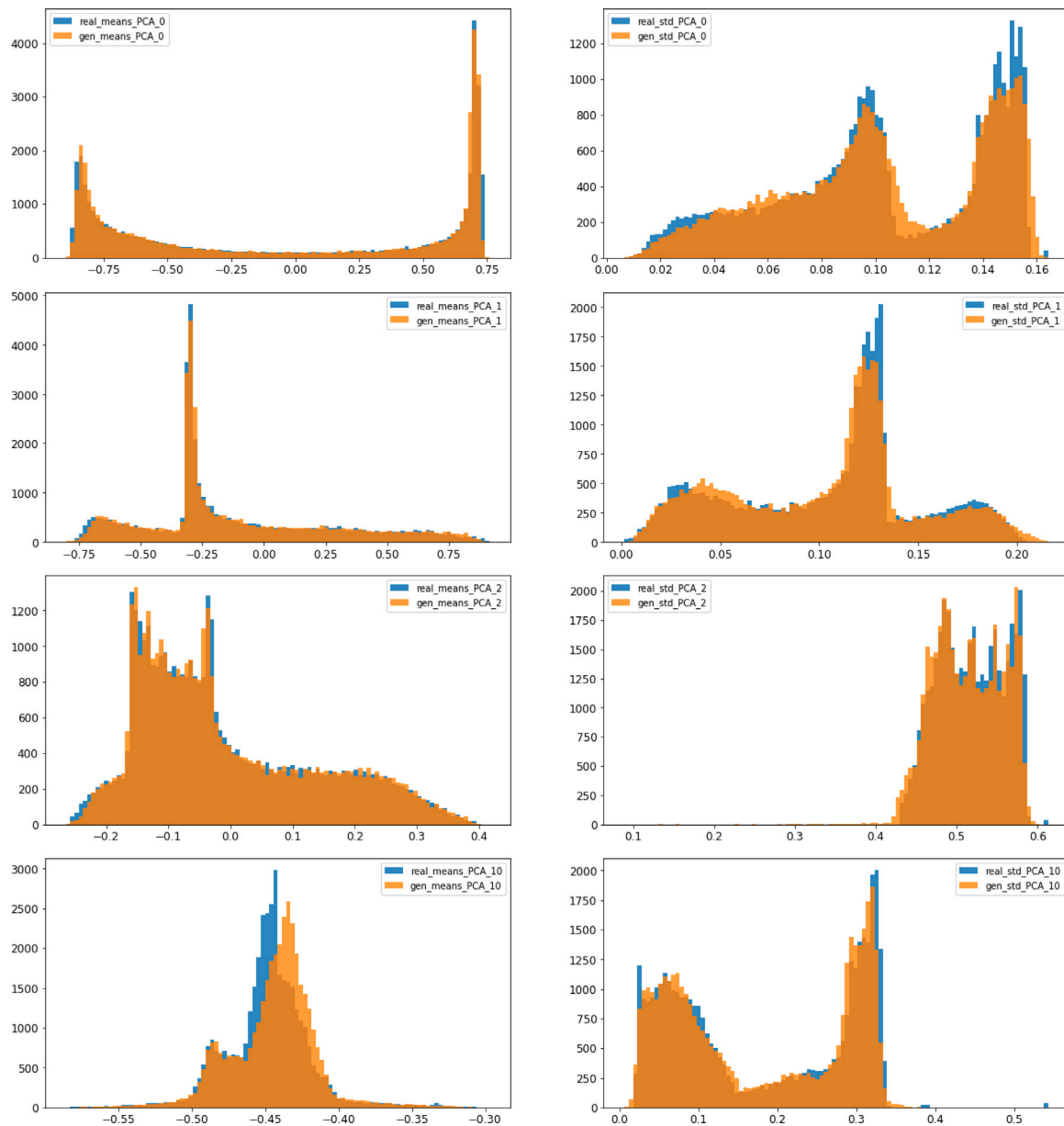


Fig. 17. WGAN-GP final epoch: POD coefficients mean (first column) and standard deviation (second column). Blue and orange are the distributions from the real and generated data, respectively. Each line represents a different POD coefficient.

Table 2

Hardware configuration.

Description	Quantity
16GB DDR4 3200 MHz RAM Memory	16
Samsung 970 EVO PLUS 2TB SSD/Solid State Drive	1
Seagate IronWolf PRO 4TB SATA HDD/Hard Drive	3
Nvidia Quadro RTX 4000 Video Card	1
AMD 32 Core 2nd Gen EPYC 7452 CPU/Processor	2
AMD EPYC 7000 EATX Gigabit Server Motherboard	1

implementation). Table 2 shows the hardware configuration. We use the machine presented in Table 2 to run all experiments presented in this work.

References

- [1] G.H. Golub, J.M. Ortega, et al., *Scientific Computing and Differential Equations: An Introduction to Numerical Methods*, Academic Press, 1992.
- [2] W.F. Ames, *Numerical Methods for Partial Differential Equations*, Academic Press, 2014.
- [3] G. Rozza, D.B.P. Huynh, A.T. Patera, Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations: application to transport and continuum mechanics, *Arch. Comput. Methods Eng.* 15 (3) (2008) 229–275.
- [4] M.A. Cardoso, L.J. Durlafsky, P. Sarma, Development and application of reduced-order modeling procedures for subsurface flow simulation, *Internat. J. Numer. Methods Engrg.* 77 (9) (2009) 1322–1350.
- [5] T. Kadeethum, D. O'Malley, J.N. Fuhg, Y. Choi, J. Lee, H.S. Viswanathan, N. Bouklas, A framework for data-driven solution and parameter estimation of PDEs using conditional generative adversarial networks, *Nat. Comput. Sci* 1 (12) (2021) 819–829.
- [6] R. Vinuesa, S.L. Brunton, Enhancing computational fluid dynamics with machine learning, *Nat. Comput. Sci* 2 (6) (2022) 358–366.
- [7] C.E. Heaney, X. Liu, H. Go, Z. Wolffs, P. Salinas, I.M. Navon, C.C. Pain, Extending the capabilities of data-driven reduced-order models to make predictions for unseen scenarios: Applied to flow around buildings, *Front. Phys* 10 (2022) 910381.
- [8] B. Sanderse, Non-linearly stable reduced-order models for incompressible flow with energy-conserving finite volume methods, *J. Comput. Phys.* 421 (2020) 109736.

- [9] D. Degen, M. Cacace, F. Wellmann, 3D multi-physics uncertainty quantification using physics-based machine learning, *Sci. Rep.* 12 (1) (2022) 17491.
- [10] N. Liu, D.S. Oliver, et al., Evaluation of Monte Carlo methods for assessing uncertainty, *SPE J.* 8 (02) (2003) 188–195.
- [11] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, *Adv. Neural Inform. Process. Syst.* 30 (2017).
- [12] S. Cheng, C. Quilodr n-Casas, S. Ouala, A. Farchi, C. Liu, P. Tando, R. Fablet, D. Lucor, B. Iooss, J. Brajard, et al., Machine learning with data assimilation and uncertainty quantification for dynamical systems: a review, *IEEE/CAA J. Autom. Sin.* 10 (6) (2023) 1361–1387.
- [13] A.A. Emerick, A.C. Reynolds, Investigation on the sampling performance of ensemble-based methods with a simple reservoir model, *Comput. Geosci.* 17 (2) (2013) 325–350.
- [14] B. Sudret, S. Marelli, J. Wiart, Surrogate models for uncertainty quantification: An overview, in: 2017 11th European Conference on Antennas and Propagation, EUCAP, IEEE, 2017, pp. 793–797.
- [15] D.G. Cacuci, BERRU Predictive Modeling: Best Estimate Results with Reduced Uncertainties, Springer, 2019.
- [16] D.S. Oliver, L.B. Cunha, A.C. Reynolds, Markov chain Monte Carlo methods for conditioning a permeability field to pressure data, *Math. Geology* 29 (1) (1997) 61–91.
- [17] D.S. Oliver, Y. Chen, Recent progress on reservoir history matching: a review, *Comput. Geosci.* 15 (1) (2011) 185–221.
- [18] A.S. Stordal, G. N vdal, A modified randomized maximum likelihood for improved Bayesian history matching, *Comput. Geosci.* 22 (1) (2018) 29–41.
- [19] P.K. Kitanidis, Quasi-linear geostatistical theory for inversing, *Water Resour. Res.* 31 (10) (1995) 2411–2419.
- [20] D.S. Oliver, N. He, A.C. Reynolds, Conditioning permeability fields to pressure data, in: ECMOR V-5th European Conference on the Mathematics of Oil Recovery, European Association of Geoscientists & Engineers, 1996, pp. cp–101.
- [21] J.M. Bardsley, A. Solonen, H. Haario, M. Laine, Randomize-then-optimize: A method for sampling from posterior distributions in nonlinear inverse problems, *SIAM J. Sci. Comput.* 36 (4) (2014) A1895–A1910.
- [22] 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.
- [23] R. Lopez, E. Balsa-Canto, E. O nate, Neural networks for variational problems in engineering, *Internat. J. Numer. Methods Engrg.* 75 (11) (2008) 1341–1360.
- [24] X. Guo, W. Li, F. Iorio, Convolutional neural networks for steady flow approximation, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 481–490.
- [25] 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.
- [26] Z. Long, Y. Lu, X. Ma, B. Dong, PDE-Net: Learning PDEs from data, in: International Conference on Machine Learning, PMLR, 2018, pp. 3208–3216.
- [27] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, P. Battaglia, Learning to simulate complex physics with graph networks, in: International Conference on Machine Learning, PMLR, 2020, pp. 8459–8468.
- [28] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, P.W. Battaglia, Learning mesh-based simulation with graph networks, 2020, arXiv preprint arXiv:2010.03409.
- [29] R. Wang, K. Kashinath, M. Mustafa, A. Albert, R. Yu, Towards physics-informed deep learning for turbulent flow prediction, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1457–1466.
- [30] W. Chen, Q. Wang, J.S. Hesthaven, C. Zhang, Physics-informed machine learning for reduced-order modeling of nonlinear problems, *J. Comput. Phys.* 446 (2021) 110666.
- [31] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, 2020, arXiv preprint arXiv:2010.08895.
- [32] Y. Xie, E. Franz, M. Chu, N. Thuerey, tempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow, *ACM Trans. Graph.* 37 (4) (2018) 1–15.
- [33] Z. Zhong, A.Y. Sun, H. Jeong, Predicting CO₂ plume migration in heterogeneous formations using conditional deep convolutional generative adversarial network, *Water Resour. Res.* 55 (7) (2019) 5830–5851.
- [34] M. Cheng, F. Fang, C.C. Pain, I. Navon, Data-driven modelling of nonlinear spatio-temporal fluid flows using a deep convolutional generative adversarial network, *Comput. Methods Appl. Mech. Engrg.* 365 (2020) 113000.
- [35] C. Quilodr n-Casas, V.L. Silva, R. Arcucci, C.E. Heaney, Y. Guo, C.C. Pain, Digital twins based on bidirectional LSTM and GAN for modelling the COVID-19 pandemic, *Neurocomputing* 470 (2022) 11–28.
- [36] S. Ravuri, K. Lenc, M. Willson, D. Kangin, R. Lam, P. Mirowski, M. Fitzsimons, M. Athanassiadou, S. Kashem, S. Madge, et al., Skillful precipitation nowcasting using deep generative models of radar, *Nature* 597 (2021) 672–677.
- [37] D. Xiao, F. Fang, A. Buchan, C. Pain, I. Navon, A. Muggeridge, Non-intrusive reduced order modelling of the Navier-Stokes equations, *Comput. Methods Appl. Mech. Engrg.* 293 (2015) 522–541.
- [38] J.S. Hesthaven, S. Ubbiali, Non-intrusive reduced order modeling of nonlinear problems using neural networks, *J. Comput. Phys.* 363 (2018) 55–78.
- [39] C. Quilodr n Casas, R. Arcucci, P. Wu, C. Pain, Y.-K. Guo, A reduced order deep data assimilation model, *Physica D* (2020) 132615.
- [40] P. Holmes, J.L. Lumley, G. Berkooz, C.W. Rowley, Proper orthogonal decomposition, in: Turbulence, Coherent Structures, Dynamical Systems and Symmetry, second ed., in: Cambridge Monographs on Mechanics, Cambridge University Press, 2012, pp. 68–105.
- [41] C. Quilodr n Casas, R. Arcucci, Y. Guo, Urban air pollution forecasts generated from latent space representations, in: ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations, 2020.
- [42] T.R.F. Phillips, C.E. Heaney, P.N. Smith, C.C. Pain, An autoencoder-based reduced-order model for eigenvalue problems with application to neutron diffusion, *Internat. J. Numer. Methods Engrg.* 122 (15) (2021) 3780–3811.
- [43] E. Kaiser, B. Noack, L. Cordier, A. Spohn, M. Segond, M. Abel, G. Daviller, J. O st, S. Krajnovi , R. Niven, Cluster-based reduced-order modelling of a mixing layer, *J. Fluid Mech.* 754 (2014) 365–414.
- [44] Z. Wang, D. Xiao, F. Fang, R. Govindan, C.C. Pain, Y. Guo, Model identification of reduced order fluid dynamics systems using deep learning, *Internat. J. Numer. Methods Fluids* 86 (4) (2018) 255–268.
- [45] S.E. Ahmed, S.M. Rahman, O. San, A. Rasheed, I.M. Navon, Memory embedded non-intrusive reduced order modeling of non-ergodic flows, *Phys. Fluids* 31 (12) (2019) 126602.
- [46] M. Guo, J.S. Hesthaven, Reduced order modeling for nonlinear structural analysis using Gaussian process regression, *Comput. Methods Appl. Mech. Engrg.* 341 (2018) 807–826.
- [47] M. Jolaade, V.L. Silva, C.E. Heaney, C.C. Pain, Generative networks applied to model fluid flows, in: International Conference on Computational Science, Springer, 2022, pp. 742–755.
- [48] D.V. Patel, A.A. Oberai, GAN-based priors for quantifying uncertainty in supervised learning, *SIAM/ASA J. Uncert. Quant* 9 (3) (2021) 1314–1343.
- [49] V.L. Silva, C.E. Heaney, Y. Li, C.C. Pain, Data assimilation predictive GAN (DA-PredGAN) applied to a spatio-temporal compartmental model in epidemiology, *J. Sci. Comput.* 94 (2023) 25.
- [50] A. Tarantola, Inverse Problem Theory and Methods for Model Parameter Estimation, SIAM, 2005.
- [51] A.A. Emerick, A.C. Reynolds, Ensemble smoother with multiple data assimilation, *Comput. Geosci.* 55 (2013) 3–15.
- [52] V.L.S. Silva, A.A. Emerick, P. Couto, J.L.D. Alves, History matching and production optimization under uncertainties—Application of closed-loop reservoir management, *J. Pet. Sci. Eng.* 157 (2017) 860–874.
- [53] M. Tang, Y. Liu, L.J. Durlinsky, A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems, *J. Comput. Phys.* (2020) 109456.
- [54] K. Fossom, S. Alyaev, J. Tveranger, A.H. Elsheikh, Verification of a real-time ensemble-based method for updating earth model based on GAN, *J. Comput. Sci.* 65 (2022) 101876.
- [55] L. Mosser, O. Dubrule, M.J. Blunt, Deepflow: history matching in the space of deep generative models, 2019, arXiv preprint arXiv:1905.05749.
- [56] B. Kang, J. Choe, Uncertainty quantification of channel reservoirs assisted by cluster analysis and deep convolutional generative adversarial networks, *J. Pet. Sci. Eng.* 187 (2020) 106742.
- [57] S.M. Razak, B. Jafarpour, History matching with generative adversarial networks, in: ECMOR XVII, 2020, (1) European Association of Geoscientists & Engineers, 2020, pp. 1–17.
- [58] S.W. Canchumuni, J.D. Castro, J. Potratz, A.A. Emerick, M.A.C. Pacheco, Recent developments combining ensemble smoother and deep generative networks for facies history matching, *Comput. Geosci.* 25 (1) (2021) 433–466.
- [59] N.T. M ckke, B. Sanderse, S.M. Boht , C.W. Oosterlee, Markov chain generative adversarial neural networks for solving Bayesian inverse problems in physics applications, *Comput. Math. Appl.* 147 (2023) 278–299.
- [60] R.M. Anderson, B. Anderson, R.M. May, Infectious Diseases of Humans: Dynamics and Control, Oxford University Press, 1992.
- [61] O.N. Bj rnstad, Epidemics: Models and Data Using R, Springer, 2018.
- [62] O. Bj rnstad, K. Shea, M. Krzywinski, N. Altman, The SEIRS model for infectious disease dynamics, *Nat. Meth.* 17 (6) (2020) 557–558.
- [63] F. Gugole, L.E. Coffeng, W. Edeling, B. Sanderse, S.J. De Vlas, D. Crommelin, Uncertainty quantification and sensitivity analysis of COVID-19 exit strategies in an individual-based transmission model, *PLoS Comput. Biol.* 17 (9) (2021) e1009355.
- [64] K. Dietz, The estimation of the basic reproduction number for infectious diseases, *Stat. Methods Med. Res.* 2 (1) (1993) 23–41.
- [65] H.W. Hethcote, The mathematics of infectious diseases, *SIAM Rev.* 42 (4) (2000) 599–653.
- [66] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, 2015, arXiv preprint arXiv:1511.06434.
- [67] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A.C. Courville, Improved training of Wasserstein GANs, *Adv. Neural Inform. Process. Syst.* 30 (2017).

- [68] T. Karras, S. Laine, T. Aila, A Style-Based Generator Architecture for Generative Adversarial Networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 4401–4410.
- [69] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, Deep Learning, vol. 1, (2) MIT press Cambridge, 2016.
- [70] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.
- [71] I. Goodfellow, NIPS 2016 tutorial: Generative adversarial networks, 2016, arXiv preprint arXiv:1701.00160.
- [72] R.E. Wengert, A simple automatic derivative evaluation program, Commun. ACM 7 (8) (1964) 463–464.
- [73] S. Linnainmaa, Taylor expansion of the accumulated rounding error, BIT Numer. Math. 16 (2) (1976) 146–160.
- [74] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, J.M. Siskind, Automatic differentiation in machine learning: a survey, J. Mach. Learn. Res. 18 (1) (2017) 5595–5637.
- [75] D.S. Oliver, A.C. Reynolds, N. Liu, Inverse Theory for Petroleum Reservoir Characterization and History Matching, Cambridge University Press, 2008.
- [76] P. Grnarova, K.Y. Levy, A. Lucchi, N. Perraudin, I. Goodfellow, T. Hofmann, A. Krause, A domain agnostic measure for monitoring and evaluating GANs, Adv. Neural Inf. Process. Syst. 32 (2019).
- [77] L. Le Cam, L.M. LeCam, G.L. Yang, Asymptotics in Statistics: Some Basic Concepts, Springer Science & Business Media, 2000.
- [78] A.L. Gibbs, F.E. Su, On choosing and bounding probability metrics, Int. Statist. Rev 70 (3) (2002) 419–435.
- [79] M. Kuchańczyk, F. Grabowski, T. Lipniacki, Super-spreading events initiated the exponential growth phase of COVID-19 with R_0 higher than initially estimated, R. Soc. Open Sci 7 (9) (2020) 200786.
- [80] A. Sajeeda, B.M. Hossain, Exploring generative adversarial networks and adversarial training, Int. J. Cognit. Comput. Eng 3 (2022) 78–89.
- [81] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, Equation of state calculations by fast computing machines, J. Chem. Phys. 21 (6) (1953) 1087–1092.
- [82] W.K. Hastings, Monte Carlo sampling methods using Markov chains and their applications, Biometrika 57 (1970) 97–109.